NEWCASTLE UNIVERSITY

DEPARTMENT OF COMPUTING SCIENCE

The Robustness of Animated Text CAPTCHAs

By

Mohamad Tayara

The thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

July 2017

Abstract

CAPTCHA is standard security technology that uses AI techniques to tells computer and human apart. The most widely used CAPTCHA are text-based CAPTCHA schemes. The robustness and usability of these CAPTCHAs relies mainly on the segmentation resistance mechanism that provides robustness against individual character recognition attacks. However, many CAPTCHAs have been shown to have critical flaws caused by many exploitable invariants in their design, leaving only a few CAPTCHA schemes resistant to attacks, including ReCAPTCHA and the Wikipedia CAPTCHA.

Therefore, new alternative approaches to add motion to the CAPTCHA are used to add another dimension to the character cracking algorithms by animating the distorted characters and the background, which are also supported by tracking resistance mechanisms that prevent the attacks from identifying the main answer through frame-to-frame attacks. These technologies are used in many of the new CAPTCHA schemes including the Yahoo CAPTCHA, CAPTCHANIM, KillBot CAPTCHAs, non-standard CAPTCHA and NuCAPTCHA.

Our first question: can the animated techniques included in the new CAPTCHA schemes provide the required level of robustness against the attacks? Our examination has shown many of the CAPTCHA schemes that use the animated features can be broken through tracking attacks including the CAPTCHA schemes that uses complicated tracking resistance mechanisms.

The second question: can the segmentation resistance mechanism used in the latest standard text-based CAPTCHA schemes still provide the additional required level of resistance against attacks that are not present missed in animated schemes? Our test against the latest version of ReCAPTCHA and the Wikipedia CAPTCHA exposed vulnerability problems against the novel attacks mechanisms that achieved a high success rate against them.

The third question: how much space is available to design an animated text-based CAPTCHA scheme that could provide a good balance between security and usability? We designed a new animated text-based CAPTCHA using guidelines we designed based on the results of our attacks on standard and animated text-based CAPTCHAs, and we then tested its security and usability to answer this question.

In this thesis, we put forward different approaches to examining the robustness of animated text-based CAPTCHA schemes and other standard text-based CAPTCHA schemes against segmentation and tracking attacks. Our attacks included several methodologies that required thinking skills in order to distinguish the animated text from the other animated noises, including the text distorted by highly tracking resistance mechanisms that displayed them partially as animated segments and which looked similar to noises in other CAPTCHA schemes. These attacks also include novel attack mechanisms and other mechanisms that uses a recognition engine supported by attacking methods that exploit the identified invariants to recognise the connected characters at once. Our attacks also provided a guideline for animated text-based CAPTCHAs that could provide resistance to tracking and segmentation attacks which we designed and tested in terms of security and usability, as mentioned before. Our research also contributes towards providing a toolbox for breaking CAPTCHAs in addition to a list of robustness and usability issues in the current CAPTCHA design that can be used to provide a better understanding of how to design a more resistant CAPTCHA scheme.

Acknowledgment

I would like to thank everyone who supported me and gave me valuable advice throughout my doctoral studies in Newcastle University. I would in particular like to thank my first supervisor Jeff Yan for his guidance and valuable insights, which helped me tremendously in completing this body of work. I would also like to thank my second supervisor Aad van Moorsel for his generous support and guidance and for the reading of my thesis during the final year of my research.

I would like also to extend my thanks to individual people who have been involved in my work including Charles Morisset, Iryna Yevseyeva and James Nicholson who participated in my CAPTCHA pilot test and provided feedback on usability test set-up. I would like also to thank the other participants in the usability test for their co-operation and commitment that contributed to the success of my final research.

Finally, I would like to express my gratitude to my family especially my parents, Abdulla Tayara and Samar Abuseif, as well as my sister Sabah for their endless love, encouragement and support throughout the past four years. I would like also to extend my thanks to other relatives and friends, who provided help and support in many ways, especially Habib Hammam and Ahmad Yousef.

Special thanks to Steven Swift for proofreading my thesis

Table of Contents

List of figures

List of Tables

Chapter 1. Introduction

Computer programs can surpass the human in many tasks. Sometimes, these tasks may turn out to be malicious, such as when the computer performs actions in web applications intended only for humans but with the aim of repeatedly abusing. For example, 107 trillion emails were exchanged in 2010, 89% of which were sent by automated computer programs, most of which were intended as spam[1]. Online voting systems also suffered from malicious programs that aimed to change the results of many online polls[2]. For example, The Times annual poll was targeted by hackers who rearranged the order of the top 100 influential people[3]. The hack resulted in the first letter of the top 21 names spelling out "Marblecake", in reference to an IRC channel. We can counter these malicious attacks by using computer programs that can tell computers and humans apart. For example, we could use challenge-response tests that create a challenge between computers and humans.

The CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is a type of challenge-response test. The test includes a variety of challenges aimed at ensuring that a human generates the response. In each challenge, the computer generates a test that most humans can pass but current automated programs cannot[4]. These challenges should be intuitive for humans but present a problem for the artificial intelligence built into current computer systems.

The level of CAPTCHA challenges is determined by two requirements which can be defined as follows:

- Robustness. The strength of the CAPTCHA resistance against computer programs designed to solve them automatically.

- Usability. The ease with which the human can solve CAPTCHA challenges in terms of effort and time (user friendly).

The most widely used type of CAPTCHA challenge is the text-based CAPTCHA, which is a text-recognition challenge proposed by Noar in 1996[5]. These challenges present text problems in the form of an image of distorted text that includes a set of distorted characters for the user to type in order to pass. We can determine the robustness and usability of these challenges by their distortion technologies. The technologies should provide humans with an easy text to read but a difficult challenge for computers to break and recognise.

Alta-Vista was the first company to include a practical implementation of a text-based CAPTCHA to defend against attacks aimed at altering the ranking of its website. Since then, this type of CAPTCHA has become a widely used technology with many websites developing their own CAPTCHA schemes. The earliest schemes relied on distorting characters to make them hard to recognise by automated attacks. However, research released by a Microsoft team in 2005 showed that these schemes were still vulnerable to attacks. They proved their point by showing that computer programs are very good at recognising individual characters even when they are highly distorted[6]. Thus, newer text CAPTCHAs relied on the segmentation-resistance principle, which is the difficulty in locating characters within the text. The main strength of this principle is centred on its degree of difficulty problem and the expensive cost of its computational solutions.

However, latest research has proved that the segmentation-resistance principle is still vulnerable to attacks against text-based CAPTCHAs. These vulnerability problems are caused by many exploitable invariants that can be strategically used in cryptanalysis to break them. This makes text-based CAPTCHAs vulnerable to attacks that target those invariants. Two categories of invariants have been identified as the following[7]:

- Pixel level invariants that contain the pixel count, colour pattern and the shape variants.

- String level invariants that contain character set, text length and dictionary words strings.

A few text-based CAPTCHAs counter these problems by including high distortion methods although these can also cause these CAPTCHAs to suffer from usability problems, as they make the text more difficult for humans to read.

This has led some developers to rely on another approach that adds motion to the text-based CAPTCHAs. This approach has the potential of providing better space for security and usability through its motion features. The main advantage of this approach is that motion adds another dimension to character cracking algorithms by animating the text and background in the CAPTCHAs. This has the potential to make CAPTCHAs more resistant to computer attacks as there is the need for special tracking systems to detect the characters

in the animated CAPTCHAs[8]. This does not present a problem to humans who are good at perceiving motion[9].

Researchers have also developed several mechanisms to support animated text-based CAPTCHAs. These include animated character distortion mechanisms such as vertical movement, rotation, scale and fade. They also include segmentation-resistance mechanisms developed for animated text-based CAPTCHAs such as the NuCAPTCHA. Other mechanisms were also recently developed to counter tracking attacks that targeted commercial animated CAPTCHA schemes like the NuCAPTCHA. These mechanisms include “emerging image”, “noisy lines” and “noisy characters”. This thesis investigates many of these mechanisms and the segmentation mechanisms included in related designs of standard text-based CAPTCHAs. Still. our focus is on the animated text-based CAPTCHAs.

1.1 The Problem Statement

The robustness of the CAPTCHA. The robustness of CAPTCHA schemes is assessed by examining the methodologies used in the attacks. This assessment plays a major role in exploiting the vulnerability of several CAPTCHA schemes and helps developers improve the security of text-based CAPTCHAs, including animated ones. For example, an examination conducted by Xu et al.[10] against the NuCAPTCHA helped to expose many vulnerability problems with it. This examination also helped to improve the design of the scheme by adding a tracking-resistance mechanism known as “emerging image” [11]. This mechanism was claimed to be the ultimate approach to making a CAPTCHA scheme highly resistant to all attacks known at the time.

Many of the recent examinations caused the developers to release numerous security mechanisms for the animated CAPTCHAs. The robustness level of these mechanisms was even viewed to exceed standard mechanisms presented in non-animated CAPTCHAs. Still, the security of these mechanisms is under question due to the nature of the vulnerability issues of early animated schemes.

These issues also include known segmentation-resistance problems in standard (non-animated) text-based CAPTCHAs. The animated method in this CAPTCHA was an addition to those problems by making the connected characters move individually. Such an addition caused the segmentation attack to be more likely to succeed. The limited number

of examinations of the animated schemes also helped in providing an argument about the security of the animated mechanisms.

To sum up, although recent animated CAPTCHA schemes could resist early attacks, they could be still vulnerable to other attacking methods. Thus, we need to undertake further examination to confirm their resistance to the overall attacking techniques. We also need other examinations of standard mechanisms to confirm that the animated mechanisms could replace them. Both additional examinations are also essential in developing a robust animated CAPTCHA.

1.2 Research Objectives

Our objectives are to find the best design of a robust and usable animated text-based CAPTCHA. Such a design is propelled by the open problem of both segmentation and tracking resistance mechanisms identified in the recent text-based CAPTCHAs and by the available space of development in that area to create a robust version of animated text-based CAPTCHAs.

1.3 Research questions

Three questions have been specified to reflect our research objectives. Those three questions are as follows:

Research question (1):

Can the animated techniques included in the new CAPTCHA schemes provide the required level of robustness against the attacks?

Research question (2):

Can the segmentation resistance mechanism used in the latest standard text-based CAPTCHA schemes still provide the additional required level of resistance against attacks that are not present missed in animated schemes?

Research question (3):

How much space is available to design an animated text-based CAPTCHA scheme that could provide a good balance between security and usability?

1.4 Research Methodology

To achieve our research objectives, we prepared a research methodology that covers all aspects of our research. This can be summarised as follows:

We implemented a series of robust examinations of recent animated and standard (non-animated) text-based CAPTCHA schemes. These examinations were necessary to understand the robust mechanisms in the schemes and determine their vulnerability issues. The results of these examinations are also essential to finding out the best way to include existing robust mechanisms or design new ones. Our examinations included a series of attacks on racking and segmentation resistance mechanisms included in these CAPTCHA schemes.

After that, we used the results of the examinations to learn about the design weaknesses in our targeted schemes and their mechanisms. Based on our learning, we created a design guideline that can provide a secure and usable animated text-based CAPTCHA. We then used this guideline to design a new robust and usable animated text-based CAPTCHA. This new CAPTCHA will be analysed in terms of its robustness by using the attacks we developed in our previous research. We also analysed the usability of the new CAPTCHA. The results of this research will be used to write a paper about this CAPTCHA, including an evaluation of its robustness and usability.

1.5 Contributions

- **A number of segmentation and tracking attacks.** A list of studies of various CAPTCHA schemes that demonstrate our methodology and attack techniques.

- **Toolbox for examining the CAPTCHA scheme.** We developed new techniques for breaking various text-based CAPTCHA schemes. These techniques are simple, easy to use and can act as a toolbox for examining CAPTCHA robustness.

- **A list of robustness and usability issues in the current CAPTCHA design.** We identified many security and usability issues found in the design of the CAPTCHA. Two of these issues are discussed in our thesis.

- **A better understanding of the CAPTCHA design.** We learned a set of lessons that will help to develop guidelines for CAPTCHA design that are resistant to segmentation and tracking attacks.

- **A new CAPTCHA design guideline.** We propose a new general design principle in addition to existing segmentation and tracking resistance principles.

- **A new animated text-based CAPTCHA.** We used our design guidelines to design a new robust text-based CAPTCHA that could resist all the attacks we developed.

1.6 Publications

Tayara, M. "The robustness of tracking resistance text-based CAPTCHAs" publication in preparation

1.7 Thesis Outline

Chapter 2 provides an overview of the CAPTCHA, including its foundation and the major issues related to it. The discussion in this chapter focuses on the current animated text-based CAPTCHAs and included the state of the art designs of those schemes and their strength in terms of security and usability. The discussion also includes previous work that examined the usability and security of those CAPTCHAs.

Chapter 3 summarises our study of the Wikipedia CAPTCHA and examines the security of its "lines as clutter" mechanism. This examination includes an attack that could break the CAPTCHA by segmenting the text and clearing the noise around it. We use the results of this attack to discuss possible defence options of future designs.

Chapter 4 presents our examination of the KillBot professional CAPTCHA, which is an animated text-based CAPTCHA used by the United States Federal Government. This examination includes a series of attacking mechanisms that catch the best recognisable image of each animated character in the text challenge. The results of our attack will be used to provide a better argument about the weaknesses of animated changes used in this scheme. We also use the results to show the ability of computer programs to defeat the human in recognising animated characters.

Chapter 5 presents our examination of CAPTCHANIM, which is an animated text-based CAPTCHA developed by Technion. In particular, our target is a design of this scheme that

combines “animated scaling” and the "horizontal deformation method". In this chapter, we demonstrate how we could break this CAPTCHA by using a similar attack to that used against the KillBot CAPTCHA. We will show how our attack could catch the least distorted image of each animated character in this scheme.

Chapter 6 includes our examination of the 2013 version of ReCAPTCHA that adopts the CCT (crowding character together) segmentation-resistance mechanism. We attacked this CAPTCHA with an automatic-segmentor technique that scans the whole text using a recognition engine with segmentation methods. The results of this attack proved that segmentation-resistance mechanisms could not replace the animated mechanisms.

Chapter 7 presents our examination of the “emerging image” mechanism in Xu's version of the NuCAPTCHA. This mechanism resists tracking attacks by displaying both the main text and background in the form of black segments. We could still break it by means of an attack that detects the segments of text and removes the background noises around them. Our attacks also succeeded in gathering segments of the text to create a recognisable image of it. In this chapter, we will demonstrate how our attack works and show the weaknesses and vulnerability problems of this animated text-based CAPTCHA scheme.

Chapter 8 explains our examination of the “noisy lines” mechanism deployed by Kund's CAPTCHA. This mechanism resists tracking attacks by displaying both text and background using small lines that share the same length. Our attack on this CAPTCHA succeeded in locating the lines related to the text and removing the lines of background around them. In this chapter, we demonstrate how our attack works and show the weaknesses of this CAPTCHA scheme.

Chapter 9 presents our examination of the “noisy characters” mechanism deployed by the Yahoo CAPTCHA. The robustness of this mechanism is dependent on displaying noisy characters in front of the animated text. These noisy characters help in hiding the main text by confusing the computer between them and the characters of that text. The human could still distinguish the main text and noisy characters through their animation in which the noisy characters moved horizontally, unlike the text. Our examination is processed by a tracking attack that distinguishes the main texts from the noisy characters through their movements. This attack then removes the noisy background and segments the main text. In

this chapter, we demonstrate how our attack works and show the weaknesses of this CAPTCHA scheme.

Chapter 10 discussed the issues related to the design of animated and standard text-based CAPTCHA schemes. It also discusses details of the design guidelines of resistance for animated text-based CAPTCHA schemes and related mechanisms.

Chapter 11 presents a new animated text-based CAPTCHA created using the design guidelines presented in the previous chapter. This new CAPTCHA uses a more complex method designed specially to make it more difficult for currently known attacks to break. In this chapter, we demonstrate how this CAPTCHA scheme works and evaluate its robustness and usability in order to detect the limitations of the animated text-based CAPTCHA.

Chapter 12 summarises our work and make conclusions, before we present our future work.

Chapter 2. Literature Review

In this chapter, we discuss previous works mainly on animated text-based CAPTCHA schemes in detail. First, we discuss CAPTCHAs in general as a reverse Turing test, and then we discuss the attacks implemented against it. After that, we discuss text- based CAPTCHAs including previous works undertaken and then we focus on the animated designs of text-based CAPTCHAs and investigate their robustness and usability.

2.1 CAPTCHA: a reverse Turing test

The original concept of the CAPTCHA that distinguishes humans and computers apart was presented by Alan Turing as an AI problem in 1950, when he raised the famous question: "can the machine think?"[12]. His answer was through a test in which a human interrogator poses a series of questions to a human and a machine competitor. The interrogator distinguishes between the human and the computer through the answers. The CAPTCHA is a different kind of Turing test whereby the interrogator is a computer program that tells human and machine apart. This computer program asks one question with the aim of increasing security of the program rather than measuring the intelligence level[13–16]. This kind of test uses a simple encoding mechanism that protects the online forms and topics against a spying computer. This mechanism uses a number of character symbols to replace similar characters to confuse the spying computer (e.g. the symbol "€" replaces the letter "E"). This mechanism was the first proposed form of automated Turing test in what came to be known as "Leet speak" language[17]. Still, this language did not represent the required level of security, as some attacks could reverse the symbols to their original letters by use of a simple mechanism.

Theoretically, the first use of the Turing test for human verification was presented for discussion in an unpublished work by Naor[5]. The discussion showed that such a test could be used with a number of open challenges in the areas of computer vision and natural language as human verification steps. These can be identified as nine areas where the computer faces the challenge of an open problem to create such a test. These can be categorised to four domains: languishing understanding, object recognition, text recognition and speech recognition. Naor, however, did not propose any practical implementation of such verification nor provided any formal definition of it. Nevertheless,

his work still contributed to the first practical implementation of the automated Turing test, which developed after researchers followed this particular research.

The first implemented automated test based on the research of Naor was presented by the Alta-Vista search engine as a text recognition challenge. It consisted of a number of segmented and distorted characters displayed inside one standard image for the human to read and type in a specific box in order to solve the challenge. The test developers exploited the first versions of this text through basic OCR recognition attacks that succeeded in locating and recognising each of the segmented characters in them. These attacks helped the developers improve the security prospects of the challenge, which lead to the release of the first commercial text-based challenge in 2001[18]. This helped Alta-Vista protect their engine from abuses by 95%[19], which proves the importance of using the Turing test against bot attacks, including spam attacks.

At the same time as Alta-Vista was releasing their automated challenge, another company was researching the use of the Turing test against spam. This project was conducted by a CMU team led by Manual Blum with two students, Luis en Ahn, and John Langford. It aimed to generate the automated Turing test under the terms of the CAPTCHA through the following properties[13]:

- The challenges under the test need to be automatically generated.
- Each challenge needs to be quickly solvable by the human.
- The test needs to be readily acceptable to virtually all humans with only very few rejections.
- The challenges in the test should reject all machine users.
- The challenges need to have longevity to be able to resist attacks for many years.

These properties helped Luis von Ahn to coin the term CAPTCHA as "Completely Automated Public Turing Test to tell Computers and Humans Apart", which he described as "a cryptographic protocol whose underlining hardness assumption is based on an AI problem"[13]. He also discussed the attributes of AI problems presented in the CAPTCHA as follows[13]:

- The AI problem should be a hard-open problem.
- The attacker should be unable to write algorithms to solve the AI problems with a success rate higher than what is known in the state-of-art of the AI community.
- Such a problem can be generated automatically
- It should be solvable for the human, while it remains a problem for computers.

The role played by AI problems presented in the CAPTCHA challenges also caused Luis von Ahn to view this problem as a win-win situation: either the CAPTCHA succeeds as a security mechanism by resisting attack or it is broken in the process of solving a difficult AI problem[14].

The major AI problem issues discussed by Luis von Ahn, in addition to the stated properties of the CAPTCHA by his team, helped Yahoo Inc. to release the earliest challenges that came in the form of CAPTCHA schemes. Other companies then followed by releasing their own CAPTCHA schemes. These schemes are grouped into three main categories based on the AI problem:

The text-based CAPTCHA. this is the first and most used type of CAPTCHA that is based on an AI problem of character recognition. It is used by several major websites including Yahoo, Google, Facebook, Twitter and Alta-Vista. It contains a text challenge formed as an ordinary word or collection of typographic symbols (characters) which are letters, numbers and punctuation. The human solve those challenges by reading the text and typing it a specific box[20]. The earliest text-based CAPTCHA schemes were known as OCR challenges that displayed the text as a number of segmented characters which were protected against attacks through a series of character distortion mechanisms[21]. Other text-based CAPTCHA schemes are also known by their segmentation resistance principles. These principles include a range of mechanisms that display and connect the characters in the text through their own shapes or other noisy shapes around them. The most recent schemes that use segmentation-resistance mechanisms also combine them with other high distortion mechanisms. An example of this is a widely used text-based CAPTCHA known as ReCAPTCHA. These recent CAPTCHA schemes are currently the only ones regarded by our research as resistant to attacks.

The audio-based CAPTCHA. This type of CAPTCHA is based on an open AI problem in the area of speech or sound identification and recognition. The challenges of this AI problem include audio clips that presents digits spoken by different speakers. These audio clips also include distortion effects such as spaced intervals, distorted pronunciations and noisy sounds to make recognition by computers more difficult[22]. The user solves this type of challenge by identifying each of the spoken digits and distinguishing them from the other noises included in the clip. Many websites use this CAPTCHA as an alternative approach to the text-based CAPTCHA specifically to provide accessibility support for people with impairment. One example of this CAPTCHA is known as Echo, which is an early sound-based CAPTCHA scheme developed by Luis van Ahn[2]. It was followed by many other sound-based schemes such as Yahoo and ReCAPTCHA. This type of CAPTCHA still presents usability problems due to the noisy sounds and distorted pronunciations[23]. People find this type more difficult, more time consuming and less user friendly than the other types of CAPTCHA as many websites have suggested[24].

The objects-recognition CAPTCHA. This type of CAPTCHA involves different challenges posing an AI problem of object recognition in the CAPTCHA images or videos in a puzzle, which the human user should solve before entering the service. Prime examples of these schemes include the following:

- Bingo CAPTCHA, which is a challenge that displays two groups of images and asks the user to suggest the difference between them.
- Assira by Microsoft[25], which is a series of 12 images divided into two different sets for the human to differentiate. The first set shows a cat and the second a dog.
- Imagination CAPTCHA[26]is a challenge to find the centre of an image.
- Image Orientation CAPTCHA[27] is a challenge of a set of rotated images and the human is required to correct their orientation.
- 3D interactive CAPTCHA is a 3D image orientation CAPTCHA where a cube needs to be rotated in a sequence of correct faces to solve the challenge.
- Video CAPTCHA[28], which is an animated video presented with a specific question about its content for the human to answer.

Developers also proposed other schemes that use human ability to recognise human faces. Prime examples of these schemes include D. D'Souza et al.'s Avatar CAPTCHA[29] and G. Goswami et al.'s FaceDCAPTCHA[30].

2.2 Attacking the CAPTCHA schemes

The term "attacking CAPTCHA" could have different defined aspects based on the mechanism of the attack. The four aspects defined by Jeff Yan[15] are as follows:

1. **Breaking a CAPTCHA AI challenge.** This type of attack solves the CAPTCHA's AI problem underlined in its challenge and would help in making progress in the field of AI. This is the hardest and the most challenging form of attack. It is also the attack where the CAPTCHA achieves the original design goal of being a win-win situation.

2. **Breaking a CAPTCHA system.** This type of attack is aimed at the CAPTCHA implementation as an authentication protocol, and usually exploits the weaknesses found in that implementation inside the CAPTCHA system. For example, Yeen showed how to break the early CAPTCHA schemes by re-using a challenge image season's ID to bypass their implementations[31]. Sharma et al. also showed other weaknesses in the CAPTCHA systems found in its implementation in which the attack could even choose the value of CAPTCHA challenges and avoid the protection offered by the CAPTCHA system[32].

3. **Breaking a CAPTCHA as a security mechanism (Relay attacks).** Such attacks are more known as relay attacks, which is about bypassing the CAPTCHA challenges by passing them to someone else to solve instead of solving them automatically. This security mechanism is formed using two typical attacks. The first attack relays in cheap labour to solve the CAPTCHA including outsourced spam attacks in which the user is offered a competitive price to solve the CAPTCHA. These outsourced attacks are formed through sites such as DeCaptcher[33] and DeathByCAPTCHA[34] in addition to general services such as GetAFreeLancer[35]. Martiyama et al.[36] described typical attacks as large scale automation of site access that the CAPTCHA cannot prevent in cases where its costs and expenses exceed profit to the spammer. The second attack relies on social engineering tricks that leads the users to solve the CAPTCHA challenges for the

attackers without being aware of it. This is done by using attacks, including pharming attacks that infect the user's computer to make them solve the CAPTCHA for the attackers. Menuel et al.[37] presented an example of these attacks using an attack mechanism known as Man in the Middle (MITM). This shifts the CAPTCHA challenge to users trying to access web content or material. It then prevents those users from gaining access until the user solves the challenge. For example, according to the BBC[38], the spammers lunched an MITM attack in 2007 that tricks the users to solve CAPTCHA challenges that enabled the attackers to register for emails. These are both basic types of attack on the design of the CAPTCHAs that allow humans to bypass the challenges without the CAPTCHA being able to distinguish between legitimate users and the attackers.

The researchers proposed many algorithms to counter those attacks. The earliest proposal was Van Oorschot et al.'s ATT with Embedded Warning[39] in which the challenges are displayed with user-ID for each solver, so it can warn the targeted users through pharming attacks. The only drawback of this algorithm is that it needs human effort to prevent the attacks. It cannot also prevent outsourced attacks through paid solvers, which can be only countered by an assumption that the price of solving the challenges can be costly. This assumption was aided by Martiyama et al.[36] and Kanich et al.[40] who showed that the least retail price for solving 1000 challenge is $1, which can be increased in case the CAPTCHA is more complex. This assumption also inspired relay prevention proposals such as Zhu et al.'s "CAPTCHA as graphical password" (CaRP) in 2014[41]. Their relay prevention system depends in increasing the effort needed by human to solve the CAPTCHA challenges, so it can prevent them to solve the challenges at cheap cost. Other developers still able present better counter these attacks such as Truong et al.'s response time algorithm[42] that identifies the attacks based on the time taken by the relay attacks to send the challenge to the third party human solver and receive the return response for every character. Other algorithms could also use the mouse events as assists to improve the detection algorithm such as Manar Mohamed et al.'s Relay prevention algorithms used in in two Gaming CAPTCHAs[43,44].

4. **Breaking a CAPTCHA's underlying security mechanism.** This attack is aimed at the security vulnerability problems found in the CAPTCHA AI challenges. It

involves implementing an automatic computer program to solve the CAPTCHA challenges automatically by exploiting the vulnerability problems found in its design. Our attack in this thesis adopts this aspect.

2.3 The text-based CAPTCHAs

As mentioned earlier, the text-based CAPTCHA is a text challenge formed as an ordinary word or collection of typographic symbols that the user needs to type in a specific box to solve the challenge. Text challenges can be grouped into three categories. Two categories are known standard designs that display the text through static (non-animated) images including OCR-based CAPTCHAs and segmentation-resistant CAPTCHAs. The third category involves recent designs that display the text challenges as animated objects in what we call the animated text-based CAPTCHA. These three categories are explained in the following sections:

OCR-based text CAPTCHA. The earliest types of standard text-based CAPTCHAs are known as OCR based CAPTCHAs. These schemes showed a number of characters (typographic symbols) segmented from each other. The set of characters included in this type of text-based CAPTCHA could be either hand-written or randomly rendered characters using different fonts. The text that includes the sets of characters could also be distorted by two different techniques. The first technique includes affine transformations that transform characters using translated, rotated and scaled text techniques. It includes wrapping methods that deform the pixels of each character using small ripples, waves and elastic deformations. The distortion techniques could also include other noisy elements used behind the characters such as coloured backgrounds and cluttered background elements. Many notable OCR based CAPTCHA schemes were released to the public during the last decade including Yahoo CAPTCHA 1, Yahoo/EZ-Gimpy, Ticketmaster[45] and Captchaservice.org[46]. The most recent CAPTCHA that could be regarded as an OCR-based text challenge is a ReCAPTCHA “street view” CAPTCHA. The challenges of this CAPTCHA display street view house numbers (SVHN) within scene images taken from Google Maps[47]. This CAPTCHA is the first CAPTCHA that display the text from scene images instead of generating them through a CAPTCHA system. Figure 2.1 shows three examples of OCR based CAPTCHAs.

Figure 2.1. OCR based CAPTCHAs

Yet, the OCR-based CAPTCHAs proved to be vulnerable to computer vision-based attacks that could separate the segmented characters and recognise them through recognition engines. These attacks included machine learning mechanisms that use OCR engines to locate characters in CAPTCHA images and separate them with minimal pre-processing[48]. Chellapilla and Simond presented an example of these attacks that helped theme many OCR based schemes, including Ticketmaster, EZ-Gimpy and Yahoo v2[45]. That attack used a recognition engine, known as a convolutional natural network, to segment and recognise them with a success rate of 66.2% on MailBlock, 34.4% on EZ-Gimpy and 45.7% on Yahoo v2. Other attacks developed against these CAPTCHAs achieved much higher success rates after they used different methods adopted by the computer vision techniques. Moy et al.[49] presented an example of these techniques that could break the CAPTCHA schemes by estimating the object shape context in their challenges.

Table 2.1. The recognition rates of distorted characters in OCR-based CAPTCHAs[6]

Characters under typical distortions	Recognition rate
	~100%
	96+%
	100%
	98%
	100%
	95+%

Moy et al.[49] showed this technique by breaking Z-Gimpy with a success rate of 99% using a form of dictionary attack that compared the whole object shape with 561 synthesised templates of the dictionary. Based on Moy et al.'s discussion, all the templates included in the attack were gathered by estimating the text represented on EZ-Gimpy of 561 dictionary words. Another example was Ahmed Al-Ahmed's dictionary attack that broke Captchaservice.org with a success rate of over 90%[7]. Ian J. Goodfellow et al.[50] also broke an early version of Google's CAPTCHA with a success rate of over 90%. The high success rate of attacks was also achievable against highly distorted individual characters that the attack could recognise better than the human, including examples presented in table 2.1.

Segmentation resistance text-based CAPTCHA. This newer type of standard text-based CAPTCHA was designed as a result of both machine learning attacks and computer vision attacks against OCR based CAPTCHAs. These attacks were a major cause for the release of another type of standard text-based CAPTCHA schemes that used non-animated static images to display their challenges. This type of text-based CAPTCHA involved schemes known for their segmentation-resistance principles that increased the difficulty for computer programs to discover the place of the characters in a text string.

The segmentation resistance principle in those schemes included several types of mechanisms that could be divided into two sets. The first set of mechanisms connected characters together through their shape patterns using a mechanism known as "crowding characters together" (CCT)[51]. Figure 2.2 shows three examples of the segmentation resistance CAPTCHAs.

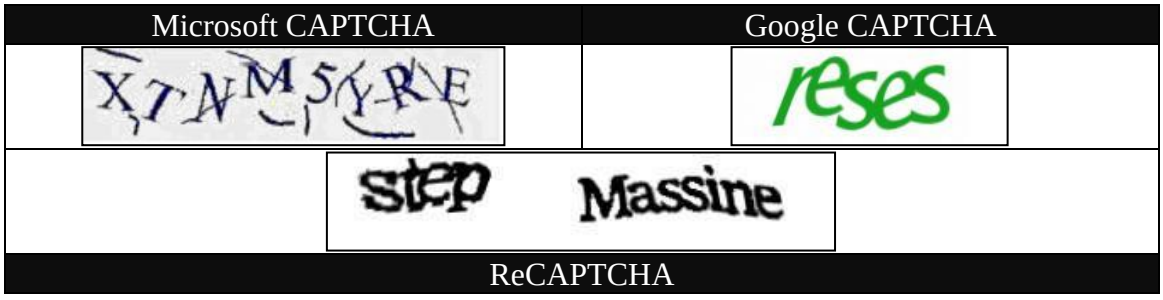


Figure 2.2. Segmentation resistance CAPTCHAs

Although segmentation resistance mechanisms proved effective against early attacks, some were still vulnerable to other computer vision-based attacks. We can present two types of attacks used against them:

Novel segmentation attacks. Those attacks include simple algorithms that aim to identify both the characters and the connected arcs between them through their shapes. Many attacks were developed using these algorithms including one by Jeff Yan and Ahmad El-Ahmad against the Microsoft CAPTCHA that used the connected arcs mechanisms[51]. This attack involved a few attacking algorithms that identified all the arcs used to hide the main characters through their shape. This broke the Microsoft CAPTCHA with an overall success rate of 90%. Another attack developed by Jeff Yan et al.[52] achieved a success rate of 62% in breaking Google CAPTCHA schemes that use CCT mechanisms. This attack included algorithms that could identify most characters through common patterns in their shapes. These patterns included dot components in "i" and "j", loop components in "o", "b", "0", "6" and "8" and cross shape in "t" and "f". Jeff Yan and Ahmad El-Ahmad targeted the Megaupload CAPTCHA in another attack that achieved a success rate of 78%[53].

Many other attacks were implemented using similar segmentation resistant CAPTCHAs. One of these was developed by Huang et al. in 2009[54] against a segmentation resistance version of the Yahoo CAPTCHA called Yahoo scheme 1. Their attack includes various of algorithms such as Vertical histogram, Sliding window, Middle-axis separation and character extraction. The combination of those algorithms succeeded in breaking the Yahoo CAPTCHA with a success rate of 79%. Gao et al. also attacked other versions of the Yahoo CAPTCHA. Their first attack targeted a hollow version of Yahoo CAPTCHA using a hallow filling algorithm that detects the hollow parts of the characters and the hollow chunks that connect them through their shapes. The series of steps included in this attack were the first to segment a hollow text with a success rate of 56%[55]. The second attack was against a later version that included projection approach that determines the shape of the head and tail of the text by filling its right-side and left-side using guide lines. The shape of other parts of the text were also determined through a similar approach fills that fills its upper-side and lower-side using the guide lines, which was aided with other attacking approaches such as the loop-detection, even-cut. Those combination succeeded in breaking the latest standard version of Yahoo CAPTCHA with a success rate of 9.2%[56]. Chad Houck[57] developed another attack that broke the first version of ReCAPTCHA with success rate of between 10% and 31%. This attack uses multiple deep segmentation algorithm that searches for valleys in the upper counter of the text and for peak in the lower counter, and

then draw a vertical line between each pair of valley and peak located in the same vertical column. Claudia Cruz-Perez et al.[58] developed another attack broke different version of ReCAPTCHA with a success rate of 40.4% using the different attacking methods that detect the characters in the text through the pattern of their own shape. In addition, Bursztein et al.[59] developed a method of attack against 15 text-based CAPTCHAs. This attack includes a combination of standard attacking methods that aim to remove the noisy background from standard text-based CAPTCHA schemes and segment them. Those methods are divided into sequence of steps, which are pre-processing, segmentation, post-segmentation, recognition and post-processing. The attack achieved a success rate of between 1% and 24% against Baidu, Skyrock, CNN and Digg, while it achieved 25% success against the Wikipedia CAPTCHA and 26% and 50% against eBay, Reddit, Slashdot and 50% or greater on Authorize, Blizzard, Captcha.net, Mega-upload and NIH. Colin Hong et al.[60] also developed a recent novel attack that broke the latest Microsoft CAPTCHA scheme with an overall success rate of 5.56% in 2015.

The overall results of the novel segmentation attacks helped Jeff Yan and Ahmad Al-Ahmad in 2010[7] to identify critical design flaws that cause CAPTCHAs to have exploitable invariants, as they could be strategically used in cryptanalysis to break the CAPTCHAs. Two categories of invariants have been identified. These are pixel level invariants that contain pixel count, colour pattern and the shape variants, and string level invariants that contain character set, text length and dictionary words strings. The developers countered those invariants with limited solutions included in recent text-based CAPTCHA. Some of the most recent examples are the Wikipedia CAPTCHA and ReCAPTCHA that included several sets of highly distorted characters to form the text challenge.

Deep-learning attacks. Deep-learning is a common name used for natural-network, which is a class of the machine learning[61] that enables computer programs to learn without being explicitly programmed. This class includes a set of methods implemented by using a representation-learning technique that feeds a learning machine with raw data to automatically discover the representations needed for detection or classification[62]. The representation-learning methods used in deep-learning include multiple levels of representation which are obtained by composing simple but non-linear modules. Each of these modules transforms the representation at one level into a representation at a higher, slightly more abstract level[62]. The most known neural-networks used in deep learning are

convolutional natural networks, which are sets of neurons with tied parameters[63,64]. These implement offline transformation as a discrete convolution and can be applied to filter several layers with each layer to the vector input followed by an element-wise non-linearity.

Many developers have used deep learning to implement the next generation of segmentation attacks that replace novel segmentation algorithms. These attacks feed a text image to a neural-network machine that segments the text by dividing its image into a series of connected layers. They then analyse each of them to identify the layers that include portions of every typographic symbol within the image of the text, before using them to recognise the symbol. One of the attacks was implemented by Goodfellow et al.[50] to break "Google Street View" challenges by recognising street view house numbers (SVHN) displayed within their scene test images. This attack achieved a success rate of 99.8% in recognising the street numbers in those challenges. Fabian Stark et al.[65] implemented another attack in 2015 to target self-generated CAPTCHA challenges that are like the ones used by ReCAPTCHA. This attack included a proposed active learning technique to reduce the required training data in which it achieved an accuracy rate of 85% in recognising the samples of this CAPTCHA, after it trained the data for epoch 40.

The developers also implemented other deep-learning attacks on numerous standard datasets between 2011 and 2014. These datasets included standard text challenges and natural scene test images downloaded from several sources including Google's image search engine. The first attacks against these databases were based on a sequential character classification that only finds the characters through sliding windows. Wang et al.[66] presented the first attack that uses this technique, which recorded an average success rate of between 61.3% on the samples of databases. Following this attack, Wang et al.[67] presented improved algorithms of his attack that achieved success rates of 81.3% against the databases. However, other attackers implemented different attacking algorithms that used the notion of holistic word recognition. The first two attacks that used this method were implemented by Novikova et al.[68] and Mishra et al.[69] in 2012. These attacks achieved average accuracy rates of between 69.3% and 74.3% against the samples in the databases. Other developers also improved attacking algorithms such as Rodriguez-Serrano et al.[70] who presented his attack in 2013 that used aggregated Fisher Vectors. These algorithms showed a huge improvement in the deep-learning mechanisms in which the average accuracy rate of the overall attacks released in 2013 were 84.5% recorded[71–73]. The

attackers also made other improvements to the algorithm including Almazan et al.[74] who formulated the joint embedding spaces. These attacks raised the average accuracy rate of the overall attacks in 2014 to 88.4%[74–78].

The developers also used deep-learning as part of advanced attacks that included novel segmentation methods. Bursztein et al.[79] presented one of these attacks in 2014 that combined deep-learning and novel segmentation algorithms. This attack achieved a success rate of 22% against a 2013 version of ReCAPTCHA, 55% against Baidu CAPTCHA and 51% against the eBay CAPTCHA. Another attack that used a similar technique was implemented in 2015 by Yan et al.[80]. This attack achieved a success rate of 77.2% against the ReCAPTCHA, 23.8% against Wikipedia CAPTCHA, 16.2% against Microsoft CAPTCHA, 25.8% against Amazon CAPTCHA, 44.2% against Baidu CAPTCHA, 5% against Yahoo CAPTCHA and 58.8% against eBay CAPTCHA. These attacks helped Bursztein et al.[79] to declare thar the standard text-based CAPTCHA to be completely solved and encouraged many websites to replace them with other AI challenges such as objects-recognition and motion-based CAPTCHAs.

The animated (motion-based) text CAPTCHA. This new type of text-based CAPTCHA, also known as motion-based CAPTCHA, involves an alternative approach to add motion to the CAPTCHA to provide a better space for security and usability. Animated (motion-based) text CAPTCHAs have become the focus of developers since solutions available in designing standard text-based CAPTCHAs were limited. E. Athanasopoulos et al.[81] were the first to propose the idea of adding motion to the CAPTCHA as a way of improving the robustness and usability of the object recognition CAPTCHA. After that, developers included it in the design of text-based CAPTCHA schemes to add a timing security mechanism that separates text information through multiple animated frames. The human can solve these CAPTCHAs by studying and recognising the animated challenge but it is difficult to achieve in automated attacks. Cui et al.[82] discussed the general steps to designing these animated schemes, which are:

Step 1: Determine the elements of the original contents set. These contents are an English alphabet set or an Arabic numeral set that forms the text challenge.

Step 2: Generate a verification code randomly and a set shown in position in animation. This verification code must include three or more elements from the original set that is chosen randomly.

Step 3: Draw frames of animation. Animation is produced by the technique used in drawing the frames. This technique includes choosing the number of moving objects in each frame such as noisy objects. It also includes choosing the initial position, colour, shape, size, and the moving orbit of each object in addition to animated attributes of those objects. All choices then need to be used to draw the frame that include the moving objects.

Step 4: Repeat each frame's drawing and save the animation. In this step, the developer chooses the total number of frames included in each challenge, and then repeats step 3 for each frame.

In later stages, we are going to include details about the design in addition to the robustness and usability issues currently identified.

2.4 The design of the animated (motion-based) text CAPTCHAs

2.4.1 *The early animated text-based CAPTCHA*

The early-animated text-based CAPTCHA schemes were designed to animate characters separately in respective vertical columns within the animated image. These schemes animate characters using a number of effects. Some of these effects include a time delay feature that displayed a character completely or partially for a short length of time before fading out or disappearing. The second category involves a number of changes that scale, rotate and move the character vertically or in different directions. These CAPTCHA schemes also include a number of animated noises in the background and foreground.

Many of the early CAPTCHA schemes animated their characters by using one specified effect such as the Dracon CAPTCHA that used the fade effect, which causes the characters to fade and blur for a certain length of time, and the AmourAngels that used the vertical move effect, which causes the characters to move up and down. However, some of the animated CAPTCHA schemes may include more than one effect. The earliest example of these schemes is the CAPTCHANIM that incorporates a variety of designs that include a few animated characters by using the scaling feature. These characters may stay in their

position or move in and out of the CAPTCHA frames from left to right or right to left. In addition, they can be distorted through an additional animated effect that is known as horizontal deformation methodology, which shifts the shapes of those characters horizontally in different directions. Figure 2.3 shows examples of early-animated text-based CAPTCHA schemes.

Figure 2.3. The early animated text-based CAPTCHA schemes[83–85]

The most complex scheme of the early motion-based CAPTCHAs is the KillBot Professional CAPTCHA. This is a commercial CAPTCHA used by the United State Federal Government as the developers state on their website[86]. It entails a variety of different CAPTCHA designs, each of which contains a specific number of random characters that change themselves in specific vertical columns. Each character changes itself using an advanced approach that combines blurring and fading effects with one or more of the major animated character changes that include scaling, rotating, or moving vertically up or down. This CAPTCHA scheme also uses different types of animated noises like raindrops, random characters that move horizontally or vertically, falling bars, and moving lines. Figure 2.4 shows an example of the KillBot CAPTCHA animated challenge.

Figure 2.4. The major design of the KillBot professional CAPTCHA[86]

2.4.2 *The moving object CAPTCHAs*

Other types of the animated text-based CAPTCHAs were developed using segmentation resistance mechanisms that normally overlap a few animated characters inside the CAPTCHA frames. The developers of NuCAPTCHA Inc.[87] created an animated text-based CAPTCHA that overlaps characters and moves them at the same time to make the

CAPTCHA resistant to segmentation attacks. This CAPTCHA moves the characters by rotating each of them individually while they are moving together from left to right. This CAPTCHA also contains additional characters that move horizontally without rotation alongside the rotated CAPTCHA characters in addition to an animated background. These two features play a major role in increasing the difficulty for the automated to track and capture the CAPTCHA characters. Figure 2.5 shows an example of NuCAPTCHA.

Figure 2.5. NuCAPTCHA

Developers designed other animated text CAPTCHAs that used complex mechanisms that show the user two groups of animated characters, which are the main characters and noisy characters. These two groups use the same font size to confuse the computer program, while they use different animation behaviours to help the human to differentiate between them. The earliest animated text-based CAPTCHA that used this kind of mechanism depended only on appearance and length of time to help the human distinguish between the main characters and the noises. For example, Atlantis-caps.com[88] released an animated CAPTCHA scheme that involved two sets of characters displayed in different colours that keep changing through the animation frames. The first set of characters represents the main CAPTCHA characters and these appear consistently in front of the animated frames. The second set is a group of characters that appear briefly in the background of the CAPTCHA frames to create noises behind the main characters. Figure 2.6 shows an example of this CAPTCHA scheme.

Figure 2.6. Atlantis-Caps

The first animated CAPTCHA released using this concept was the AniCAP, which is an experimental animated text-based CAPTCHA developed by Y. Chow and W. Susilo[89]. The main characters in this CAPTCHA appear as 3D characters coloured grey in front of the 3D CAPTCHA image. In addition, another set of 3D connected characters also coloured

grey appear behind the main CAPTCHA characters to create noises around them. Both sets of 3D characters are rotated in 3D demolition, so the human can locate the main characters by looking at their movement in parallel with the rotation direction inside the animated image. Yahoo Inc. also released an animated text-based CAPTCHA that includes the same mechanism that uses animated characters to form the main text challenge and the noises. However, the animated behaviour of each group in this animated CAPTCHA is different from the AniCAP in which the noisy characters move in horizontal directions, unlike the characters of the main text. Figure 2.7 shows an example of the AniCAP and Yahoo animated CAPTCHA.

(A)

(B)

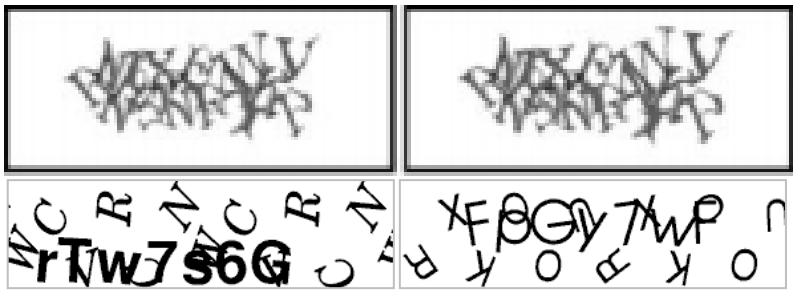


Figure 2.7. AniCAP (A) and Yahoo CAPTCHA (B)

2.4.3 *The "emerging image" in animated text-based CAPTCHAs*

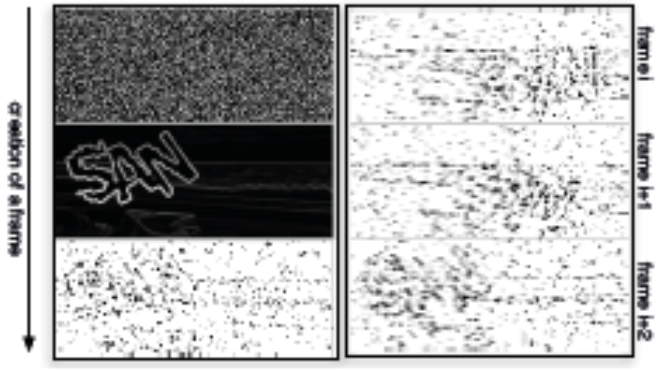
The "emerging image" is a recent robustness mechanism adopted by the animated CAPTCHA challenges. The first proposal of this concept was made by Mitra et al.[11]. They described it as "the unique human ability to aggregate information from seemingly meaningless pieces, and to perceive a whole that is meaningful". The adaptation of this mechanism was through a 3D video CAPTCHA by Mitra et al.[11] that was released to defend tracking attacks that use "frame-to-frame" analyses.

The first adaptation of the "emerging image" mechanism in an animated text-based CAPTCHA was by Y. Xu et al.[10] who used it in a new version of NuCAPTCHA. This version helped to defend tracking attacks that broke the original design of animated text-based CAPTCHAs with a high success rate. Y. Xu et al.[10] presented their implementation of the "emerging image" version of the NuCAPTCHA through methods that generated two sets of animated frames. The first set included grayscale frames that displayed the edges of animated text of the NuCAPTCHA alongside the edges of the background. The second set were from a set of noisy images that were generated using Gaussian distribution. This

method combines the two sets of grayscale frames through an equation that calculates the grayscale value of each pixel in each frame as follows:

$$P(x, y) = S(x, y)e^{(F(x,y)/C)}$$

In which F is a frame from the first set and S is a frame from the second set and C is constant number. The method makes each pixel point $P(x, y)$ with a value greater than constant number T white to create an animated final set of frames that display the edges of the animated characters in blurred images alongside a noisy background. Figure 2.8 (A) shows an example of the "emerging image" mechanism and figure 2.8 (B) shows an example of the resultant image.



(A) **(B)**

Figure 2.8. The design of the "emerging image" in NuCAPTCHA

The concept of "emerging image" also inspired other animated text-based CAPTCHA schemes. An example of these schemes was created by Ivo Kund from the University of Manchester that contained a number of small lines and background clutters[90]. This CAPTCHA aims to use the small lines to create two types of moving particles. The first type contains groups of moving particles that represent the CAPTCHA characters, while the second type contains individual moving particles that aim to add noises to the background. The groups of moving particles that represent characters inside the CAPTCHA image are divided into multiple frames to resist frame-to-frame tracking attacks. These particles also move in different directions to make it difficult for attacking bots to use the PDM tracking method to detect those characters. Figure 2.9 shows an example of the Kund's CAPTCHA animated challenge.

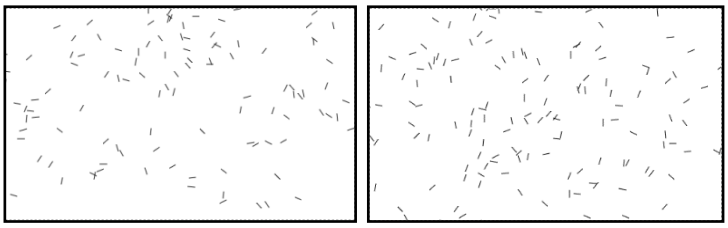


Figure 2.9. The Kund's CAPTCHA

The Gao et al.[44] also developed another "emerging image" CAPTCHA in 2015. This CAPTCHA is designed only for gaming engines to resist both automated and human-solver relay attacks that target video games, as it keeps generating infinite numbers of frames at a rate of 40fps. Each animated challenge in this scheme displays five segmented characters that move randomly within the middle and right sides of a 3D noisy background. Each of those characters are distorted using animated 3D object rotation, incomplete object contour and tiling background, so they can reduce the information exposure to the computer program. Each challenge also distorts three other characters by using the same features mentioned before in the left side of the animated challenge. Each of the three character resembles one of the other five animated characters that move randomly within the 3D noisy background. The normal human could solve this challenge simply by dragging each of the three characters with its resembled one in the middle/right side of the animated image. However, attackers would need a more difficult challenge to answer it by using a computer program due to the infinite and high rate of frame generation in its samples. The infinite and high frame rate generation also resists the relay attack, as the attacker needs to keep sending streaming images of the animated challenge to the human solver at a speed of 40fps. He would also require to asynchronies the solver's dragging and dropping answer from his computer at very high speed. Those attacking requirements would make the CAPTCHA challenges impossible to break with the relay attack, as current streaming technologies are not enough to reach any of them.

2.5 The security of the animated text-based CAPTCHAS

Although the concept of the animated text-based CAPTCHA is still new to the research community, many animated schemes were still broken by two algorithms. These algorithms include character capture attacks that targeted the early-animated text-based CAPTCHA schemes and frame-to-frame tracking attacks that targeted the NuCAPTCHA scheme.

2.5.1 *The character capture attacks: the security of the early animated CAPTCHAs*

The earliest attacks against the animated text-based CAPTCHA schemes used technology to catch their animated characters in good images. Vu Duc Nguyen et al.[91] presented an example of this attack that broke many early animated schemes. It included a strategy to gather the relevant information from the animated frames and extract them into a static image that displays the whole text. This strategy included a major approach that extracted the characters that pause, move and scale in respective columns and which were run through two methods.

The first method is the pixel delay map (PDM), which was implemented against an animated CAPTCHA that contained characters appearing or pausing in fixed locations for a long period. This method calculates the length of time that foreground pixels take to appear in each coordinated point in the animated image using an algorithm that count the number of captured frames that contain the foreground pixels in that coordination. It then uses the results of the calculation to create a mapping image known as pixel delay map (PDM) that shows the foreground pixels that appear in coordinated point for a certain length of time. We could describe the pixels that appear for the highest number of times as pixels of the animated characters that appear and pause in fixed locations for a long length of time.

Figure 2.10 shows an example of this method.

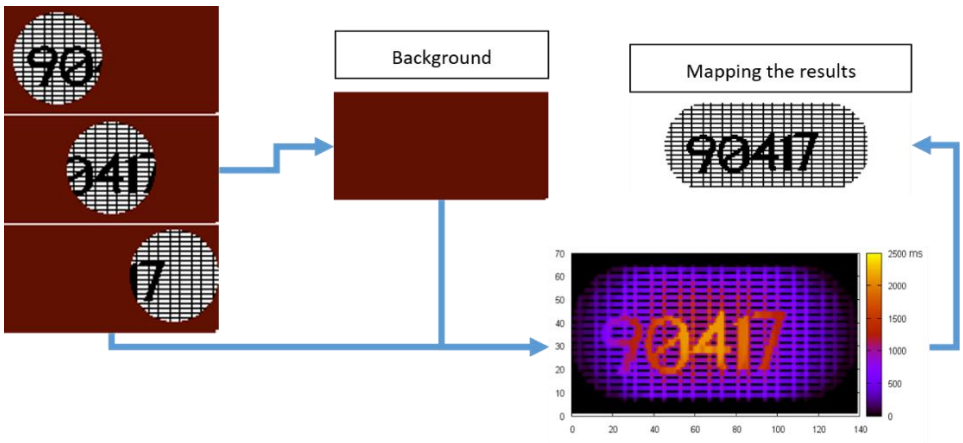


Figure 2.10. The Pixel Delay Map (PDM)[91]

The second method is Catching Lines (CL) which is a method used against the characters that move or scale in vertical columns within the image. This method uses the PDM map to calculate the maximum pixel for each character and determines the vertical area where

pixels of each animated character are located in the animated frames. Then, it uses the results to segment the characters vertically by adding two vertical lines on the side of its vertical column in the PDM. After that, it analyses each of the segmented animated characters to catch it when it reaches a particular height. This height is normally in the position where the character is fully displayed and easy to obtain and after it reaches a certain height as stated by the developers in their paper. Figure 2.11 shows an example of this method.

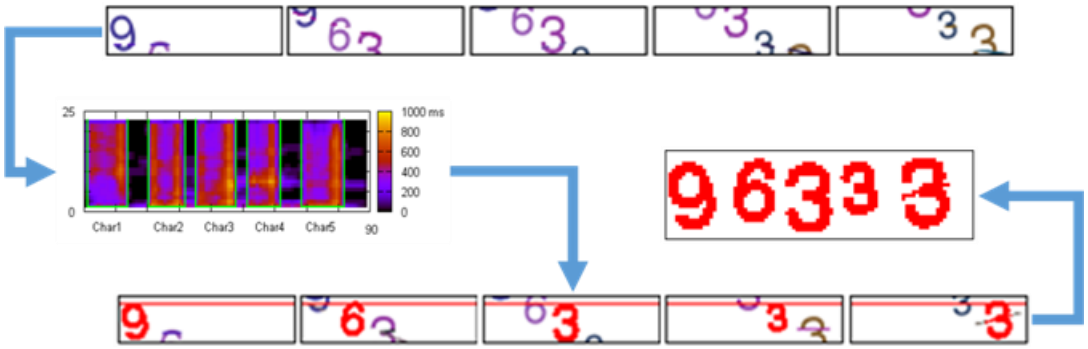


Figure 2.11. Catching Lines (CL)[91]

The method identifies the height (horizontal location over x-axis) that the head of the character needs to reach in one fixed location in case the character moves vertically. This height can also be determined for each character in a position where it reaches its maximum height, which can be identified as the height where the pixels of the character give a value bigger than zero in the PDM, in case the target is either rotated or scaled characters. However, one method determines the height for scaled characters that may go outside the boundary as the highest position of the area where the PDM of that character can give the highest value. Vu Duc Nguyen et al.[92] showed other methods used only against specific animated CAPTCHAs during the character extraction process. One of these methods is Colour Selection (CS) used against the animated CAPTCHA that includes characters that use different colours and move at random speeds and directions. This method is used to separate characters based on their colours and selects each from a frame where the maximum number of pixels are coloured using a colour of the one displayed. A second method is Frame Selection (FS) that targets the animated CAPTCHA that includes characters that use the same colour and move at random speeds and directions. This method depends on selecting the frames that display the highest number of pixels that do not belong to the background. After that, it chooses the frame where all characters are separated into

distinct regions, while none of them is in contact with image boundaries. In addition, a third method called Roller Selection (RS) targets the animated CAPTCHA that includes rotated characters. This method segments the characters using a flood-fill. After that, it extracts them using the CL method that catches each of them when its highest pixel reaches its maximum height.

The attack also includes extra methods to filter out the noises during the pre-processing in the attack. These methods distinguish the noises from the main characters through their movements, colours, brightness and the length of time they take to remain displayed through the CAPTCHA frames. The attack may also include a step to correct the order of the characters that move horizontally in the CAPTCHA frames through a method that swaps the two contents on the sides of the vertical line. This method is used in case the targeted challenge is a second variation of CAPTCHANIM.

The CAPTCHA attack achieved an average 85.5% success rate against the CAPTCHA schemes that include a time delay feature, after it used the PDM method against them. In addition, it achieved a 47% success rate against AmourAngels CAPTCHAs using the CL method. However, the attack achieved only a 21% success rate against CAPTCHANIM using the CL method besides a character correction method that swaps the two contents on the sides of the vertical line in the second variation of CAPTCHANIM. It also achieved an 18% success rate against the KillBot professional CAPTCHA after it used both attacking methods against it.

Thus, the attack proves the ability of the computer program to read characters that move, scale and fade in and out in the correct order and extracts them in a good image. It also proves the weakness of the noises used in the animated CAPTCHAs as they have different behaviours than the main characters. Yet, the attack showed limitations against the complex distortion methods including the rotated character methods such as the CL method, which failed to catch them in the correct rotation form. This may cause problems in recognition especially when the rotated character is one of the confusing characters like “n”, “u”, “p”, “q”, “b”, “d”, “N” and “Z” that appear as rotated by more than 90 degrees. In addition, the character correction in the CAPTCHANIM could catch the character when it moves partially out of the frame in the second variation. This problem is caused by the inability of the attack to detect these frames, as they appear randomly within the animated image. These

issues caused increased difficulty for the automated attack to achieve a high success rate against animated CAPTCHAs such as KillBot and CAPTCHANIM.

2.5.2 *The frame-to-frame tracking attacks: The security of NuCAPTCHA and "emerging image".*

The frame-to-frame attacks are the first attacks against the animated CAPTCHAs that use tracking attack analyses against the animated text-based CAPTCHA schemes. These attacks included two against the NuCAPTCHA that used approaches to crowd three rotated characters together and make them move horizontally alongside many ad characters that did not rotate. The developers of this animated text-based CAPTCHA describe it as one of the most secure CAPTCHA scheme because of two features[93]. The first feature is character overlapping that increases its resistance against OCR-attacks. The second feature is random security letters that make it difficult for the attacks to decode the letters in the CAPTCHA after decoding one of them[93]. However, the attackers still proved the opposite through two attacks which were composed of two major steps used to identify and segment rotated characters. We are going to discuss each of these attacks individually in the next sections.

The Elie Bursztein attack. Elie Bursztein[94] created one of the attacks that broke the NuCAPTCHA with a high success rate of above 90%. This attack catches the CAPTCHA video and decodes it to break it into frames. Then, it analyses all the moving objects, which include all the animated characters in all frames to find the most rotated CAPTCHA characters. After that, it identifies all the frames that contain the CAPTCHA characters and uses them to create many images of these characters. It then runs a segmentation attack against each of the images, before it selects the best answer using a voting system.

The first step used to find the characters in the Bursztein attack relies on the image and motion tracking methods. After that, it clears the background around the moving objects that includes the rotated characters in addition to additional ad characters. Then, the attack identifies the rotated characters through a method that adds interest points to the edges of characters using the SIFT (scale-invariant features[95] algorithm). Those interest points help to track the movement of the character, so the attack can locate the main characters through their rotations. This method also adds bounding boxes around the edges of animated characters to locate the overlapped characters of the main text. The method analyses the bounding boxes to find the ones that have a width/ height ratio of greater than one. It then

searches for the bounding box that includes more corners and edges than the other bounding boxes through a density metric algorithm. This algorithm computed in each bounding box is as follows:

$$\sum_{k=0}^{n} \binom{n}{k} \frac{1}{\text{Distance}(P_k,\ \text{Box\_Centre})}$$

In which the distance is Euclidian Distance and P_k is each interesting point. The result determines the most interesting boundary box that contains rotated CAPTCHA characters and achieves the highest score computed through the density metric algorithm.

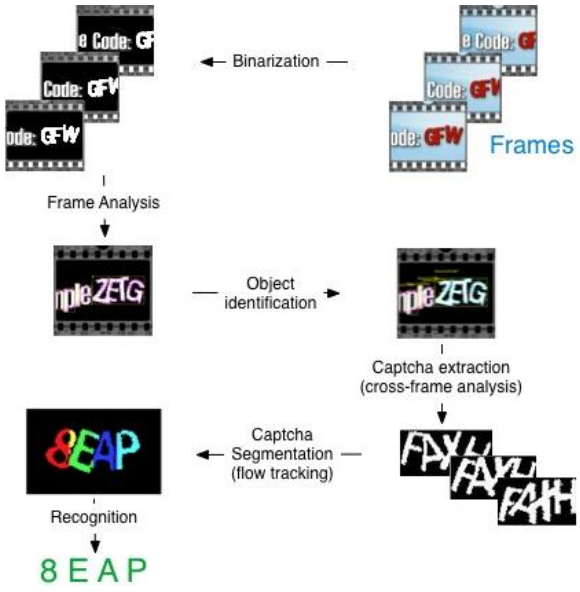


Figure 2.12. The Bursztein attack against NuCAPTCHA[94]

The rotated characters inside the selected object are segmented using an algorithm that tracks the interest points to link the points that rotate together around one centre. Each group of the linked points forms a body of one rotated character. The segmentation algorithm used in the attack analyses each group of linked points to find the beginning and end of each rotated character, so the algorithm segments this character in each frame. Finally, it analyses the segmentation result in each frame and selects the best answer. Figure 2.11 shows an example of the rotated characters identification process and the segmentation method used in the Bursztein attack.

Y. Xu et al. attack. In parallel to this attack, Y. Xu et al.[10] have developed an attack that uses different features to break the CAPTCHA. This attack identifies moving characters

through a method that detects the silent features, which include the corners of the characters using the Harris corner detector[96]. The attack then executes a "tracking" procedure that uses the KLT-tracking method[97] to find the characters that move together in the image. This method identifies and connects the silent features in each frame to trajectories and discards the salient points with short trajectories that belong to the background. After that, the attack searches for trajectories with the largest deviation from the more common motion trajectory and identifies it as the trajectories of the main text. Then, the attack executes a "foreground extraction" procedure that uses the Gaussian mixture mode[98] to identify the foreground colour of the main characters and extract them. The attack then executes a "segmentation" procedure that uses k-means clustering[99] to create a bounding box based on the centre of the rotated object and its orientation in each frame. It then segments and extracts each character marked within one boundary box. After that, it sends the character to a natural network engine to process a "classifier" that analyses the segmented images of each character from each frame to find the best answer. The attack could also include an optional "feedback" procedure that uses the classifier to reduce the distraction caused by overlapping the character. This attack has achieved a success rate of 75% against the NuCAPTCHA scheme without activating the feedback approach which took an average of 30 seconds to break each CAPTCHA challenge. It also achieved a success rate of 77% against the NuCAPTCHA scheme with the feedback approach, taking an average of 250 seconds to break each CAPTCHA challenge. Figure 2.12 shows an example of procedures included in the Y. Xu et al. attack against NuCAPTCHA

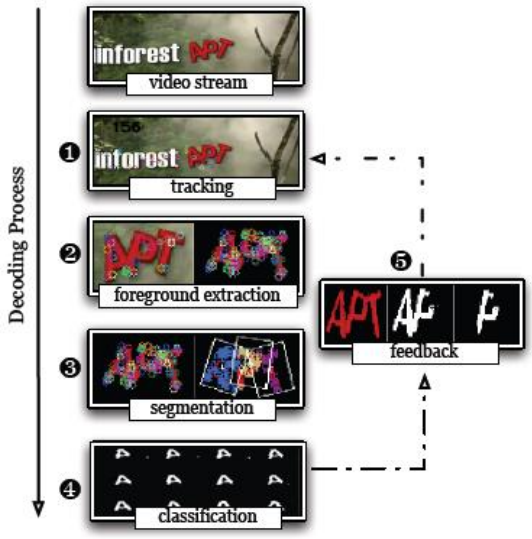


Figure 2.13. Y. Xu et al. attack against NuCAPTCHA[10]

Y. Xu et al.[10] used their attack against other variants of the NuCAPTCHA that offer different defences. The first variation of the NuCAPTCHA is the extended variation that included an extended number of rotated characters. The second variation included in the study is the overlapping variation, which is the same as the standard version of the NuCAPTCHA except that the characters are more overlapped. The third variation of the NuCAPTCHA is the semi-transparent that is also identical to the standard version of the NuCAPTCHA except the characters in it are semi-transparent. The results of the study showed that the attack could still break the three variations of the NuCAPTCHA. Based on these results, the attack achieved a success rate of 5% against the first variation of the NuCAPTCHA that included 23 rotated characters in each challenge. It also achieved the same score against the other two variations of the CAPTCHA scheme.

The developers of these attacks used results to show the vulnerability of segmentation-resistance mechanisms on animated text-based CAPTCHAs. Based on their arguments, the attacks can exploit those mechanisms by tracking the movement of each character in the animated image. This issue can also make the animated text-based CAPTCHA less safe than the standard text-based CAPTCHA.

However, Y. Xu et al.[10] showed that these issues can be still encountered by using an "emerging image" mechanism included in a version of NuCAPTCHA . Based on their research, this version of NuCAPTCHA could resist all automated attacks developed against other versions of this scheme. Y. Xu et al. used this result to prove the resistance of the "emerging image" against tracking attacks through its feature that separates the pieces of the text and background through the animated frames. Another examination by Kund[100] showed similar results on an inspired scheme of the "emerging image" that uses small lines instead of noises. This research showed how the scheme can raise the difficulty threshold in breaking the scheme by using the same shaped noises to display the main text and the background.

2.6 The usability studies of CAPTCHAs

Many usability studies have been conducted on CAPTCHA schemes, including some research on animated text-based CAPTCHAs. In this section, we discuss those usability studies, especially those related to the animated-text based CAPTCHA. These studies are mainly based on the five Jakob Nielsen[101] usability principles. These principles are:

- Learnability: how easy is it for users to solve the CAPTCHA the first time they encounter it.

- Efficiency: once users have learned to solve the CAPTCHA, how quickly can they solve each challenge.

- Memorability: when users return to the CAPTCHA scheme after a period of not using it, how easily they can re-establish proficiency.

- Errors: how many errors users make, how severe these errors are, and how easily to recover from them.

- Satisfaction: how pleasant for the human is it to use the design.

2.6.1 *The text distortion related issues*

Many studies have focused on the usability of distortion methods. Microsoft [102]showed one of these studies that focused on the level of standard distortion methods such as translation, rotation, scaling and wrap. This study helped in designing the Microsoft CAPTCHA scheme that was deployed for a year on Microsoft websites before it was broken in 2007[51]. Another study by Ahmad El-Ahmad[103] showed how distortion issues related the confusing characters in segmentation-resistance CAPTCHAs. These issues included one in which connected characters (letters and digits) could resemble other characters when connected to each other. Example of this are “vv” that can resemble “w”; “cl” that can resemble “d”; “nn”, and “rn” that can resemble “m”. Other issues related to the confusing characters is the similarity between some of them. For example, you cannot tell 1 from l, z/Z from 2 and S/s from 5. These studies also focused on the effect of distortion issues with foreign people whose mother tongue does not use the Latin alphabet. One of those studies was conducted by Chew and Baird[104] using students whose mother tongue uses the Latin alphabet and others whose mother tongue does not. The study showed that the students whose mother tongue does not use the Latin alphabet performed much worse than those whose first language is based on the Latin alphabet.

The usability studies also focused on the animated distortion methods such as the animated character changes that are included in several animated character schemes. Nguyen et al.[91]

discussed the usability issues of three methods currently used in the animated text-based CAPTCHAs, which are as follows:

- **Time delay.** The animated text-based CAPTCHAs that use this type of distortion aim to display each character for a specific duration before it disappears or fades out. The duration of time chosen to display the character should be enough for the user to read the challenge and determine the position of each character inside it. Instead, it causes usability problems especially for people who read slowly, as those people need to wait every time they miss a character until it appears again.

- **Character movement and scale.** Few text-based CAPTCHAs use animated changes that scale the characters or move them in different directions to prevent their full appearance in each frame. We can regard these animated changes as usable, as humans can predict and solve them by following the animated characters until they appear clearly in the CAPTCHA frames.

- **Character rotation.** The rotation is another type of animated change applied to the CAPTCHA characters. This method is used to represent the characters in several orientations that can be greater than 90 degrees. The human can guess most rotated characters by their appearance. However, the human may face usability problems if the rotated character is one of the confusing characters such as “n”, “u”, “p”, “q”, “b”, “d”, “N” and “Z” - especially when the orientation of those characters reaches nearly 180 degrees.

Other methods used to animate the text include the horizontal deformation method, which shifts the shapes of the characters horizontally in different directions. This method does not present a problem for the user who can easily identify the distorted character by using the methods in any form through the animated frames.

2.6.2 *The text layout related issues*

Many issues related to the text layout were mentioned by the researchers. These issues include those found in both standard and animated schemes and are related to character sets and string lengths mentioned by Ahmad El Ahmad[103] in which an increase in the character set and string length can decrease the usability of the CAPTCHA. Other studies also compared the usability of both random string and dictionary words. These studies showed

that dictionary words are always more solvable and understandable to the human than the random string[105,106].

The studies also include ones focused on the effectiveness of the animated feature in animated text-based CAPTCHAs. These features include making animated characters stay either in fixed locations or move and scale in respective columns. These features can play a major role in facilitating the usability of the CAPTCHAs, as they can make it easier for the human to identify the correct order of the main characters[91]. They could also prevent the human from missing any animated character that appears randomly, as the human could determine the number of columns that contain each character in the animated challenges.

Other animated mechanisms included in these CAPTCHAs are animated segmentation-resistance mechanisms that move the animated characters horizontally and overlap them together. These schemes were considered as usable, because humans can locate each character by looking at its individual animation as Giasson et al. stated in their paper[93]. As such, the user can identify the main characters in the NuCAPTCHA challenges by looking at the rotated characters inside the animated challenges. Then, the user can identify each character by tracking its rotation to find each of its corners.

Even so, these features can have a negative impact on the usability of the animated text-based CAPTCHAs, based on the number of characters and their level of overlapping inside the CAPTCHA challenges. Y. Xu et al.[10] showed this negative impact in their study by using different variations of the NuCAPTCHA. The variations included the standard variation of the NuCAPTCHA, which is the original version of the NuCAPTCHA. They also included the other three variations mentioned before which are the extended variation, the overlapping variation, and the semi-transparent variation. This study involved many participants presented with ten challenges from each variation of the NuCAPTCHA. The results of the study showed that the users achieved a significantly lower score in the extended, overlapped and semi-transparent variations of the NuCAPTCHA than the standard variation. These results showed also that users spent much longer in resolving the three other variations of the NuCAPTCHA than the standard one.

As a result, the features used to move the animated characters horizontally and overlap them together could be usable in only one case, which is when the characters are not closely overlapped. Otherwise, it can cause a considerable problem to the usability of the animated

text-based CAPTCHAs that could affect the accuracy of the human's solving ability and the time needed to solve each challenge. This problem has had a major impact on the user's understanding and satisfaction as Xu et al.[10] proved through their study which showed participants in usability examination reacting negatively towards the extended, overlapped and semi-transparent variations of the NuCAPTCHA, and preferring the standard variation.

2.6.3 *The presentation related issues*

The way that the animated CAPTCHAs are presented can play a major role in identifying the usability of the animated text-based CAPTCHA. This can include the use of noises and colours and the way the characters are presented alongside those noises. It can also include integration problems with web pages and browsers and the loading time.

The use of noises. The noises are extensively used in the current animated text-based CAPTCHAs to hide the characters from the automated attacks. As several types of noises have colours and brightness values that are different from the main characters the human focuses only on the main characters. Few types of noises are being different from the main characters in their movement and the time needed to display in the animated challenges. The usability of these noises depends on the human's ability to distinguish between the main text through their animation. This main case is more present in schemes that display the noises and main text in a similar shape, such as the schemes that use the "emerging image" mechanism.

One of the main CAPTCHA schemes that revealed a problem with noises is Kund's CAPTCHA[90] which displays the main characters and noises using clutters that move in different directions. This CAPTCHA suffers from usability problems that make the CAPTCHA challenges too difficult for the users to solve. One of these is the similarity between the noise small lines and the characters' small lines which makes it too difficult for the user to differentiate between the two groups of lines. In addition, the characters' small lines make the character unreadable in some cases, especially where the lines are very close to each other. These issues caused participants in a usability examination of this scheme to achieve an accuracy rate of 61% within an average speed of 11 seconds. Y. Xu et al.[10] countered this problem in his "emerging image" version of the NuCAPTCHA by making the animated text highly visible and distinguishable from the other noises around

it. However, the usability examination of this version still showed users giving a mixed rating with a negative rating in error-prone and pleasant-to-use questions.

The use of colour. Colours are extensively used in many different text-based CAPTCHAs because of their usability advantages that include:

- The number of different configurations provided by colours that could make the CAPTCHA suitable to many different user preferences[107].
- The additional computability with the design of a colour theme of websites makes them less intrusive[21].
- Their appealing and strong attention mechanism could make the CAPTCHA more interesting[103].

However, the use of colour can also cause usability problems that were identified in the early text-based CAPTCHAs where the background was filled with different colours. The earliest usability issues were present in standard text-based CAPTCHA schemes that included many colour blocks in both the background and foreground of CAPTCHA images. Research conducted by Yan et al.[108] showed difficulties for people with normal vision in recognising these CAPTCHAs, including 8% of males and 1% of females in North America and Europe. Xu et al.[10] also encountered usability issues in his versions of the NuCAPTCHA. These issues were present during the usability examination of those versions in which participants skipped many challenges that included highly coloured backgrounds due to the difficulty of reading the text in front.

Integration and loading problems. The animated text-based CAPTCHAs also have a major usability problem in the ability of browsers and platforms to handle this kind of technology. The main cause of this problem is that the most of animated text-based CAPTCHAs are displayed using incompatible technologies with many browsers. These issues were mostly present in mobile browsers such as those included on IOS and Android. This can lead the users of these browsers to encounter problems in opening and reading the animated text-based CAPTCHAs. Animated CAPTCHAs also suffer from loading problems resulting from the number of frames used inside them. These problems can also increase the time needed to solve the animated CAPTCHA challenges.

2.6.4 *Other usability studies of CAPTCHAs*

The researchers also conducted other general studies of the AI problems in the CAPTCHA schemes. The W3C working group[106] presented one of these studies in which they highlighted the inaccessibility of visual CAPTCHAs for disabled web users, including blind and poor vision users and those with learning disability problems such as dyslexia. They also discussed the use of alternative solutions to the visual CAPTCHAs such as the audio CAPTCHAs instead of discussing improvements to the usability of challenges. In connection with this study, Tim Converse[107] argued that the CAPTCHA always poses an accessibility problem, as it is a test of human sensory or cognitive abilities, which some humans can lack in general.

Other research included comparisons between the text-based CAPTCHAs and AI problems-based CAPTCHAs in terms of usability. Brodić et al.[109] presented the most recent study which examined two types of standard text-based CAPTCHAs and two types of image-based CAPTCHAs, which are as follows:

- Text-based CAPTCHAs:
    - The first scheme displays letters
    - The second scheme displays only numbers
- Image-based CAPTCHAs
    - The first scheme includes a set of wild life animals
    - The second scheme display images of other objects

The results of this research showed that the users could recognise the image objects in image-based CAPTCHAs faster than the text. The results showed the fastest time recorded for wild life animals (between 1 second and 23.14 seconds) and the slowest time recorded for text (between 3 seconds and 59.78 seconds). Other research was by Elie et al. in which he compared the usability of sound-based CAPTCHAs with the usability of the text-based CAPTCHA. The results of this research showed that the text challenges were much easier for the human than the sound challenges. Based on these results, the text challenges required an average of 9.8 seconds to be resolved with an accuracy of 71%. Sound

challenges required an average of 28.4 seconds to resolve the challenges with an accuracy of 31.2%.

Chapter 3. Wikipedia CAPTCHA

In this chapter, we have studied the robustness of the Wikipedia CAPTCHA scheme, which is a widely known, text-based CAPTCHA used on Wikipedia websites. This CAPTCHA scheme was one of the last text-based CAPTCHA designs based on segmentation and recognition of resistance principles. The security of this CAPTCHA relies mainly on a "lines as clutter" mechanism that increases the robustness of the CAPTCHA against segmentation attacks. It also relies on the high distortion of characters. The mechanism and the noises also impair the ability of recognition engines to recognise segmented characters. Even so, our attack achieved a 70% success rate in recognising and segmenting this CAPTCHA.

3.1 Introduction

In this chapter, we present the first answer to our second question about the capability of segmentation resistance mechanisms that provide a level of resistance missed in animated schemes. In particular, we present a novel attack on the "lines as clutter" mechanism implemented by the Wikipedia CAPTCHA. This CAPTCHA adopts the "lines as clutter" mechanism by adding noisy lines around distorted characters and the gaps between them.

The robustness of this mechanism is highly dependent on the distortion methods used against the characters. These methods could help generate infinite varieties of shaped images of the letters included in text challenges. These images could also include ones that contain partially displayed characters in the shape of noisy lines. Combining distortion methods with the "lines as clutter" mechanism increases the change in the shape of the text or the noisy lines and provides better resistance to segmentation attacks that identify either. In addition, these combinations could cause characters in the text to be close to the shape of other characters and, as a result, lower the recognition rate of the characters.

In this chapter, we attempt to investigate the capability of computer programs to reverse the adaptation of the "lines as clutter" mechanism in this CAPTCHA. The investigation includes an attack that detects and remove the noises around the characters and in the connection areas between them. The attack also includes other steps to remove the empty dots inside the characters before analysing the components to identify if they are related to a single letter.

In this chapter, we also discuss the steps included in our attack and show how we used them to reverse the “lines as clutter” mechanism to lower the distortion level of the text. Our intention in this chapter is to show that relying on this mechanism may not improve the resistance of the CAPTCHA to attacks.

To the best of our knowledge, the work presented in this chapter includes an analysis that is critical to our research. It reports the most successful attack on a Wikipedia CAPTCHA that includes one of the most robust mechanisms and highlights vulnerability problems in one of the segmentation resistant mechanisms to the novel attacks. Our work also contributes to answering the second question of our research and helps us determine the main effect of noises even if they are included in the current animated schemes, as these schemes are still developed using standard techniques (as discussed in chapter 10). This attack achieved a success rate of above 70% in both segmentation and recognition attacks against this CAPTCHA. This result proves the capability of our attack to compete with the human in solving these challenges, as the success rate of this attack is very close to the human’s accuracy rate in solving these challenges.

The sections in this chapter are organised as follows. Section 3.2 provides an overview of the targeted CAPTCHA scheme. Section 3.3 describes our attack followed by the evaluation in section 3.4 and our results and the limitations of our attack in section 3.5. We then discuss the other attacks against CAPTCHA in section 3.6 followed by our defence against it in section 3.7 before a summary in section 3.8.

3.2 The characteristics of the Wikipedia CAPTCHA

The Wikipedia CAPTCHA (as seen in figure 3.1) implements a high distortion mechanism that commonly relies on the noises, affine transformations and image warps. However, this type of CAPTCHA also sometimes relies on the segmentation resistance mechanism. The key characteristics of the CAPTCHA are the following:

- Each challenge uses a grayscale image
- Each CAPTCHA image commonly contains two words, which are English in the most challenges.

- The number of characters inside the CAPTCHA image could be between seven and nine.
- The characters inside the CAPTCHA are highly distorted using affine transformations, local warps and global wrap
- Each character in the CAPTCHA image contains noises around it
- The characters inside the CAPTCHA are commonly segmented from each other. However, some images could contain one or two words connected with each other
- A few characters may also have some parts removed from them, which may cause the CAPTCHA character to have some empty areas inside them. In addition, they may also cause parts of the CAPTCHA character to separate.

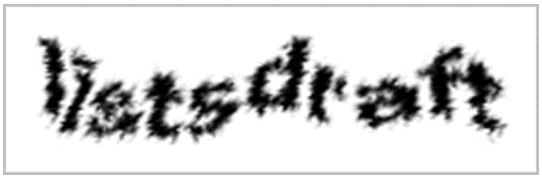


Figure 3.1. The Wikipedia CAPTCHA

3.3 Our attack against Wikipedia CAPTCHA

This attack is divided into two main steps; the first step is the segmentation attack, which includes attempts to segment the characters and clean each character from the noises and find any part that is segmented from it. The second step is the recognition attack that will be performed using a CNN engine.

3.3.1 *Segmentation attack*

The segmentation attack against the Wikipedia CAPTCHA is the first main step in our attack and includes two steps that can be represented in the following way:

- Pre-processing: this step converts the image into black and white (binary) image.
- Colour filling segmentation: this step analyses the CAPTCHA to find the non-connected character components within the text and segment them

- Segmenting the connected characters: this step analyses the CAPTCHA to find the connected characters and segment them

Pre-processing. The first method in our attack converts an 8-bit grayscale image into a binary image buffer. This method includes an algorithm that converts every grayscale coloured pixel with shade less than 128 (50%) into black, while converting the rest of the pixels into white (as in figure 3.2). After that, the method converts the resulting monochrome image into a RGB image buffer to make it receptive to the colour filling segmentation.

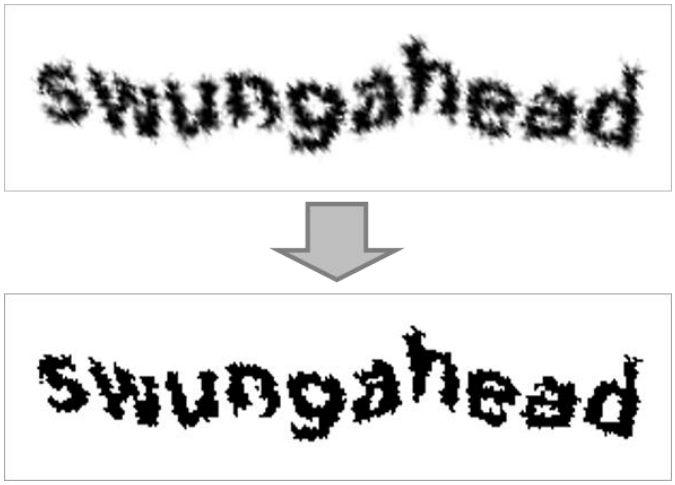


Figure 3.2. Converting grayscale image into monochrome

Colour filling segmentation. In this step, we used the colour filling segmentation attack, which separates the components by filling each with a different colour, as described by Jeff Yan and Ahmed Al-Ahmed[51]. This attack searches for each component in the image by locating its black pixels in the CAPTCHA image. After that, it colours it with a unique colour identified for the component. It then traces every black pixel beside it until all pixels in the component are filled with that unique colour.

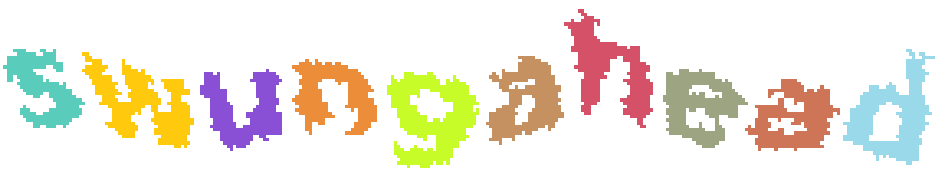


Figure 3.3. Colour segmentation attack

Finally, every coloured component will be extracted and segmented from the other components using its unique colour as shown in figure 3.3.

Segmenting the connected characters. In this stage, we are going to find the connected characters in the image. Those characters are mostly connected though one-line noise so it is so easy to discover them and segment them.

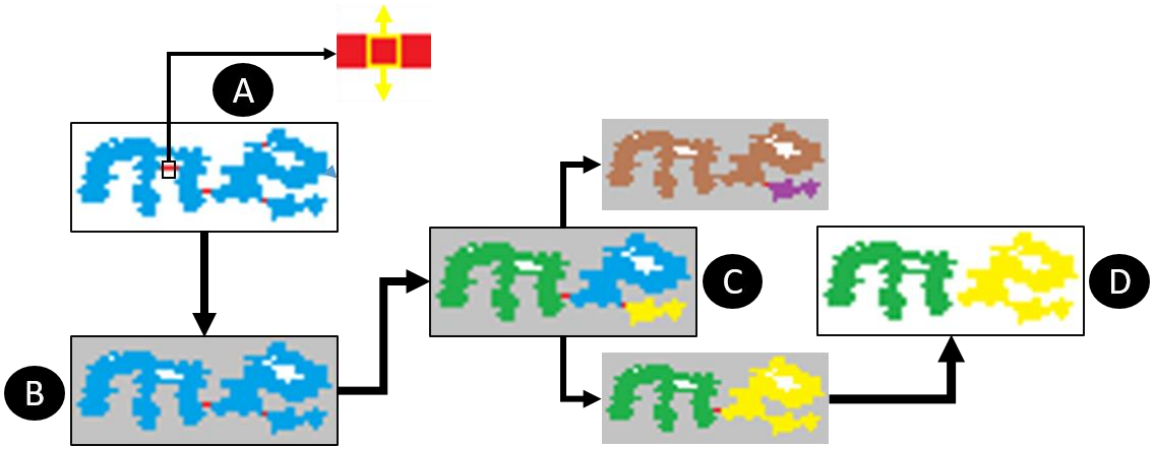


Figure 3.4. Segmentation attack

The attack works as shown in figure 3.4 as follows:

- During the colour filling attack (in the previous step), we used a method to find and catch each pixel point that resembles one-pixel thick horizontal line in each character component. This method determines if the upper and lower sides of each point are white pixels (the pixels of the background and the looped areas). Figure 3.4 (A) shows an example of where the detected pixels are marked by a red colour.

- After that, it excludes the detected pixel lines that are linked with loop components such as the empty components inside “o”, “p”, “q”, “e”, “a”, “d” and “b” by flooding the background with one colour. It then removes every detected pixel if any of its upper or lower side is not coloured by the chosen colour to flood the background. Figure 3.4 (A) shows an example of where the attack flooded the background with the grey colour. It also shows how the attack excludes two components connected by loop components that are still coloured with the old background colour.

- It selects each marked line and scans its two sides within the component that contains this vertical line using a colour filling attack to determine if the total of the

pixel points counted on each side is higher than 150, and its width and height is higher than 30. Figure 3.4 (D) shows an example of this step where the attack excluded the marked line on the right side of the component as it separated the lower side component of “e”. The pixel points in that side were less than 150 and the width and height of each is higher than 30 pixels. It still did not identify the line on the left side, which separates “e” and “m”, as the pixel points on that side are higher than 150, and the width and height is higher than 30. This caused the attack to select the line in the final step at this stage.

- The attack finally separates the side between the selected components. After that, the background is refilled with its old colour, after all lines have been identified in the CAPTCHA challenge. Figure 3.4 (E) shows an example of the final step in this stage.

3.3.2 *Recognition attack*

The second main step in our attack is to remove the noises and the empty holes from characters to make it easy for the recognition engine to recognise those characters, before we use them to train the recognition engine. It includes three steps:

- Remove the noise: this removes the noises around each character.
- Remove the empty holes: this step removes the empty holes inside each character.
- Train the recognition engine: this step trains the recognition engine to recognise the CAPTCHA characters.

Remove the noise. This step is considered as the main step in our attack. During this process, we aim to remove the noises from each recognised and segmented component, after we record it in an individual array.

We can identify those noises as two types of lines that are spread around the edges of each character. The first type of lines is the one-pixel noise lines shown in figure 3.5 using the red colour, while the second type of lines is the two-pixel noise lines shown in figure 3.5 in blue.

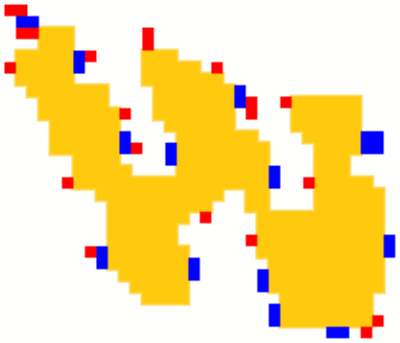


Figure 3.5. The noises marked around the character: the 1-pixel noise line is coloured by red, while the 2-pixels noise line is coloured by blue

The method used in this step included the following stages:

- **Detecting the noises.** The method in this step scans the edges of each character to find the edge of each one-pixel noisy object (marked in red in figure 3.6). This method uses an algorithm that records each pixel that is not connected with any other pixel from three different sides. It also scans the edges of each character to find the edge of each two-pixel noisy object (marked in blue in figure 3.6). It then records each pair of connected pixels that are not connected with any other pixel from two connected sides. The yellow arrows and boxes in figure 3.6 (A) shows how the method detects every noise line on the edge of the character.

- **Removing the noises.** The method removes each single pixel noisy object found by removing every pixel from its edges. It also removes each two-pixel noisy object found by removing every two pixels from its edge. After that, it scans the area besides the removed noisy object through the noise detection (first) step to check if it is connected with another noise. It then removes any detected noisy object by passing it to the noise removal (current) step. It then keeps repeating the previous steps that detect and remove the noises again, until it stops detecting noisy objects beside the removed one. For example, in figure 3.6 (B), the method detected one-pixel noisy object in the upper-left side of the text object. After that, it scanned the area around the removed one-pixel line and detected two-pixel noisy objects below it to be removed. It then detected the last two sequences of pixel noisy objects and removed them, before it stopped detecting any one or two-pixel noisy objects around them.

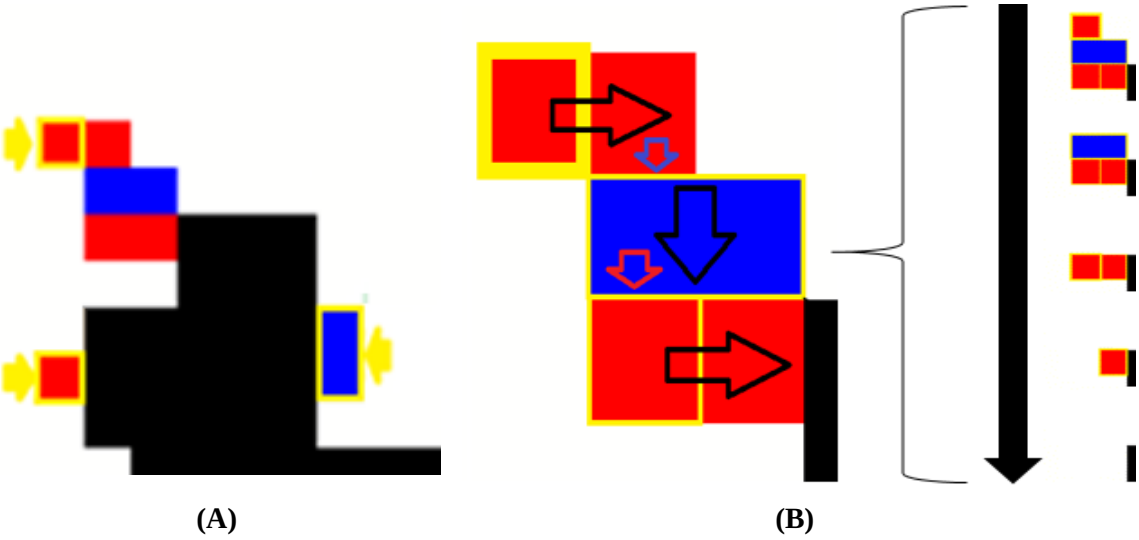


Figure 3.6. Remove the noises: (A) shows the detected noisy lines in the edges marked using a yellow square/rectangle and arrow, while (B) shows the removal process

Figure 3.7 shows the results of our attack. It also shows what the CAPTCHA image will look like, when this step is completed.

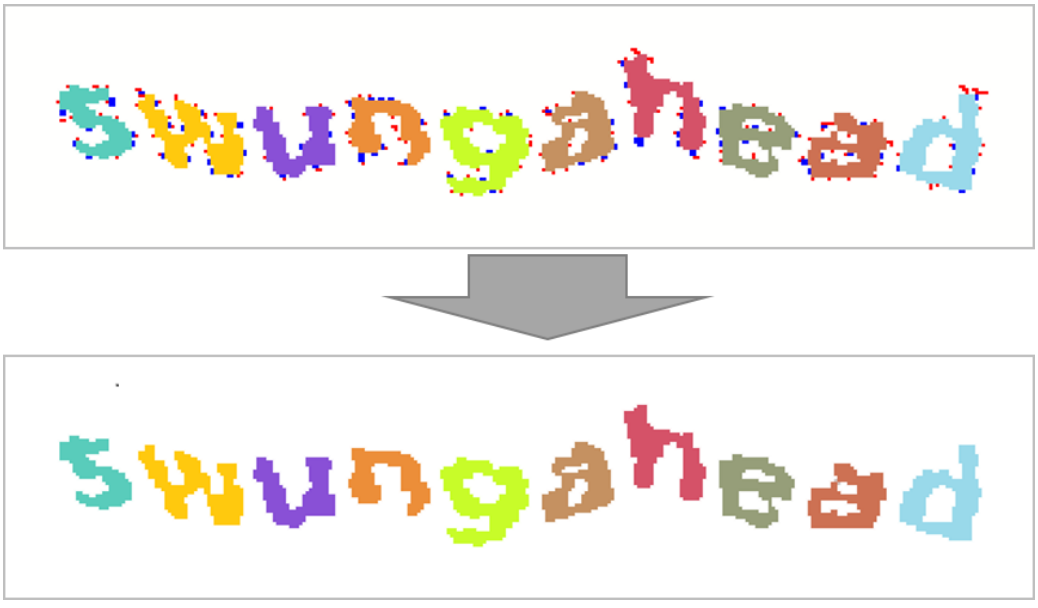


Figure 3.7. The CAPTCHA image after removing the noises

Remove the empty holes. The empty holes are small white holes inside the CAPTCHA character. These small holes indicate the places where each character has a few small parts of it erased. Figure 3.8 shows three examples of the empty holes inside the characters.

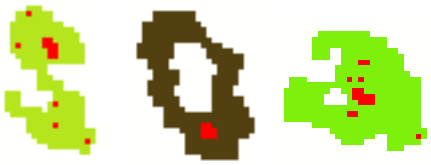


Figure 3.8. Three examples of the empty holes inside characters: the empty holes are marked in red

Our intention in this step is to remove the empty holes so we can improve the morphological look of the character and avoid any possibility of recognising its empty holes as loop components. Our attack simply removes the empty holes in the following way. Firstly, it scans each character to find the empty areas inside it. After that, it calculates the size of each of the empty areas by counting its pixels using a colour filling method. It then checks the size of each empty area to decide if it is a part of the character or if it is a removed part from the main body of the character. Finally, the method fills the empty areas with the colour of the character.

Training the recognition engine. In this step, we are going to recognize the CAPTCHA characters by using a recognition engine called Convolutional Neural Network (CNN)[110]. This engine is the implementation of a framework using Microsoft C# language code that allows training and testing against a dataset of recognisable objects that could be either handwritten digits or natural objects. Each dataset should contain two sets of images, the first of which is a set of training images input to train the engine and test it against them without distortion. The second set is the set used to test the engine against them after they have distorted.

This engine works by dividing each input image from the training set into multiple layers. It then creates a map that connects between those layers using a method called Gaussian connection. This method analyses and studies the relationship between each two layers in the image using their patterns. Then, the engine analyses the generated connection maps and tests them against the training and test sets to generate an error rate for the number of passes through the training set and the testing set. Next, the engine repeats the training for another season to reduce the error rates by modifying the connection map between the layers. The recognition engine should repeat the test for multiple seasons, until the recognition engine achieves the lowest possible error rate for the number of passes of both the training and testing sets[64].

We trained this engine to recognise characters inside the CAPTCHA images by creating a training set of 2886 sample images of characters (111 sample images for each character) and testing sets of 944 sample images of characters. After that, we used two datasets to train our engine 24 times and the result was the following:

- The engine achieved a 1.4% error rate in training, after it trained 2847 sample images of characters from the training dataset successfully.

- It achieved a 0.8% error rate in testing after it recognised 936 sample images of characters from the testing dataset successfully.

Figure 3.9 shows how the training and testing accuracy rates were improved during the training task.

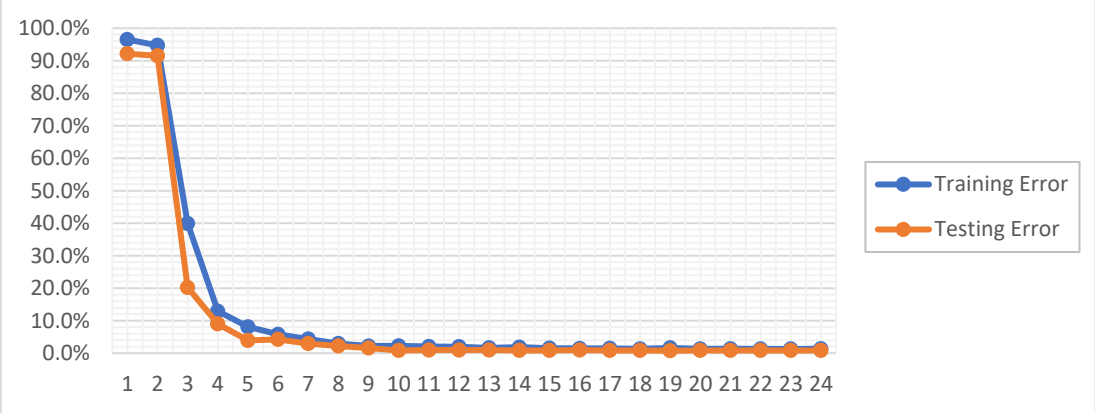


Figure 3.9. Training and testing accuracy rate during the training of the CAPTCHA

3.4 Evaluation

We discuss the results of our attack in terms of segmentation and recognition. The method used in our evaluation was based on an examination of 600 random samples of this CAPTCHA from the Wikipedia website[111]. We obtained these samples between October and November 2012. We used 100 of these as a sample set to design, train and process our attack. We saved the other 500 samples locally as a test set to be used in the final examination of our attacks. We used both samples to evaluate our attack through a simple methodology that included a visual inspection to determine the success rate of both segmentation and recognition attacks by counting the samples broken by each of the attacks individually. In addition, it included a comparison of the results of the two stages to determine the accuracy rate of our recognition attack. Our methodology also included

another visual inspection of the test set that was not broken by our segmentation and recognition attacks to determine the failures of our attacks. It also included a speed examination of our attack against 100 random samples of the test set using JAVA and C#. We also analysed the possible trade-offs between accuracy and success rate by determining the additional computational effort that could help increase both the success rate and the accuracy rate of our attack.

3.5 Results

Attack success. We firstly tested our attack against a sample set of 100 of the CAPTCHA challenges used for training purposes, and it achieved a success rate of 84% in recognising them, after it achieved a 91% success rate in segmenting them. After that, we tested our attack on 500 random test samples of the Wikipedia CAPTCHA, and it achieved an overall success rate of 70.4 (351 out of 200) segmenting and recognising the samples of this CAPTCHA. Those results include a success rate of 82.6% (413 out of 500) in segmenting the samples of this CAPTCHA, and an accuracy rate of 84.988% (351 out of 413) in recognising the segmented samples. The results also showed that the use of additional computational effort does not increase the success rate of our attack on this CAPTCHA scheme.

We have also detected some failures in the attack that can be represented in the following way:

Failure 1: segmenting characters. The segmentation algorithm that was used in this attack is only usable against the components that have a weak link between each other. However, some types of character components are still very difficult to be removed with this method due to the link between them that is very thick, while their appearance makes them very difficult to differentiate from normal characters. However, we were still able to counter this problem through the recognition engine, as the most unsegmented characters were limited to two identified sequences of characters that are either “ry” or “rt”. Figure 3.10 shows some examples of the unsegmented characters.

Figure 3.10. Unsegmented characters

Failure 2: Detect the segmented parts of some character components. This failure was caused by the segmented character components that could be recognised as two individual characters. For example, figure 3.11 shows that the character "m" was segmented into two separated parts, the first one looks like "l", while the second part looks like "n". The possible solution to this problem is to develop a new version of a recognition attack that can analyse any two or three components besides each other and recognises them as two or three characters that resemble one letter when gathered together. For example, if the recognition tool highlights three components beside each other as "l", "l" and "l", it will check if they can be three segmented parts of the letter "m".

Figure 3.11. Segmented components that are not recognised

Figure 3.12 shows the rate of failure in our attack against the Wikipedia CAPTCHA.

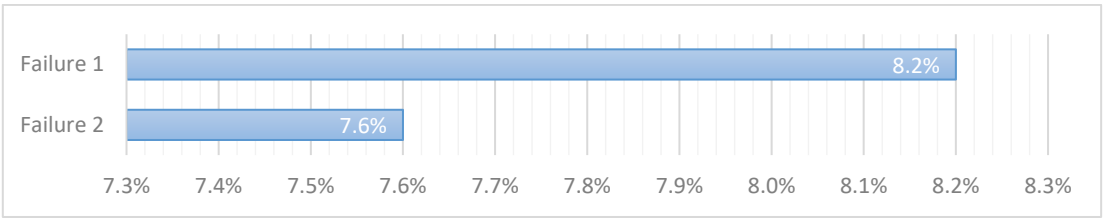


Figure 3.12. Failure rates in our attack against the Yahoo CAPTCHA

Attack speed. Our CAPTCHA attack was implemented using JAVA, and C# programming languages, and tested on a computer with an Intel(R) Core(TM) i7-4700MQ CPU 2.40GHz and 8.00 GB ram. The results show that the attack took an average of 17.67 milliseconds to analyse each animated challenge taken from the Wikipedia with standard deviation of 3.755844235 milliseconds. The shortest attacking time was 12 milliseconds, while the longest was 29 milliseconds

3.6 Other attacks

E. Bursztein et al.[59] developed an attack called DeCAPTCHA that was used against the Wikipedia CAPTCHA. This attack included five stages, which are pre-processing, segmentation, post-segmentation, recognition and post-processing. The attack firstly uses a pre-processing stage that includes a step to remove any background behind the main

characters, and then converts the image into a binary (black and white) image. It then stores the data of the image as a binary matrix to make the attack implementation through the rest of stages easier, even though it causes the pixel intensity in the CAPTCHA image to be lost. The attack during the segmentation stage then attempts to identify and segment the character by using a number of segmentation methods including the colour segmentation method that is regarded as the default method during this stage. After that, the attack uses the post-segmentation stage to process the segments through a method that normalises the size of the segment to facilitate them for the recognition attack in the next stage. This recognition attack then recognises each character through the KKN (K nearest neighbours) method that classifies the characters to recognise each segmented character generated in the previous stages. The results of the recognition attack are analysed during the post-processing stage using a dictionary of words. This attack achieved a 25% success rate against the Wikipedia CAPTCHA as records show. However, these records do not show any data about the time taken to break the CAPTCHA challenges.

Other attacks were developed after our attack. These attacks included a different attacking strategy that combined the automated segmentor mechanism based on an attack by Chellapilla[45], which uses the recognition engine to create a heat map to identify the position of the characters, and the other segmentation mechanisms used against the current CAPTCHAs. One of these attacks was developed by Bursztein et al. in 2014[79] and which achieved a success rate of 28% against the Wikipedia CAPTCHA. Another attack that used a similar technique was implemented by Yan et al. in 2015[80]. This attack achieved a success rate of 23.8% against the Wikipedia CAPTCHA.

3.7 Defence

The best way to make the Wikipedia CAPTCHA scheme more resistant to our attack is by simply combining the segmentation technique with distortion techniques that are used inside the CAPTCHA scheme. We can also improve the distortion technique after we exclude the current noise distortion techniques, as attacks could break them easily, as we proved through our attack.

However, we can still use the noise distortion through another method by implementing some type of noise line that looks similar to the font used in the CAPTCHA image. This

method may make the noise lines more difficult to detect and to be removed by the CAPTCHA attacking bots.

3.8 Summary

Our attack has demonstrated that the new recognition resistance mechanism for text-based image CAPTCHA that is implemented by Wikipedia is flawed. This attack has been implemented using a new combination of simple technologies to break this type of CAPTCHA and its resistance mechanism. The results show that this mechanism can be defeated with a high success rate. Therefore, using lines around the characters does not improve the resistance of the CAPTCHA, because they are predictable and easily removed from around the characters. The lines take forms that make them appear different to the CAPTCHA characters in order to make the CAPTCHA usable and easier to read.

Chapter 4. KillBot professional CAPTCHA

In this chapter, we examine an animated text-based CAPTCHA scheme known as the KillBot professional CAPTCHA. This scheme animates characters using different effects including blurring, fading, rotation, scaling and moving up and down. Few of these effects showed resistance to attacks, including the rotation effect. However, our attack broke those effects with a high success rate of 71% against the samples of this CAPTCHA. In this chapter, we also discuss the attack which adopted an improved method from the previous attacks deployed against it. We also present a number of ways of defending it.

4.1 Introduction

In this chapter, we have undertaken an attack on an "animated distortion" mechanism deployed by the KillBot professional CAPTCHA scheme. This scheme is a commercial and animated text-based CAPTCHA used by the United State Federal Government[86]. It uses many animated distortion methods that combine animated effects, such as scaling, rotating, blurring, fading and vertical movements.

These methods aim to add time dimension to the character distortion, so it can increase resistance to recognition attacks. Some of these methods were difficult to break with existing attacks such as rotation changes. A recent study showed that this method could limit the recognition success rate of current attacks. This limitation is more noticeable when the rotated character belongs to confusing characters, which are "n", "u", "p", "q", "b", "d", "N" and "Z". These characters are difficult to recognise when they are rotated by more than 90 degrees.

We targeted this animated text-based CAPTCHA with an inspired attack of Nguyen et al.'s "character extraction" mechanism. This attack includes methods that catch the characters that appear in a way that can be easily recognised by recognition tools. These methods are aimed at detecting all the frames that display CAPTCHA characters in a full image. It then analyses the frames to find the cleanest image of the character so the recognition engine can identify it. Our attack also included a method that analyses rotated characters, so it can find a recognisable image of the confusing ones.

This attack achieved a success rate of 70% against the KillBot CAPTCHA. The results reveal more information about the weaknesses of the animated changes used in the

CAPTCHA and proves the ability of our attack to match humans in recognising the animated characters. It also shows the capability to defeat the human in recognising rotated characters. To our knowledge, this is the first attack that has used a method to extract recognisable images of rotated characters. This method has resulted in an increased success rate of tracking attacks against a CAPTCHA.

In this chapter, we are going to summarise the methods used in our attack. We also discuss the results and the defence against our attack. In addition, we discuss the other attacks and how our attack shows a huge improvement in comparison with them.

The rest of the chapter is organised as follows. Section 4.2 discusses the related works. Section 4.3 provides details of the attack against the KillBot CAPTCHA. Section 4.4 discuss the evaluation of our attack followed by the results in section 4.5. This is followed in section 4.6 by details of the attack against the KillBot CAPTCHA. The defence against our attack is detailed in section 4.7 with a summary of our attack in section 4.8.

4.2 The characteristics of the KillBot professional CAPTCHA

The KillBot professional CAPTCHA comprises a variety of CAPTCHA designs. Each design contains five character changes that change themselves in specified vertical columns. All these changes are implemented using advanced methods. Each method includes blurring and fading effects in its design and can combine with other effects such as scaling, rotating and vertical moving. The background of this scheme could include rain drops, and characters that move in one direction, falling bars and moving lines.

Our target is the major design of the KillBot professional CAPTCHA shown in figure 4.1. As this design includes all the effects included in other designs it can give more detailed information about the how our attack performs against all designs of the KillBot professional CAPTCHA. The features of this design are as follows:

- Each challenge uses five random characters that can include numbers, capital and lower-case letters
- All characters can be displayed using the same font although they may appear in different sizes.

- The characters are coloured by using a range of two colour sets - the bluish and the grayscale.

- The first two characters move vertically up and down, while they are blurred and faded in and out. However, the speed of the blurring and fading is the same as the vertical movement. This match in speed causes the characters to appear not to have faded in the specified vertical range in the image.

- The third character scales, as it is blurred and faded. The vertical scale used may lead the character to be displayed, in some instances, in two different sizes.

- The fourth character only blurs and fades in and out without moving or changing.

- The fifth character rotates, blurs and fades. In addition, it also scales in some challenges. The character may stay rotated to a specified degree through the whole animated frames in some challenges or may rotate at all times.

- The animated background contains a lightly coloured background image. It includes other animated noises such as raindrops and semi-transparent characters that move horizontally.

Figure 4.1. The major design of the KillBot professional CAPTCHA

4.3 The attack against the KillBot professional CAPTCHA

In this section, we discuss our attack on the KillBot professional CAPTCHA. This attack uses a screen capture to catch the frames needed during the attack from the animated challenges. It then uses them in an attack process that includes four steps:

- The pre-processing
- Segmentation attack
- The character extraction process

- Recognition attack

4.3.1 *The pre-processing*

In this step, the attack uses a series of methods to remove the background image and the animated noises included in the background. The first method removes the lightly coloured background image and the semi-transparent characters and converts all frames from RGB coloured frames into an 8-bit grayscale frames. The resulting frames display the pixel using a range of grayscale shade colours between 0% (the black-coloured pixels) and 100% (white-coloured pixels). After that, the method removes the pixels with the grayscale-shaded value of above 75% from each frame. Thus, it removes both the background and noisy characters, because their shade values are above 75%. The second method removes the raindrops by detecting their highly thin, rounded patterns. The resulting frames show the only characters in a ranged grayscale value of between 0% and 75% in front of a white background. Figure 4.2 shows an example of the character cleaning.

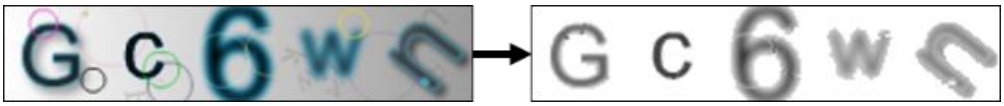


Figure 4.2. Noise cleaning

4.3.2 *Segmentation attack*

Next, the attack segments the characters using a method implemented by Vu Duc Nguyen et al. [91] that includes two steps. In the first step, the attack uses the CAPTCHA frames to create a mapping image called a pixel delay map (PDM). This map shows the lengths of time that foreground pixels take to appear in each coordination point within the animated image. We can mainly locate these pixels using their colours that are distinct from the background. In our attack, we use a method that locates the foreground pixels using their grayscale value, which is less than 100%, in each frame. It then calculates the ratio of frames that contain foreground pixels to the total number of extracted frames in each coordinated point. After that, it creates the PDM map that shows the resulting value of calculations in each coordinated point as the determined length of time.

Figure 4.3 shows an example of our PDM map. It also includes a metre that shows the colour value of each possible resulting value of calculation. We determined this value as the percentage of frames that contain foreground pixels to the total numbers of frames.

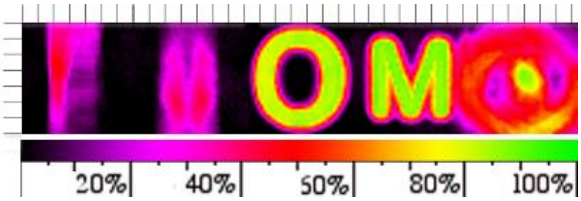


Figure 4.3. The PDM map

Our attack then uses another method to locate the vertical areas that contain each CAPTCHA character in the PDM map. This method selects every y-coordinated line that contains pixel points with a ratio value greater than 10%. After that, it identifies the vertical columns containing a sequence of five or more selected lines as the vertical columns where the characters are located. Then, it segments those vertical columns by adding two vertical lines in their sides.

After that, the attack uses a second method to segment vertical columns that contain connected characters. This method locates vertical columns containing connected characters by checking their widths. If this method finds that the width of the column is greater than a specific value, it determines that it contains connected characters. We decide this value as the greatest width value of a single character in which we can calculate using the following algorithm:

$$M = W/C$$

in which W is the width of the animated image, and C is total number of characters in the text, which equal to five. Finally, it segments the columns by locating the vertical line with the smallest ratio value within that column in the PDM.

Figure 4.4 shows an example of the character segmentation using the PDM. In this figure, the first method segmented the first three characters by adding two blue vertical lines beside the horizontal sides of each character. It also added a vertical line to the left side of the fourth character and to the right side of the fifth character. The second method then detected a failure to segment those two characters by adding vertical lines between them. Thus, it segmented them by adding another line between them (marked in figure 4.4 using the white colour).

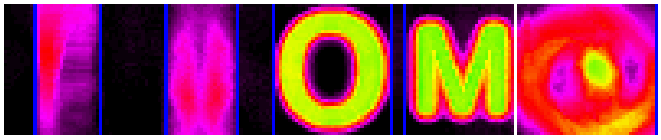


Figure 4.4. Segment the characters vertically using the PDM

4.3.3 *The character extraction process*

In this step, the attacks analysed five animated characters and extracted them with four major methods that check their movements to select the best image of those characters during the extraction process. The attack extracted those characters through four extraction methods.

Extracting the vertical moving characters (first and second characters). The first method exploits the distortion method combines the fade and blur effects with vertical movement. This method uses an algorithm to select the frames that display the whole character inside an area between two specified heights. These two heights are as follows:

- The first height is in the tenth position from the top of the CAPTCHA frames.
- The second height is in the twentieth position from the bottom of the CAPTCHA frames.

We explain this algorithm as follows:

Algorithm: selecting the range of images for 1^{st} or 2^{nd} character
Input: PDM (PDM map), $AIamge$ (Animated image), X_L, X_R (Two sides of the character)
Output: $ImgA$(the list of elected images)
1: $H \leftarrow height$ of $AImage$
2: $Y_T \leftarrow H/10$
3: $Y_T \leftarrow H - (H/20)$
4: **for each** $frame$ in $AImage$ **do**
5: **for** $x \leftarrow 0$ **to** H **do**
6: **for** $y \leftarrow X_L$ **to** X_R **do**
7: **if** all of $character$ is within Y_T and Y_B in the $frame$ **then**
8: ImgA. add($image$ of $character$ in $frame$)
9: **end if**
10: **end**
11: **end**
12: **end**

This method uses a second algorithm that extracts characters in the least blurred and faded out image. The algorithm calculates the average grayscale value of foreground pixels within the vertical column of the character in each frame. It then excludes the frames where the average grayscale value of foreground pixels is less than grey (below 50%). After that, the algorithm selects the frames that display the character using a low range of grayscale values. This selection process starts by identifying the range of grayscale values used to colour foreground pixels within the vertical area of the character in each frame. It then identifies the frames that match the condition mentioned in the following equation:

$$-(V_S/V_L) \leq 0.99$$

In which V_S is the lowest grayscale value of a foreground pixel within the vertical area of the character, and V_L is the highest grayscale value.

It then lowers its selection process to one frame using one of three different methods. The method is selected based on the results of the first steps in the selection process:

The first method is used if we detect one frame that matches the condition mentioned in the first step of the selection process. This method extracts the image of the character from this frame, before it processes the final step of the selection process on it.

The second method is used if we detect that more than one frame matches the stated condition in the first step of the selection process. This method selects the frame that contains the darkest grayscale value between the selected ones in the first step. After that, it extracts the image of the character from this frame before it processes the final step of the selection process on it.

The third method is used if we do not find any frame that matches the stated condition in the first step of the selection process. This method selects the frames that contain the highest number of darkest grayscale values between the selected frames in the first step. After that, it extracts the image of character from this frame before it processes the final step of the selection process on it.

The final step in this selection process includes a method that cleans the extracted image of the character of noises. The method detects the noises through their grayscale value that is

less than three quarters of the highest grayscale value in that image. It then removes them from the extracted image of the character.

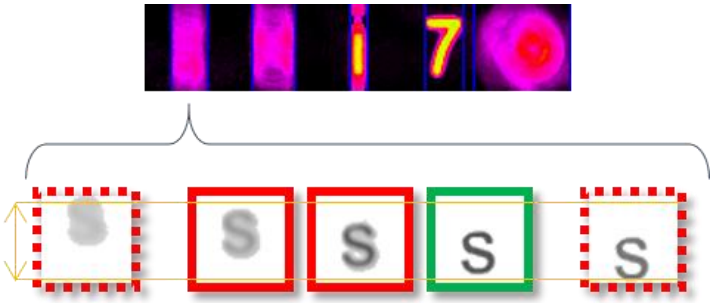


Figure 4.5. Extracting the first character: the green square identifies the selected one

Figure 4.5 shows an example of our attacking method by using it to extract the character "s" that moves vertically up and down. In this figure, we have firstly selected the frames where the character "s" appears within a range between the tenth position from the top of the CAPTCHA frame and twentieth from the bottom. After that, it analysed each frame and selected the frame where the character "s" is displayed in the least blurred and faded out image.

Extracting the scaled character (third character). This method aims to exploit the scale effect used to animate the third character and uses the PDM to determine the range between two heights that the upper side of the character reaches. The position of the first height is determined as the highest position in which the PDM of that character can give the highest value. The position of the second height is the highest position that the character can reach when this position is not at the top of the image. We can locate this position in the PDM map as the highest position in which the PDM characters achieve a value greater than zero. If the position of the second height is at the top edge of the image, we change this position to a different height. We determine this height as a tenth from the top of the CAPTCHA frames in case the first height is below it. Otherwise, we select the first height as the second height. Our method then selects all the frames where the head of the character appears inside the area between the two horizontal lines. After that, it analyses each of the selected frames to determine the frame that displays characters in the least faded and blurred image. We can explain the selection of character images algorithm as follows:

Algorithm: selecting the range of images for 3^{rd} character
Input: PDM (PDM map), $AImage$ (Animated image), X_L, X_R (Two sides of the character)

Output: $ImgA$(the list of images)
1: $Y_T, Y_B \leftarrow -1$
2: $H \leftarrow height$ of $AImage$
3: $A \leftarrow 0$
4: **for** $x \leftarrow 0$ **to** H **do**
5: **for** $y \leftarrow X_L$ **to** X_R **do**
6: **if** $PDM(x, y) > 0$ **then**
7: **if** $Y_T = -1$ **then** $Y_T = y$ **end if**
8: **if** $PDM(x, y) > A$ **then** $A = PDM(x, y), Y_B = y$ **end if**
9: **end if**
10: **end**
11: **end**
12: **if** $Y_T = 0$ **then if** $Y_B < H\ /10$ **then** $Y_T = H/10$ **else** $Y_T = Y_B$ **end if**
13: **for each** $frame$ in $Aimage$ **do**
14: **for** $x \leftarrow 0$ **to** H **do**
15: **for** $y \leftarrow X_L$ **to** X_R **do**
16: **if** top of $character$ is between Y_T and Y_B in the $frame$ **then**
17: ImgA. add($image$ of $character$ in $frame$)
18: **end if**
19: **end**
20: **end**
21: **end**

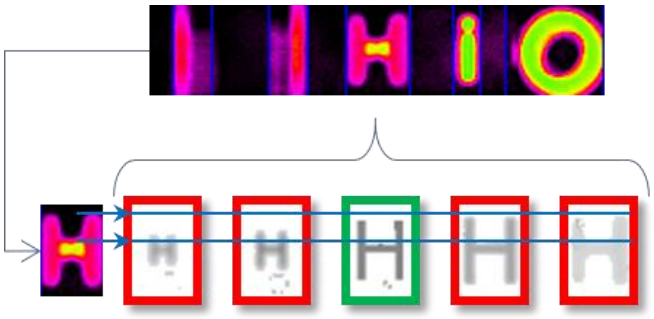


Figure 4.6. Extracting the third character: the green square identifies the selected one

Figure 4.6 shows an example of our attacking method by using it to extract the character scaled "H". In this figure, our method sets the position of the first height as the highest position in which the PDM of that character can give the highest value. It then sets the position of the second height as the highest position in which the PDM of that character can give a value greater than zero. We did not change this position, as it was below the top edge of the image. After that, it selected all the frames where the head of the character "H"

appear within a range between the two heights. It then analyses each frame and selects the frame where the character “H” is displayed in the least blurred and faded out image.

Extracting the faded/blurred character (fourth character). The attack extracts the fourth character by using the same method deployed in the earlier attack by Nguyen et al. [91]. This method extracts the pixels of the character displayed for a long time from the PDM map. It then extracts the pixels points that have a PDM value of greater than 90% of the highest PDM value recorded within the vertical area of that character.

Figure 4.7. Extracting the fourth character

Figure 4.7 shows the method used on the fourth character “a”. In this figure, the method extracted the yellow coloured pixels within the vertical area of the character inside the PDM. It did not select any other pixel, because its value is less than 90% of the highest PDM value recorded within the vertical area of that character

Extracting the rotated character (fifth character). The final method extracts the rotated character in the nearest rotated angle to its correct orientation angle. This method records the height of the fifth character in each frame. It then uses the results to calculate the average height of this character. After that, it selects every frame where the character height is greater than the average.

Next, it uses an algorithm to identify the frame where the ratio of the height of the character to its width is greatest between the selected frames. This algorithm measures the width and height of the rotated character in each frame and calculates the ratio by dividing the height of the character with its width in each selected frame. After that, it compares the calculation results of between all selected frames before finally extracting the character from the frame that records the greatest resulting value of the selected frames. This value is equal to the ratio value of confusing characters such as “b”, “d”, “p”, “q”, “N”, and “Z” when they are close to their correct orientation.

The final step in this method catches the image of the character from the determined frame and uses it as the extracted image of the fifth character. We can explain the full image extraction processor as follows:

Algorithm: image extraction process for 5^{rd} character
Input: PDM (PDM map), $AImage$ (Animated image), X_L, X_R (Two sides of the character)
Output: $Image$ ($the\ returned\ image$)
1: $H \leftarrow height$ of $AImage$
2: $Y_T \leftarrow H/2$
3: $N, Y_B \leftarrow H$
4: $Image \leftarrow new\ Image$
5: $Arr \leftarrow new\ array[length$ of $AImage][3]$
6: **for each** $frame$ in $Aimage$ **do**
7: **for** $x \leftarrow 0$ **to** H **do**
8: **for** $y \leftarrow X_L$ **to** X_R **do**
9: $T \leftarrow top$ of $character$ in $frame$
10: **if** $Y_T < T$ **then** $Y_T = T$ **end if**
11: **if** $Y_B > T$ **then** $Y_B = T$ **end if**
12: $W \leftarrow width$ of $character$ in $frame$
13: $H \leftarrow height$ of $character$ in $frame$
14: $Arr[number$ of $frame] \leftarrow \{T, H/W\ , image$ of $character$ in $frame\}$
15: **end**
16: **end**
17: **end**
18: **for each** R in Arr **do**
19: **if** $R[0] > (Y_T + Y_B)/2\ \&\ R[1] > N$ **then** $N \leftarrow R[1], Image \leftarrow\ R[2]$ **end if**
20: **end**
21: **return** $Image$

Figure 4.8 shows an example of our method by using it to extract the rotated character "u". In this figure, our attacking method determined that the average height is 17 and selected all the frames where the width of the character is greater. After that, it calculated the ratio of the height of the character "u" to its width in each of the selected frames. It then used the results to locate frames where the ratio of the height of the character to its width is the greatest between the selected frames. This ratio of the character was equal 1.818, as the figure shows.

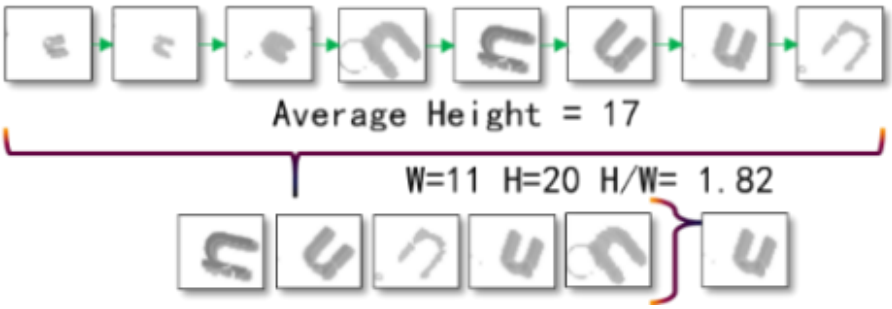


Figure 4.8. Extracting the fifth character

4.3.4 *Recognition attack*

In this step, we used the Convolutional Neural Network (CNN)[110] engine to recognise the extracted characters as in our attack against Wikipedia CAPTCHA. We trained this engine to recognise characters inside the CAPTCHA images by creating a training set of 2240 sample images of characters (40 sample images for each character) and testing sets of 400 sample images of characters. Some of these samples were manually rotated to different degrees to help the program recognise the rotated characters. After that, we used two datasets to train our engine 24 times and the result was the following:

- The engine achieved a 1.3% error rate in training after it trained 2209 sample images of characters from the training dataset successfully.
- It achieved a 1.7% error rate in testing after it recognised 393 sample images of characters from the testing dataset successfully.

Figure 4.9 shows how the training and testing accuracy rates were improved during the training task.

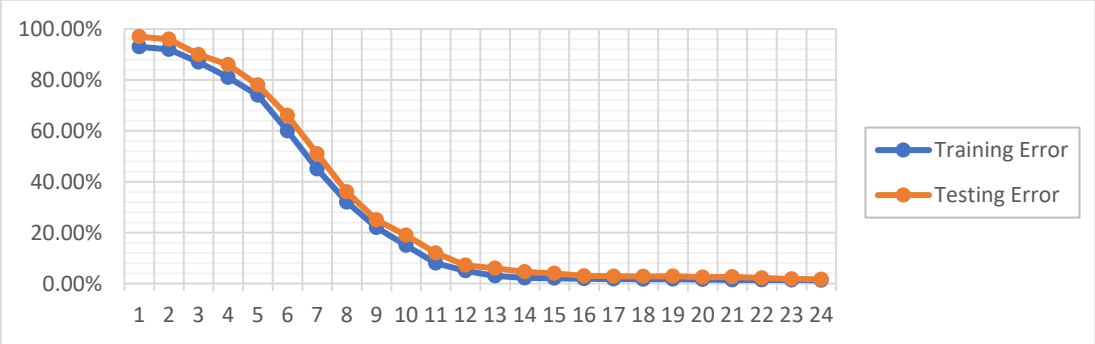


Figure 4.9. Training and testing accuracy rate during the training of the CAPTCHA

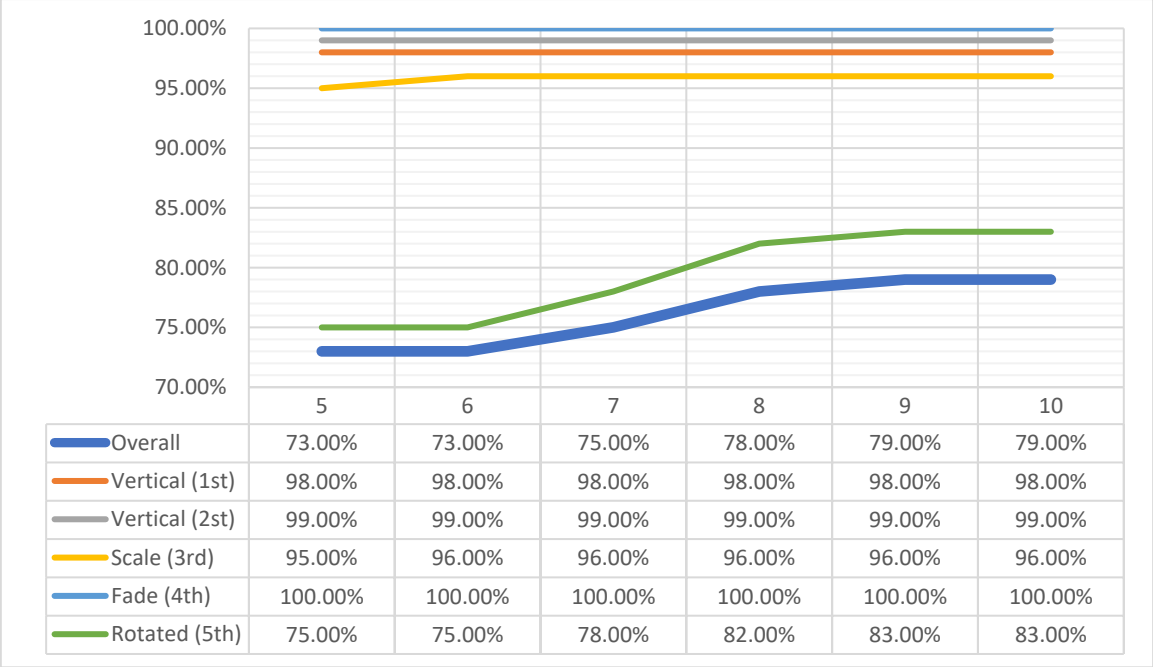
4.4 Evaluation

We discuss the results of our attack in terms of segmentation and recognition using the same method used on the Wikipedia CAPTCHA. Our evaluation in this chapter was based on an examination of 200 random samples of this CAPTCHA. We obtained some of those samples through an automated screen capture designed using C# between June and July 2013 from NotOneBit website[86]. We used 100 of these samples in designing and training our attack. We saved the other 100 samples locally to be used as text set in the final examination of our attacks. We did not include any sample used during the design of the attack in the evaluation process of our attack.

4.5 Results

Success rate. We firstly tested our attack against a sample set of 100 of the CAPTCHA challenges used for training purposes, and it achieved a success rate of 93% in recognising them, after it achieved a 100% success rate in segmenting them. After that, we tested our attack on 100 random test samples of the KillBot professional CAPTCHA, and it achieved an overall success rate of 71% (71 out of 100) segmenting and recognising the samples of this CAPTCHA. Those results also include a success rate of 100% (100 out of 100) in segmenting the samples of this CAPTCHA.

The use of this attack in scanning each animated challenge for a default time of 5 seconds (50 frames extracted from each challenge at rate of 10 FPS) is essential in achieving our recorded accuracy and success rate against this CAPTCHA scheme. The increase of this default time could help increase the accuracy rate of our attack in recognising the rotated characters, as a few of those characters may not appear in their correct orientation for a maximum of 10 seconds. We could verify this finding by running our examination against live samples of this scheme from the developer's website in which we used different extraction times on 100 samples. The results showed that the accuracy rate in recognising the rotated characters reached a maximum of 83% after we set the default time to 10 seconds. It also showed that the average rate of increase per second was 1.8166%. The recognition rate of other animated characters remained the same except the scaled (third) character that was slightly increased to 1%. As a result, the overall success rate increased from 73% to 79% with an average rate of 1.2% per seconds. Figure 4.10 shows the increase of accuracy rates in recognising each animated character and the overall success rate of our attack per seconds.



	5	6	7	8	9	10
Overall	73.00%	73.00%	75.00%	78.00%	79.00%	79.00%
Vertical (1st)	98.00%	98.00%	98.00%	98.00%	98.00%	98.00%
Vertical (2st)	99.00%	99.00%	99.00%	99.00%	99.00%	99.00%
Scale (3rd)	95.00%	96.00%	96.00%	96.00%	96.00%	96.00%
Fade (4th)	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Rotated (5th)	75.00%	75.00%	78.00%	82.00%	83.00%	83.00%

Figure 4.10. The increase in accuracy/success rate per scanning round

Attack speed. Our CAPTCHA attack was implemented using JAVA and C# programming languages, and tested on a computer with an Intel(R) Core(TM) i7-4700MQ CPU 2.40GHz and 8.00 GB ram. We ran this test using both screen-capture and our attack against 10 samples from NotOneBit website[86] setting the default screen-capturing time to 5 seconds to verify the accuracy of our speed examination. The results show that the attack took an average of 5161.822917 milliseconds to analyse each animated challenge taken from the KillBot professional with standard deviation of 100.346347 milliseconds. The shortest attacking time was 5010 milliseconds, while the longest was 5485 milliseconds.

4.6 Other attacks

Vu Duc Nguyen et al.[91] have developed an attack that has achieved 18% success against KillBot professional CAPTCHA with an average time of 5.8 seconds. This attack identifies the areas that contains the CAPTCHA characters and segments them vertically using the pixel delay map (PDM). Then, the attack extracts the fourth character through the PDM method that catches its image from the PDM. The first and second characters are extracted through an approach using the CL method to create a fixed line in the tenth position from the top of the CAPTCHA frames. The attack uses this line to catch the animated character, when its head touches the bottom of the line. The third character is also extracted through the catching line. However, the position of this line is determined based on the highest

position of the area where the PDM of the third character can give the highest value. The fifth rotated character is extracted using a method that catches the character when it reaches a position where the pixels of the character reach the maximum height.

There are many changes and developments in the current attack in comparison to the earlier attack against the KillBot professional CAPTCHA. Those differences can be summarised as follows:

- The current attack uses a developed version of vertical areas segmentation using the PDM, as it can recognise the characters that are connected through the pixel delay map for a length of time.

- The current attack uses a technique to detect the least faded and blurred image of the animated character. However, the earlier attack depends only on the PDM and catching line to detect the character and extract it. The method used to scan the area is important, especially in cases where the character is very small.

- Our attack detects the animated characters that are moving up and down through the analysis of the horizontal area between two heights (horizontal lines). Yet, the earlier attack caught the character by using one horizontal catching line at the top of the CAPTCHA to detect the character and extract it, when it touched this line from the bottom.

- The method developed in the earlier attack to extract the third character through the catching line has a major error. This major error is when the attack determines the position of the catching line as the highest position of the area where the images of the scaled character meet through the PDM. This error may lead to the position of the catching line being set in the wrong position. This condition is more visible when the images of the scaled character meet in an area below the head of the image of the scaled down character. However, the current attack uses a different method to analyse the images of the scaled CAPTCHA through a specified range between the scaled up and down images of the character. This will prevent the error in the earlier attack from happening in the current attack.

- The earlier attack extracts the rotated character using a simple technique that catches the character when it reaches the greatest height. This way of extracting the rotated characters causes a major problem with the confusing rotated characters like “n”, “u”, “p”, “q”, “b”, “d”, “N” and “Z”. The current attack has resolved this problem by using a different technique that depends only on the pattern of the character, instead of its position.

4.7 Defence

Some methods need to be added to defend against our attack as follows:

- An approach is needed to make the character move randomly in different directions to avoid being detected by the horizontal columns

- The CAPTCHA needs to add animated horizontal and vertical scale to the rotation effect. This addition prevents the attacks from detecting the correct form of the characters through their width and height.

- It needs to include other distortion methods to make the animated characters unidentifiable from their appearance

- The segmentation resistance principle could also play a major role in securing this kind of animated CAPTCHA

4.8 Summary

We exploited the security vulnerability of all the animated methods used by the KillBot professional CAPTCHA through our attack. The attack included several methods to catch the animated characters in a clean image, one of which showed the rotated character in the closest image to the correct image. Our attack achieved a 71% success rate and showed the false of sense of security of this CAPTCHA. Our results show the weakness of the methods used to animate the characters against the attacks and the ability of the computer to beat the human in recognising the individual characters in the animated challenges. This shows the importance of resistance mechanisms omitted from this CAPTCHA, such as the segmentation-resistance mechanisms.

Chapter 5. CAPTCHANIM

In this chapter, we examine another animated text-based CAPTCHA which uses the "animated distortion" mechanism. This animated text-based CAPTCHA is known as CAPTCHANIM and includes a variety of CAPTCHA designs that we can divide into two categories. The first category shows only a few distorted characters that are scaled vertically from left to right. The second discrepancy includes an additional step that moves the number of distorted characters horizontally, mainly inside frames but sometimes outside them. Our target is a version of this scheme that distorts the characters using a horizontal deformation method. This method shifts the shapes of the characters horizontally into different directions. We examined this target with an attack that achieved a success rate of 85% in segmenting and recognising it. This attack identified the characters that go partially out of the frames after correcting the order of the second category. It then analysed each of the animated characters to determine the least distorted version of the character and then extracted it.

5.1 Introduction

In this chapter, we present an attack on a different form of the "animated distortion" mechanism than that used in the KillBot CAPTCHA. This form is implemented by CAPTCHANIM, which is an animated scheme created by Technion. The scheme includes varieties of CAPTCHA designs which are distinct in the number of random characters used inside each challenge and their fonts, colour and background colour. The "animated distortion" used in this scheme was implemented with a variety of CAPTCHA designs. These designs show only a number of distorted characters that are scaled vertically up and down from left to right. These varieties can be divided into two categories; the first shows only a few distorted characters animated within a few vertical columns while the second shows distorted characters that move horizontally in one direction. Both categories may also distort the CAPTCHA characters through a horizontal deformation method. This method shifts the shapes of those characters horizontally in different directions.

Our targets in this chapter are the two categories of this CAPTCHA that use the highest levels of distortion of horizontal deformation. These targets present a version of the CAPTCHA that can withstand attacks through the horizontal deformation method. This method causes the characters to be distorted by shifting their bodies into two horizontal

sides. Therefore, it could prevent the attacks from identifying the positions of the characters through the CL and PDM methods. The method could also play a major role in resisting these attacks when it is used on the second category of CAPTCHA. This feature is mainly centred on effects that cause characters to move out of the frame on some random frames. These effects can prevent attacks from distinguishing between the frames where the characters move in and the frames where they move out.

Our goal in this chapter is to answer a question about the role of tracking methods in improving the accuracy rate of the recognition attack against this CAPTCHA. We aim to achieve this goal through a tracking method that finds the lowest distorted image of the animated character. Another aim is to answer a question about the ability of the attacks to catch a full image of the characters that pass through the edges of the animated image in the second category. The question relating to the main aspects of this CAPTCHA is not answered by the previous attacks and remains a challenge because of the animated distortion method. This method could cause the characters to move partially out of the animated image in random frames. It also causes the frames that contain those characters to be less distinguishable from the frame where characters are fully displayed in that image.

With our attack on this CAPTCHA, we present technologies that could answer the two main questions in our current research on the animated text-based CAPTCHA scheme. Such an attack could help us identify the vulnerability of this scheme and find an answer to increasing its defence against future attacks. This attack achieved a success rate of 85% in segmenting and recognising this CAPTCHA, which is the highest success rate recorded to date. This success rate could also match the possible human accuracy in answering the CAPTCHA challenges, taking into our account the complexity of our target.

The sections in this chapter are organised as follows. Section 5.2 provides an overview of the targeted CAPTCHA scheme. Section 5.3 describes our attack followed by our evaluation in section 5.4 and results in section 5.5. We then discuss other attacks implemented against it in section 5.6 followed by our defence recommendations in section 5.7, before we summarise our findings in section 5.8.

5.2 The characteristics of CAPTCHANIM

By looking at 100 samples of CAPTCHANIM, we can present its characteristics as follows:

- The CAPTCHA includes a number of characters ranged between four and six for each challenge.
- All characters used in our CAPTCHA scheme are capital letters.
- The characters could be displayed by using one font in each challenge that is different from those used in other challenges.
- The animated challenges only show the edges of the animated characters, while they leave the main bodies of the character filled using the background colour.
- Each challenge could use different foreground and background colours for each challenge.
- The CAPTCHA designs can be divided into two categories. The first category shows the characters animated in their position while the second category shows the characters move horizontally.
- Both categories showed the distorted characters scaled vertically in respective ways from left to right.
- Both categories may also include an additional animated approach called horizontal deformation methodology which distorts the characters by shifting their shapes horizontally in different directions.
- There are multiple levels of character distortions using the horizontal deformation method. The highest level is the most challengeable one as it increases the distance and speed of the character shapes that shift to their maximum.
- The second category could move the number of distorted characters horizontally in one horizontal direction from left to right or right to left.
- The animation method used in the second category displays a full sequence of characters divided into two and swapped horizontally on the sides of a full vertical line. The content on the right side of the vertical line is the left side of the text, and the content on the left side is the right side of that text.

- The start of the character sequence in the second category has been marked using a vertical line to help the user identify the correct order of the characters in each frame.

- The horizontal movement of the character sequence in the second category could happen mainly in the CAPTCHA frames. Yet, it could also happen out of them randomly.

- The use of horizontal deformation included in the second category could cause several vertical parts of the animated characters that pass through the edge of the animated frames not to be shown on any side of the animated frames.

Figure 5.1 (A) represents the first category while figure 5.1 (B) shows an example of the second category.

(A) **(B)**

Figure 5.1. The two categories of the CAPTCHANIM

5.3 Attacking CAPTCHANIM

Our attack targets a range of different designs of the CAPTCHANIM, including the most complicated designs that use the highest level of horizontal deformation distortion. Each challenge is an animated GIF image. Each image is extracted from the website and separated into individual frames through our attack. After that, those frames are used to run major steps of our attack, including:

- The pre-processing

- Correcting the order of the characters (in case the challenge belongs to the second category)

- Segmentation attack

- The character extraction process

- Recognition attack.

5.3.1 *The pre-processing*

Firstly, the attack detects the background colour by scanning the animated frames to find the highest number of pixels displayed using the same colour. After that, the attack recognises those pixels as the pixels of the background and transfers their colour into white. Next, it identifies the other pixels which are displayed using different colours as the pixels of the CAPTCHA characters. It then transfers their colours into black. We can explain the algorithm used in this step as follows:

Algorithm: The pre − processing step
Input: $AImage$ (Animated image)
Output: A$Image$ (Animated image)

1: $background \leftarrow Unknown\ Colour$
2: **for each** $frame$ **in** $AImage$ **do**
3: **if** $background = Unknown\ Colour$ **then**
4: Identify all the $colours$ used in the $frame$
5: determine "the $colour$ used in the most pixel points in the $frame$"
6: $background \leftarrow$ the colour used in the most pixel points in the frame
7: **end if**
8: **for** $x \leftarrow 0$ **to** $width$ of $AImage$ **do**
9: **for** $y \leftarrow 0$ **to** $Height$ of $AImage$ **do**
10: **if** $colour$ in $frame(x, y) = backround$ **then**
11: $colour$ in $frame(x, y) = white$
12: **else then**
13: $colour$ in $frame(x, y) = black$
14: **end if**
15: **end**
16: **end**
17: **end**
18: **return** AImage

Figure 5.2 shows an example of the pre-processing step.

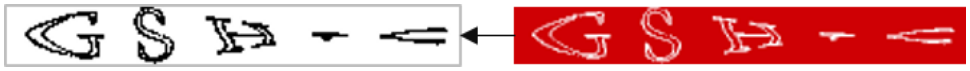


Figure 5.2. The pre-processing step

5.3.2 *Correct the order of the characters*

In this step, we use an enhanced version of Nguyen et al.'s method that checks if the challenge is from the second category, where the characters move horizontally[91]. This method finds the vertical line that helps the user identify the correct order of the characters in each frame. It then swaps the two sides of the frame that appear beside the vertical line. In our attack, we include an additional step that marks the area between the two swapped components in each frame using a green vertical line. Then, it analyses each character touching the green vertical line to check if it moved partially out of the frame to the hidden area within the green line. The analysis works through a method that checks parts of characters divided between the two sides of the green vertical area and attempts to connect them. In case the attack failed to connect between those two parts, it determines that the character moved out of the frame. It then fills the character with a special colour to prevent the attack from selecting them in the next stages. If the method detects parts of the character that are connected to only one side of the green line, it checks only the parts of that character that appear in the shape of the pixel line that moves through the green line. If the method detects multiple parts that appear in that shape, it then considers those parts as the edges of the main parts of the character that moves out of the frame and marks them using the special colour. We can explain the algorithm used in this step as follows:

Algorithm: correct the order of the characters
Input: $AImage$ (Animated image)
Output: $AImage$(Animated image)

1: $W \leftarrow width$ of $AImage$
2: $H \leftarrow height$ of $AImage$
3: **for each** $frame$ **in** $AImage$ **do**
4: **for** $x \leftarrow 0$ **to** W **do**
5: **if** $colour$ in "$frame(x, 0 \rightarrow H)$" $= black$ **then**
6: determine the chicknes of $line$ detected in x
7: $x_1 \leftarrow left\ side$ of $line$
8: $x_2 \leftarrow right\ side$ of $line$
9: $NF \leftarrow new\ frame(W, H)$
10: $NF(W - x_1 \rightarrow W, 0 \rightarrow H) \leftarrow frame(0 \rightarrow x_1, 0 \rightarrow H)$
11: $NF(0 \rightarrow W - x_2, 0 \rightarrow H) \leftarrow frame(x_2 \rightarrow W, 0 \rightarrow H)$
12: $NF(W - x_2 \rightarrow W - x_1, 0 \rightarrow H) \leftarrow Green$
13: **if** character parts detected in
$NF(W - x_1, 0 \rightarrow H)$ & $NF(W - x_2, 0 \rightarrow H)$ **then**
14: connect the two parts in $NF(W - x_1, 0 \rightarrow H)$ and $NF(W - x_2, 0 \rightarrow H)$

15: **if** connection failed **then** fill the parts in two sides using Red colour **end if**
16: **else if** character parts detected in
$NF(W - x_1, 0 \rightarrow H)$ OR $NF(W - x_2, 0 \rightarrow H)$ **then**
17: $Num \leftarrow$ the number $objects$ in shape of
18: one pixel line that moves through the green line
19: **if** Num > 0 **then** fill the parts in that side using Red colour **end if**
20: **end if**
21: $frame \leftarrow NF$
22: **end if**
23: **end**
24: **end**
25: **return** AImage

For example, figure 5.3 (A) shows three frames taken from one challenge belonging to the second category of CAPTCHA. Each frame has been targeted using the method that corrects the order of the characters by swapping two components beside the vertical line and marking the empty area between them using a vertical line as figure 5.3 (B) shows. Next, we have analysed two areas beside the green vertical line, which shows the empty area between the two components inside each frame. We have then found one frame that contains an image of the character "1" that moved horizontally out of the frame causing this character to be displayed partially inside the frame. Then, we have marked this character to exclude it from the analysis in the next stages as figure 5.3 (C) shows.

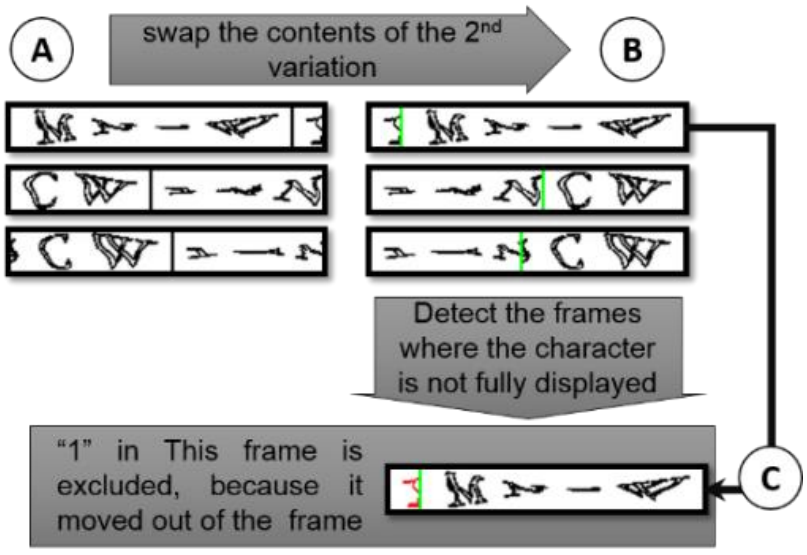


Figure 5.3. Correct the order of the characters

5.3.3 *The PDM segmentation*

In this step, the attack segments the characters using a similar approach to that used in our attack against the KillBot CAPTCHA (section 4.3.2). This approach uses Nguyen et al.'s PDM method[91] to locate and segment the vertical components that contain the characters. It also includes an attack that analyses each component to find the ones that contain a number of connected characters. This method determines the maximum width that the segmented component containing a single character should not exceed through the following algorithm:

$$A = W/N$$

in which W is the width of the animated image, and N is the total number of selected components in the PDM map. Then, it uses an algorithm that segments each component that exceeds this maximum width. This algorithm determines the weakest vertical line between them through the PDM.

Figure 5.4 shows an example of the character segmentation using the PDM in which the attack segmented the first, second, fourth and fifth characters using the first attacking method by adding two vertical lines to both horizontal sides (marked in figure 5.4 in the blue colour). The attack also added a vertical line to the left side of the third character and to the right side of the fourth character using the main attacking method and segmented them using the second method by adding another line between them (marked in figure 5.4 in the white colour).

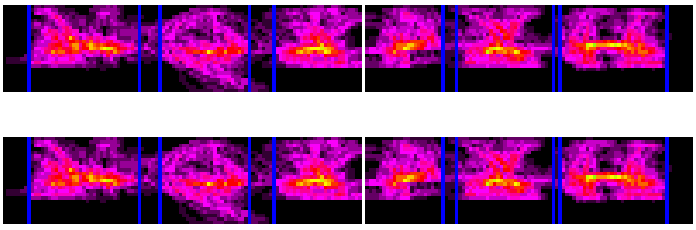


Figure 5.4. Segment the characters vertically using the PDM

5.3.4 *The character extraction process*

In this step, we process a number of methods that aim to extract the most recognisable image of each character in the animated image. We firstly use a method that selects all the frames where the characters are displayed completely. This method excludes the frames where the character is not fully displayed in case our target is the second category of

CAPTCHANIM. It then locates the frames where each character reaches its highest vertical position, which is the position where the character is displayed completely.

Next, we use a method that checks the number of selected frames where the character reaches its highest position. If the number of the frames is three or less, our attack determines that the character as not highly distorted. It then selects the only middle frame and extracts the character from it. This prevents the character from selecting a frame where the character has parts that are not completely scaled. However, if the number of frames is more than three, our attack excludes frames that are outside a range of between 1/3 and 2/3 of the mainly selected frames. After that, we use a method that finds the lowest distorted image of the character. This method calculates the difference between the height and width of the character in each frame. After that, it uses the results to find the frame where the ratio of the height of the character over its width is the greatest between the selected frames. The reason behind this method is the horizontal deformation distortion method that moves the parts of the characters horizontally. This distortion method usually increases the ratio of the width of the character over the height. We can explain the algorithm used in this step as follows:

Algorithm: correct the order of the characters
Input: $AImage$ (Animated image), X_S, X_E (Coordinations of the animated character)
Output: $FImg$(chosen image of character)

1: $M \leftarrow$ Coordination of the bottom of the image over y–axis
2: $Arr \leftarrow$ new $Array$ of $images$
3: $FImg \leftarrow new\ image(\ X_E - \ X_S, height$ of $AImage)$
4: **for each** $frame$ **in** $AImage$ **do**
5: $CImg \leftarrow frame(X_S \rightarrow \ X_E, 0 \rightarrow height$ of $AImage)$
6: $E \leftarrow \ highest\ vertical\ position$ of $character$ in $CImg$ over y–axis
7: **if** $E > M$ **then**
8: remove all $components$ in Arr
9: $M = E$
10: Add $CImg$ to Arr
11: **else if** $E = M$ **then**
12: Add $CImg$ to Arr
13: **end if**
14: **end**
15: $L \leftarrow length$ of Arr
16: $R \leftarrow 0.0$
17: **for** $C \leftarrow L/3$ **to** $2L/3$ **do**

18: $S \leftarrow height$ of $character$ in $CImg[C]/width$ of $character$ in $CImg[C]$
19: **if** $S > R$ **then**
20: $R = S$
21: $FImg = CImg[C]$
22: **end if**
23: **end**
24: **return** $FImg$

Figure 5.5 shows an example of the character extraction process using a set of images cached from marked vertical column through the PDM segmentation, which contain the character "N". Firstly, we have run the analysis through our attack to find pictures of the character that reach the highest position as figure 5.5 (A) shows. Afterwards, our attack has selected all the images of the character "N" that reach the highest position. It has then selected the two frames between the 1/3 and 2/3 of the mainly selected frames as figure 5.5 (B) shows. Finally, it selected the frame where the ratio of the height of the character to its width is the greatest between the selected frame, before extracting the character from it as shown in figure 5.5 (C).

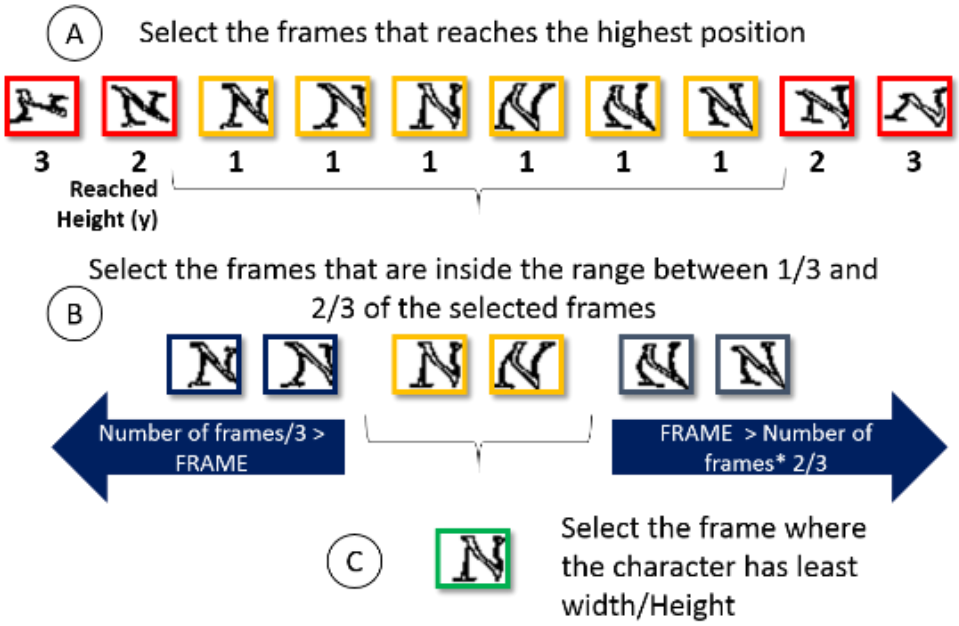


Figure 5.5. The character extraction process

5.3.5 *Recognition attack*

In this step, we used the Convolutional Neural Network (CNN)[110] engine to recognise the extracted characters as in our attacks in previous chapters. We trained this engine to recognise characters inside the CAPTCHA images by creating a training set of 1645 sample

images of characters (47 sample images for each character) and testing sets of 1465 sample images of characters. After that, we used two datasets to train our engine 24 times and the result was the following:

- The engine achieved a 0.9% error rate in training after it trained 1631 sample images of characters from the training dataset successfully.
- It achieved a 0.1% error rate in testing after it recognised 1463 sample images of characters from the testing dataset successfully.

Figure 5.6 shows how the training and testing accuracy rates were improved during the training task.

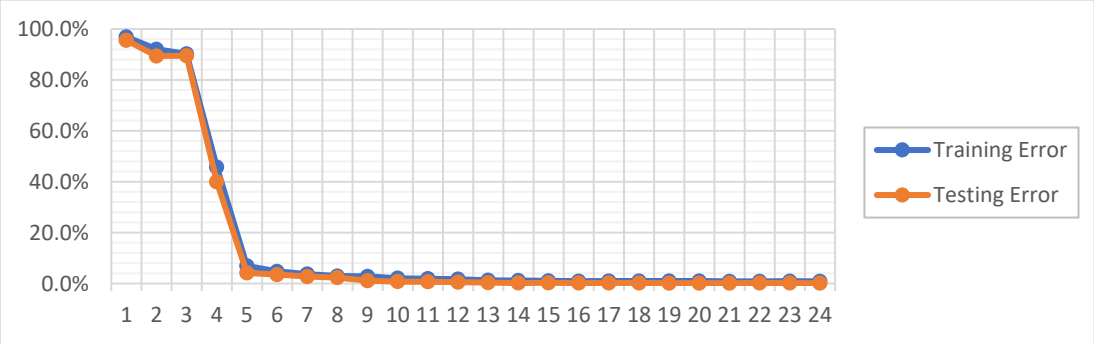


Figure 5.6. Training and testing accuracy rate during the training of the CAPTCHA

5.4 Evaluation

We discuss the results of our attack in terms of segmentation and recognition using the same method included in the previous chapters. Our evaluation in this chapter was based on an examination of 1100 random samples of this CAPTCHA from the CAPTCHANIM website[85]. These samples belong to the designs that uses the highest level of character distortion through horizontal deformation methodology. We obtained the samples between January and February 2015. We used 100 of the samples in designing, training and processing our attack. We saved the other 1000 samples locally to be used in the final examination of our attacks. We did not include any sample used during the design of the attack in the evaluation process of our attack.

5.5 Results

Success rate. We firstly tested our attack against a sample set of 100 of the CAPTCHA challenges used for training purposes, and it achieved a success rate of 98% in recognising them, after it achieved a 100% success rate in segmenting them. After that, we tested our attack on 1000 random test samples of the CAPTCHANIM, and it achieved a success rate of 85.1% (851 out of 1000) in recognising them, after it also achieved a 100% success rate in segmenting them. Those results also include a success rate of 92% against 200 samples of the first category (184 out of 200) and a success rate of 83.375% against 800 samples from the second category (667 out of 800). The results also showed that the use of additional computational effort does not increase the success rate of our attack on this CAPTCHA scheme.

Attack Speed. Our CAPTCHA attack was implemented using JAVA, and C# programming languages, and tested on a computer with an Intel(R) Core(TM) i7-4700MQ CPU 2.40GHz and 8.00 GB ram. The results show that the attack took an average of 64.41 milliseconds to analyse each animated challenge taken from the CAPTCHANIM with standard deviation of 4.689497989 milliseconds. The shortest attacking time was 54 milliseconds, while the longest was 78 milliseconds

5.6 Other attacks

Vu Duc Nguyen, et al.[91] developed a more basic attack against CAPTCHANIM. This attack uses a method that swaps the images of the second category. After that, it processes the CL method to segment the text and extract its characters in their complete images. This method uses the PDM map to determine the vertical area for the character pixels in the animated frames. It then uses the results to segment the characters vertically. Afterwards, it scans each vertical area within the PDM map to determine the maximum height reached by each character. Finally, it extracts this character from any frame where its height equals the maximum height.

Unlike our attack, this attack was designed to exploit the vertical scaling method. Therefore, the designers did not include any method to exploit the horizontal deformation method. The attack achieved a 21% success rate against mixed samples of the first and second categories of the CAPTCHANIM with an average time of 1.9 seconds. The

attackers did not mention if any of those challenges used horizontal deformation methodology.

5.7 Defence

We implemented this attack against the animated text-based CAPTCHA scheme. As we mentioned before, this scheme shows each of the animated characters in one fixed location all the time. It also displays each of the characters completely for a length of time through the animation frames. The high success rate of our attack helped us to find vulnerability problems in this CAPTCHA. We could avoid those problems by preventing any character from being displayed completely in any animated frames. The character should also not remain in one fixed location within the string itself. In addition, we would advise to connect the string and move it completely in random directions to resist the PDM segmentation method.

5.8 Summary

Our attack has shown the capability of the automated attacks to break CAPTCHANIM with an overall success rate of 85%. Our attack has also demonstrated how automated attacks can bypass the rotation effect that was regarded as the most difficult effect to break in the previous attacks. As a result, we have proved that the computer can pass all the procedures used in this animated text-based CAPTCHA scheme more efficiently than the human brain. Therefore, a new mechanism that uses a strong cracking algorithm needs to be designed and used to create a new animated text-based CAPTCHA scheme that could provide better security against automated attacks. This mechanism should not display the character clearly in any frame as computers can easily determine and extract the character from it, as we have proved from our study.

Chapter 6. ReCAPTCHA

The ReCAPTCHA is widely known as the most resistant CAPTCHA scheme to attacks. This CAPTCHA contains two segment resistance words that contain many characters that have two different levels of distortion and segmentation resistance. We attacked this CAPTCHA by using a combination of automated segmentor techniques that identifies characters through heat and other types of segmentation attacks. This attack achieved a 43% success rate in recognising the challenges of this CAPTCHA.

6.1 Introduction

This chapter presents our first examination against a segmentation-resistance mechanism, which is the "Crowding Characters Together" (CCT) mechanism implemented by ReCAPTCHA. Our attack will target the latest version of this scheme that combines the complex distortion system and the segmentation resistance mechanisms. This CAPTCHA was originally developed by Luis von Ahn et al. in Carnegie Mellon University in 2000[112]. Since then, this standard text-based CAPTCHA scheme has become one of the most popular schemes, as many major websites have deployed it in their pages. These websites include Facebook, Twitter, CNN, StumbleUpon, and many other internet services. This scheme was acquired by Google in September 2009.

ReCAPTCHA is distinguishable from all others by its functionality including the ones we targeted previously including the Wikipedia CAPTCHA, CAPTCHANIM, and the KillBot CAPTCHA. It employs two text strings; one string is known to servers as the main string of the CAPTCHA challenge, while the second is used only for labelling functionality. Both strings were selected from a list of words copied from a list of digital books and journalists. The earliest versions of the ReCAPTCHA scheme used to display both strings using the same distortion and anti-segmentation mechanisms. The recent versions of ReCaptcha used two different combination of distortion and anti-segmentation mechanisms. The most difficult combination that includes the CCT mechanism is only used against the main text, and the easy level of distortion is used in the second string. The main purpose of these two levels is help the human to distinguish between the text images in them.

The CCT segmentation mechanism made by the ReCAPTCHA scheme is one of the reasons for its resistance against attacks. This mechanism relies on overlapping the characters with each other at a random transition. The main purpose of this mechanism is

to prevent attacks from identifying the position of the characters and the connection points between them. Combining this mechanism with distortion mechanisms plays also an essential role in improving the resistance of the CAPTCHA against attacks. Such a combination could help to make the characters much less distinguishable by those attacks.

In this chapter, we are going to target the most recent version of this CAPTCHA scheme, released in 2013. This version combines CCT a segmentation-resistance mechanism with multiple distortion methods. These methods include noises, partial fading, hiding, affine transformations, text italicisation and text warps. We could easily differentiate between the two strings in each challenge by their distortion, as the main string combines more distortion methods than the other. Another difference between the two strings is that the other string connects fewer characters than the main string in each challenge.

Our goal is to break this CAPTCHA through a new type of attack that uses an automatic-segmentor mechanism based on an attack by Chellapilla [45]. This mechanism uses the recognition engine to create a heat map to identify the characters within the text with their order. Our attack also combines this mechanism with other segmentation methods used in novel attacks. In addition, it includes various methods to correct errors in recognition within the heating map.

The argument outlined in this chapter achieved our goal, as we exploited the pattern of the distorted characters in this CAPTCHA through our attack. The key insight of this attack is to show that the computer could also segment the distorted characters through recognition attacks that create the heat map of the characters. Our attack achieved a 43% success rate in recognising the challenges of this CAPTCHA, which is the highest success rate recorded against this CAPTCHA in 2014. It is also more than the 1% success rate needed to consider the CAPTCHA broken[59]. The average time taken to break the CAPTCHA was 4.8755 seconds, which exceed the average time needed by the human to solve this scheme.

To the best of our knowledge, the attack presented in this chapter is one of the most critical analyses, as it contributed towards answering the challenging questions regarding the robustness of the CCT mechanism. This contribution is vital to our research against the current animated text-based CAPTCHA schemes, since many CAPTCHAs already included this mechanism in their challenges, such as the NuCAPTCHA.

The sections of this paper are organised as follows: Section 6.2 provides an overview of the targeted CAPTCHA scheme. Section 6.3 describes our attack followed by our evaluation in section 6.4 and results in section 6.5. We discuss other attacks against the ReCAPTCHA that include attacks targeted at earlier versions of this CAPTCHA in section 6.6. Section 6.7 discusses the defence against our attack, followed by a summary in section 6.8.

6.2 The characteristics of ReCAPTCHA

ReCAPTCHA is implemented using a segmentation-resistance mechanism that crowds a number of highly distorted characters together. The key characteristics of the current version of this CAPTCHA are the following:

- Each challenge uses a grayscale image.
- Each challenge employs two text strings that are utilised as a crowd sourcing system for digitising books.
- One of the two strings are known to the server and functions as the CAPTCHA string.
- The other string is unknown and used only for labelling functionality.
- The CAPTCHA main text string known to the server contains characters that are very distorted and crowded with each other. These few connected characters could be above or below the line level while connected horizontally with other characters beside them.
- The characters in the main text string are distorted through a mechanism that relies on the noises, affine transformations, word warps, character rotation, partial character fading and removal mechanism. This mechanism also relies on another distortion feature that includes different levels of text italicisation within the string in which it slants different parts of the string into different angles either to the left or to the right.

- The unknown string used for labelling functionality contains characters that could be faded partially and blurred in addition to some noises around them to connect those characters together.

- This unknown string can appear in different fonts and sizes unlike the CAPTCHA string.

Figure 6.1 shows an example of this CAPTCHA, where the first string is the CAPTCHA string that is known to the server while the second string is the unknown one used only for labelling functionality.

Figure 6.1. An example of ReCAPTCHA

6.3 Attacking ReCAPTCHA

Our attack works through a technique that includes the following two steps:

- Segmentation attack: in this step, the attack identifies the objects related to the CAPTCHA string and the unknown string. It then segments those objects into two strings and identifies the CAPTCHA string. This step also includes another method to correcting the italicisation of the CAPTCHA string, and segments through connection loops.

- Recognition attack: the attack will use an automatic segmentor technology to create two maps of the text. The first map shows the text without segmentation while the second shows a segmented text through connection loops. The attack then compares the two results to get the right result, before it identifies errors in recognitions.

6.3.1 *Segmentation attack*

This attack includes two main steps:

- The first step identifies the CAPTCHA string using the colour filling attack[51].

- The second step corrects the alignment of the CAPTCHA string using the houghlines (Hough transformation) algorithm[113].

Identifying the CAPTCHA string. The attack first uses the colour filling attack to find and mark every black object in a CAPTCHA image using one unique colour. Then, it determines the string related to each of the objects through a method that links every two nearest objects together, until it creates the two strings.

After that, the attack determines which of the two strings is the CAPTCHA string through a method that selects the string that has the lowest ratio of components that appear in different vertical columns to the width. This method locates the object within every string through its colour. It then combines the segmented objects that have more than half their parts gathered in the same vertical column. It determines the number of separated objects within each string by counting the number of colours in it.

It then determines the ratio of string components to the width for every string using the algorithm $f(S) = S_{colours}/S_{width}$, where $S_{colours}$ is the number of colours used in the string, and S_{width} is the width of the string. Finally, it chooses the string that achieves the lowest result through the algorithm as the CAPTCHA string.

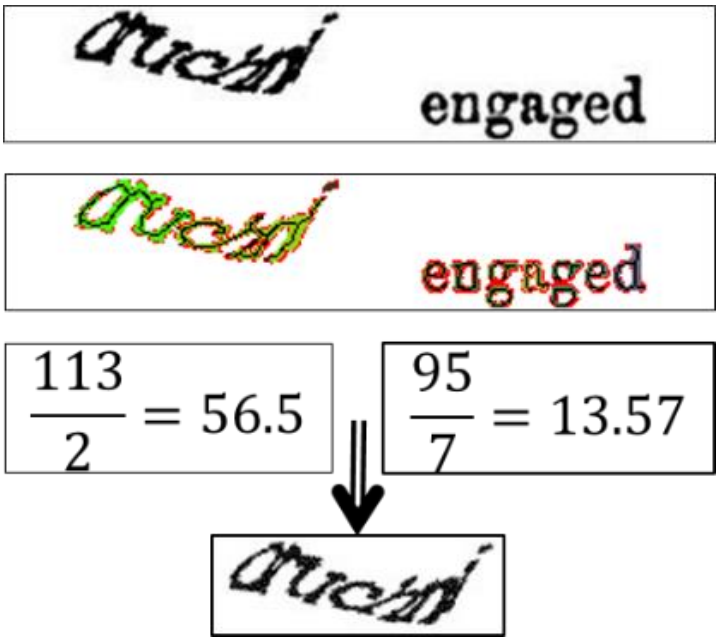


Figure 6.2. The identification of the CAPTCHA string

Figure 6.2 shows an example of the identification of the CAPTCHA string. The first string in the image has a width of 133 pixels and is filled with two colours which show that it has two components separated vertically. However, the second string in the image has a width of 95 pixels and is filled with seven colours that show it has two components separated vertically. The attack identified the ratio of the number of components that appear in different vertical columns over the width of the two strings. The first string achieved a ratio of 5650%, while the second string achieved a ratio of 1357%. Therefore, the attack identified the second string as the CAPTCHA string.

Correcting the alignment of the CAPTCHA string. In this step, our attack corrects the alignment and orientation of the main text using an method inspired by an attack developed by Cruz-Perez et al.[58]. This method first generates a morphology image that contains a thinned area for each of the identified components in the CAPTCHA image. It does so by using skeleton morphological image processing that thins the objects in the images and creates thinned connected lines that are one pixel thick at the centre of each object.

The method detects and selects all vertical lines inside the morphology image such as the two vertical lines on the sides of "H" using the houghlines algorithm[113]. It then identifies the orientation degree for every identified vertical line. After that, it calculates the average of alignment angles for all the vertical lines to determine the alignment angle for the string. This method then identifies the string with an alignment angle that is not close to 90 degrees and corrects its alignment angle to 90 degrees.

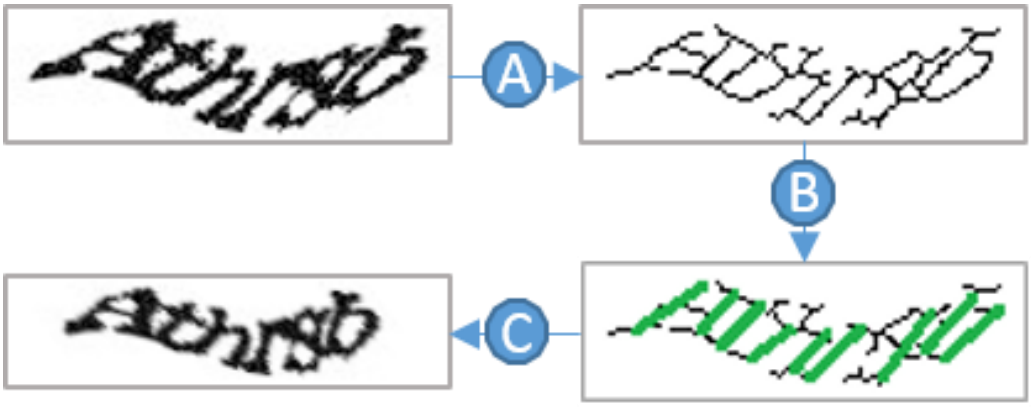


Figure 6.3. Correct the italicisation of the CAPTCHA string

Figure 6.3 shows an example of this method, where it thins the image in the string and converts it into morphology an image as shown in figure 6.3 (A). The method scanned this

image to identify the vertical lines and marked them using the green colour, as shown in figure 6.3 (B), and it then calculated the rotation angle of each line and the average rotation. After that, it found that the string was aligned to the right, and then it corrected the aligning of the string, as shown in figure 6.3 (C).

6.3.2 *Recognition attack*

In this step, we analyse the main string using an automatic segmentor inspired by an attack by Chellapilla [21]. This automatic segmentor uses the convolutional neural networks [110] to create two maps that label the characters in their correct positions in the CAPTCHA image. The first map shows the whole string without segmentation. The second map shows divided parts of the string segmented by a method that detects the connection loops and segments the areas between them. Both maps are used to create another map that shows the middle result between them. This map will be subsequently analysed through various methods to correct every recognition error in it. In this section, we will discuss the methods used in the attack through numerous sections that represent the steps of the attack. These steps are:

- Catching the screenshots for the heating maps: this step includes methods that generate screenshots of every vertical column within the image of main text and sends them to recognition engine to generate the heating maps.

- Generating the heating maps: this step includes a method to create a number of heat maps that identify the areas of the CAPTCHA characters.

- Creating the automated segmentation maps: the attack in this step uses the heat maps of the characters to create two major maps that show the positions of the characters in the image, which are used to create another map that shows the result between them.

- Correct errors in map: this step includes additional methods to correct common errors in the map.

Catching the screenshots for the heating maps. In this step, we generate screenshots for every horizontal column within the main text. We then send those screenshots to the recognition engine to create the heat map for every typographic symbol (character) included in the character set of the ReCAPTCHA. This map shows the areas where the

character is detected in the CAPTCHA image. The way to catch the screenshots is through a set of sliding rectangles of the area that appears completely inside them. This set includes a number of rectangles with different widths that match the widths of the character symbols included in the main string. The width of the thinnest rectangle included in the set is 4 pixels, which matches the thinnest width of the character "i" found in the training samples of ReCAPTCHA, while the width of the widest rectangle is 40 pixels, which matches the widest width found for the characters "W" and "M" in those samples. Each rectangle moves horizontally from the left side to the right side of the string as while it changes height each time it moves to a new area to match the height of the black component in that area.

We use the sliding rectangles to create two different sets of screenshots of the same text. The first set includes screenshots of the whole image of the text string. We use these screenshots to generate a map that shows the recognition results of the whole string without segmentation. The second set includes screenshots of different segmented parts of the same text. We use these screenshots to create the second map of the text that shows segmented parts of the string by a loop segmentation method. This segmentation method detects the connection loops and segments the areas between them using a map that shows the loop components in the text. It also includes other steps that scan the loop components within the map to locate the connection loops and segments the text through them.

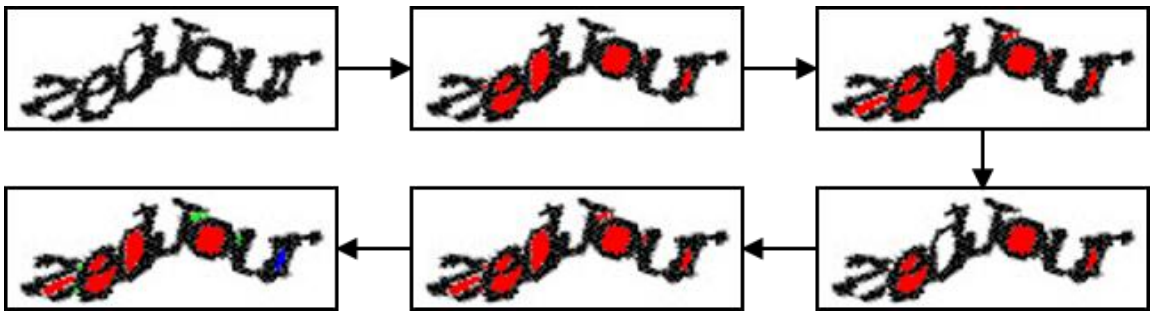


Figure 6.4. The loop detection

Our method creates the map of connection loops with a colour filling algorithm to fill all the loop components in the text string. These components include those filled in different grayscale values. This method locates these components by dividing the image of each string into different layers. Each layer contains a different image of the string that is filled with white until it reaches a different grayscale pixel value that keeps increasing through the layers. It then analyses each layer of the image to find the loop components in it. The

method then adds all the components to a map that shows the results of previous layers where the highest grayscale pixel value filled with a white colour is lower than the current one. Figure 6.4 shows an example of this detection approach.

Our method creates the final map that shows all the loop components found from the analysis of the image layers through the detection approach. It then scans the loop components in this map to find each small or thin loop component and marks it as a connection loop. The method then uses a vertical segmentation algorithm that draws a vertical line between the middle of the upper area of the connection loop to the middle of the lower area of each marked loop component. Afterwards, it uses the sliding rectangles against each segmented part of the text besides the vertical lines instead of the whole text. Figure 6.5 shows an example of this segmentation method.

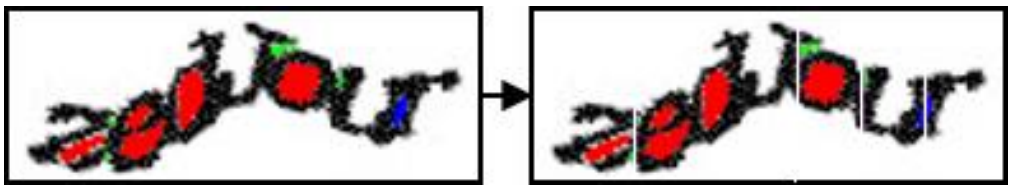


Figure 6.5. Loop segmentation

Finally, the attack sends a screenshot image generated in this step to an automatic segmentor to generate the heating maps in the next step. We also include associated data for each screenshot that shows the coordination of the area of the image, its width and the ratio of the height to width.

Generating the heating maps. In this step, we analyse each of the images with a recognition engine to identify the confidence rates for the CAPTCHA characters, which show the accuracy rate of the image that displays a character.

The screenshots used to generate the confidence rates for each character are limited to a range of screenshots with widths. These widths are limited to a range between two widths specified for the analysed character. The attack uses these confidence rates to generate the heat map that shows the average confidence rate for the character. The method used to generate this map calculates an average of the confidence rate for each pixel from the images that display it in the CAPTCHA image for each character. This calculation is through the following algorithm:

$$P_{(x,y)}(\delta) = \frac{\left(\sum_{i=0}^{a} 10\, A_{(x,y)}[i]\right) + \left(\sum_{i=0}^{b} B_{(x,y)}[i]\right)}{10\, a \;+\; b}$$

Where A includes the set of confidence rates of the character δ recorded from the screenshot in which the character δ achieved the highest confidence rate between all the characters in the coordinated point (x, y), and B includes the set of confidence rates recorded from other screenshots in that point.

The attack then gathers the average confidence rate for characters generated through the pixels in the entire area of the scanned string in two maps for each of those characters. The first map shows the results from the screenshots taken through the first method that scans the whole area of the string. The second map shows the results generated from screenshots taken by the second method that includes a segmentation approach. Figure 6.6 shows an example of the heating maps created in this step.

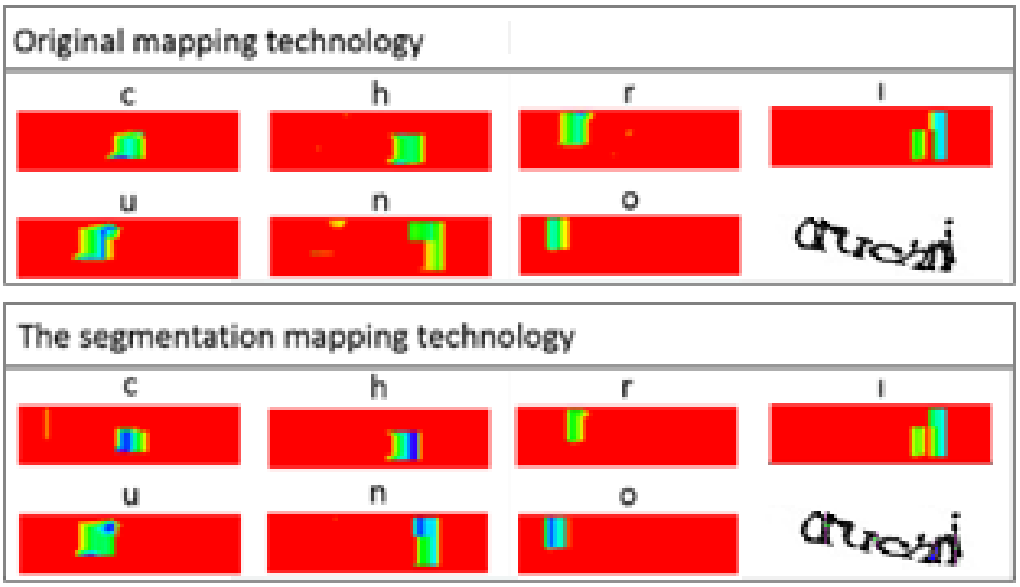


Figure 6.6. The characters heating maps

Generating the major automated segmentor maps. After the attack creates the heat maps for all the characters, it uses them to create two major maps that show the position of the characters from the maps generated in the previous steps. The first map shows the character with the highest average confidence rate in each pixel within the area of the string. However, the second map excludes pixels from the vertical columns that contain the connected loops. The main reason for this exclusion is the false recognition results caused

by the changes in the pattern of the text by the loop segmentation method. Figure 6.7 shows an example of the two maps created from the heating maps in this step.

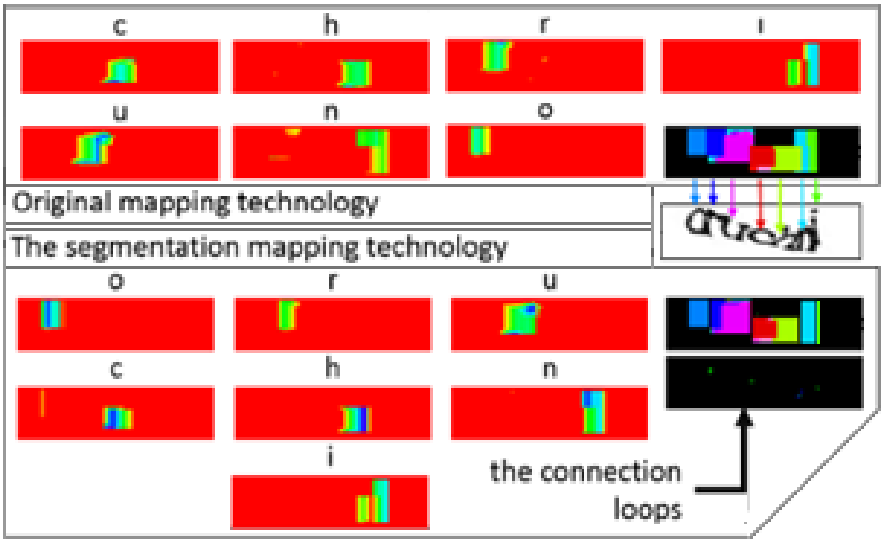


Figure 6.7. The two major maps

After that, the attack uses both maps to create another map that merges the results of those maps. This map can show the results without any recognitions errors from one of the two maps using the results generated from the other map. The attack then processes a method that uses the original image of the text to create a final map that shows the result from the latest map only, inside the pixel point of the main text. This method replaces the black pixels in the original text image with the pixels of the latest map, while it keeps the pixels of the white background in that image unchanged. The main purpose of this method is to remove the recognition errors in areas that do not contain any character within the image of the main text. Figure 6.7 shows how we created the final map

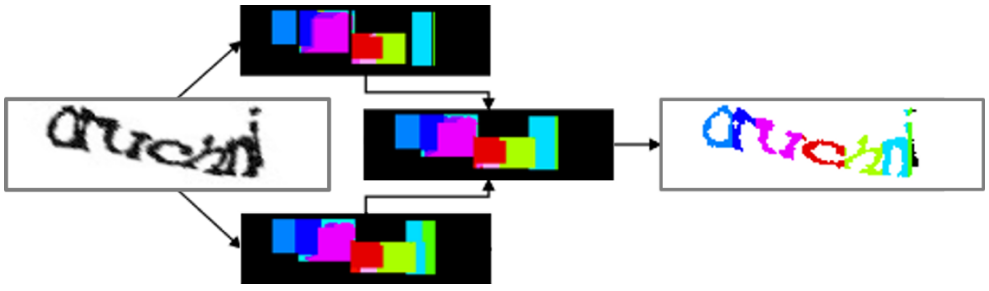


Figure 6.8. Creating the middle results map in addition to the final map

Maintains the map and corrects errors. The attack in this step uses numerous methods that maintains the map and corrects the errors shown in the animated image due to common

recognition problems. It also includes other recognition checkers that use the unique patterns of characters to determine if the recognition results of characters are correct. These methods are as follows:

The checker of characters with dotted points. The attack firstly uses a method that corrects errors caused by identifying any area containing lines from a character and small parts of other characters above it as "i". This error is caused because of extracted screenshots of a character that display lines and a small area above it in an image that makes them resemble the body of the character "i". This method scans every area identified as "i" to determine if it contains a vertical line and a segmented or rounded point and checks if the line and point are not parts of the other character. After that, it marks every area that does not pass the two conditions as wrongly recognised as "i". It then re-colours the area using the second highest result after it in the heating map. Figure 6.9 shows the recognition map before and after we implemented the checker in which it removed two areas wrongly recognised as "i".

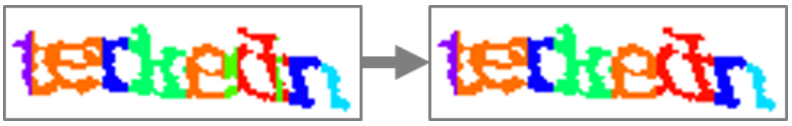


Figure 6.9. The removal of wrongly recognised areas as "i": the wrongly recognised areas as "i" are marked in green in the first image

The removal of errors in recognising a "side of character". This method removes the common errors in our final map that comes from recognising a side of a character as another character. These types of error usually occurred due to cases in which one side of a character resembled another character. These errors included:

- Recognising the left side of the characters "n" and "m" as character "r".
- Recognising the left side of characters "b", "h", "k" and "p" and the right side of characters "d" and "q" as character "l".
- Recognising any of the two sides of the character "w" as character "v".
- Recognising any of the two sides of the character "m" as character "n".

We resolve this problem by running an analyser that works as follows:

- It scans results within the string until it finds an error in which a component δ_a recognised as A and resembles a specific side of B is located in that specific side beside a component δ_b recognised as B.

- It then uses the heating map of the character B to identify and selects pixel points within the component δ_a where the confidence rate of the character B is above 50%, and the second highest confidence rate between the recognised characters. After that, it excludes a group of selected pixel points located within one x-coordinated vertical line from the group of pixel points that could be part of δ_b. It marks the rest of the selected group of pixel points from the component δ_a as θ_a and the group of the unselected group of pixel points from that component as ρ_a.

- It then determines the width of the components δ_b, θ_a and ρ_a. It also calculates the width of the two joined components δ_b and θ_a with the following equation:

$$Width(\delta_b, \theta_a) = Width\ of\ (\delta_b) + Width\ of\ (\theta_a)$$

  As well, it calculates the ratio of height over the width of those joined components in which through the following equation:

$$Ratio(\delta_b, \theta_a) = (Y_H(\delta_b, \theta_a) - Y_L(\delta_b, \theta_a) + 1)/Width(\delta_b, \theta_a)$$

  In which the $Y_H(\delta_b, \theta_a)$ is the highest coordination over y-axis for δ_b and θ_a, and the $Y_L(\delta_b, \theta_a)$ is lowest coordination over y-axis for δ_b and θ_a.

- After that, it uses an algorithm that compares the results of the calculation with the minimum and maximum possible width and ratios of height over the width of the characters A and B in the ReCAPTCHA scheme. We process this comparison using a database that includes details that show the range between the minimum and maximum possible widths and ratios of height for every character included in the character set of that scheme. We use the results of the comparison to determine the result of the comparisons match with the following conditions:

    - The width of the component ρ_a is either "less than 5 pixels and less than 50% of the minimum width determine for the character A in the database"

or “more than the minimum width determine for the character A in the database”.

- The width of the component δ_b is less than the minimum width determined for the character B in the database.
- The width of the joined components δ_b and θ_a $\left(Width(\delta_b, \theta_a)\right)$ is within the minimum and maximum width of character B.
- The ratio of height over width of the joined components δ_b and θ_a $\left(Ratio(\delta_b, \theta_a)\right)$ is within range between minimum and maximum ratio of the character B.

- If the results matched those conditions, it merges the component θ_a with the component δ_b by filling the area of the component θ_a with ID colour of character B.

Figure 6.10 shows an example of this method in which it corrected an error in recognising a side of character “n” (cyan colour) wrongly as “r” (blue colour) leaving a small area recognised as character “r” to be removed and corrected in the next step.

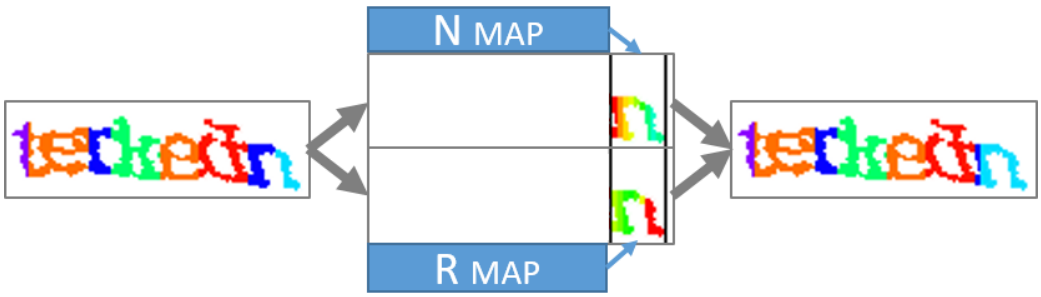


Figure 6.10. Correcting an error in recognising the side of character “n” as “r”

The removal of the thin area. This method corrects a common problem in the heating in which thin vertical component with widths of less than five pixels can be falsely recognised as one character. The method identifies these areas through an algorithm that calculates the width of every vertical column identified as one character within the text string. After that, it selects every vertical column that has a width of less than five pixels and identifies the character recognised in the vertical column through the colour ID used to fill it in the main map. It then compares the width of the vertical column with the determined minimum width of the character symbol using the same database included in the removal of errors in

recognising the "side of character". After that, it adds it to the list of thin vertical components that contain wrongly recognised characters in case the determined width is less than the smallest possible width of the recognised character. Finally, it marks together each group of connected vertical components that are in the list of the wrongly recognised thin components. After that, it re-colours those marked components using the colour ID of the characters that achieved the highest heating value between the characters recognised within the vertical components of the main text that are not in the list besides the marked group. Figure 6.11 shows an example of how the method removed a thin area recognised as "r" (blue colour) between two areas recognised as "n" (cyan colour) and "d" (red colour).

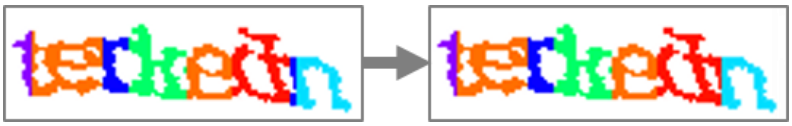


Figure 6.11. The removal of a thin area recognised as "r" between "n" and "d"

Detecting the pairs of characters. This method locates and segments every connected pair of one character in the map, as the automatic segmentor always identifies these pairs as one character by colouring their area with the same colour ID. The method locates these pairs by determining the width of each area coloured by the colour ID in the text and checking if it wider than the maximum width of one character. It then selects each area wider than the maximum width of one character and segments it in the middle. Figure 6.12 shows an example of how the method detected two pairs of the character "r" beside each other, and then segmented them in the middle.

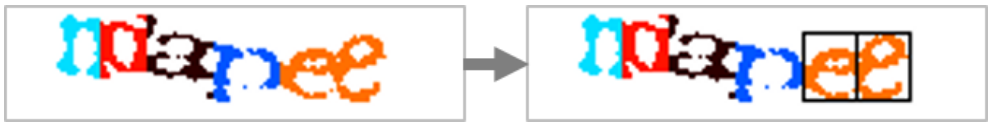


Figure 6.12. Detecting and segmenting pairs of characters recognized as "e"

6.4 Evaluation

We discuss the results of our attack in terms of segmentation and recognition using the same method included in the previous chapters. Our evaluation in this chapter was based on an examination of 200 random samples of this CAPTCHA. We obtained those samples through the ReCaptcha website[114] between March and April 2014. We did not download any further samples due to a limitation on the maximum number of samples that can be

downloaded from the website for each IP address. We used 100 of the samples in the designing and training process in our attack and saved the remaining 100 samples locally to be used in the final examination of our attacks. We did not include any sample used during the design of the attack in the evaluation process of our attack.

6.5 Results

Success rate. We have tested the automatic segmentor that generated the middle results between both maps from the original technique and the map from the segmentation technique on 100 training samples from the ReCAPTCHA. It achieved an overall accuracy rate of 49% (49 out of 100) in recognising them. We then tested the CAPTCHA against another 100 test samples and it achieved an overall accuracy rate of 43% (43 out of 100) in recognising them. The results also showed that the use of additional computational effort does not increase the success rate of our attack on this CAPTCHA scheme.

We also detected some failures in the attack that can be represented in the following way:

Failure 1: Errors in identifying and segmenting the connected loop components. This failure is caused by the distortion methods that change the patterns of loop components into the connection loop. In addition, they can also cause the opposite problem in which the segmentation method results in one character being segmented into two characters. Most errors arising from those two issues can be normally solved through the map of the characters, if the segmentation does not cause the segmented area to resemble any other character. Figure 6.13 shows examples of the two issues explained in the attack. The first example shows that the loop component of the character “e” caused it to be cut it in half, while the second example shows another error in the connection loop between “e” and “b” which caused the character “b” to be cut in half.

(A) **(B)**

Figure 6.13. The loop segmentation errors:
(A) In identifying the components and (B) in segmenting the components

Failure 2: Major issues with correction of the italicisation in the CAPTCHA string. One problem we faced in our attack was the italicisation of the CAPTCHA strings. These problems resulted from an error in calculating the italicisation angle of some strings due to the lack of lines that could be used to calculate the angle. The percentages of strings where this issue occurred was only 3%. Figure 6.14 shows an example of strings where the italicisation angle could not be corrected due to the lack of lines that could be used to calculate this angle in our attack.

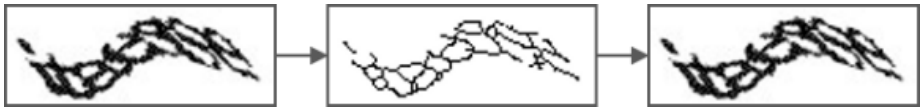


Figure 6.14. Error in correcting the italicisation of the string

Failure 3: Inaccuracy in recognising the full body of the characters in the map. This problem usually arose in the type of CAPTCHA with highly connected and warped letters. The main reason for this problem comes from the way these characters are distorted and connected, which leaves only a small space between them. Thus, the generated result inside the characters can always be affected by the recognition result of the characters beside them. This problem can only cause problems in segmenting the CAPTCHA using the current method. However, it can have a lesser impact on the recognition result through the identification of the colour ids of the characters on the map.

Figure 6.15 shows an example of this problem, where a huge part of the third character ("t") was included as a part of the second character ("r"). This causes the segmented area of "r" to be displayed as "n", while the segmented area of "t" is displayed as an unknown character. It also shows a similar problem in which the right side of the character "o" is identified as part of character "T". Such a problem could cause the segmented area of the character "o" to be displayed as "c".

Figure 6.15. Example of the inaccuracy in recognising the full body of the characters

Failure 4: Errors in identifying the connected areas between characters that resemble different characters. The most common problem in our attack is the identification of the

connected characters that resemble different characters. This difficulty normally occurred when two highly connected pair of distorted characters resampled another character. The huge distortion used against the characters could cause the attack to have difficulty in identifying those characters and distinguish them from the other character they have resembled. Figure 6.16 shows an example of this problem where the third character “r” and the left side of the character “t” is identified as character “n”.

Figure 6.16. Example of the connected characters defined as different characters

Figure 7.11 shows rate of failure in our attack against the ReCAPTCHA.

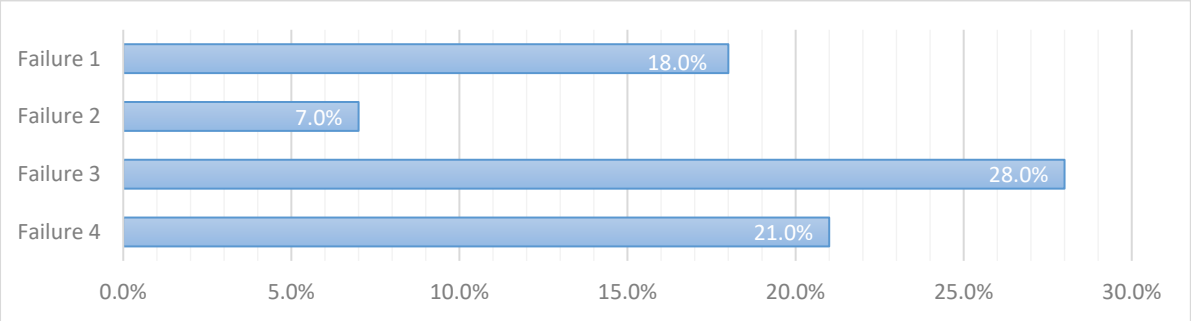


Figure 6.17. Failures rate in our attack against the ReCAPTCHA

Attack speed. Our CAPTCHA attack was developed using JAVA, C# and MATLAB programming languages, and tested on a computer with Intel(R) Core(TM) i7-4700MQ CPU 2.40GHz and 8.00 GB ram. We tested our attack only against the samples of the test set. The result shows that the attack took an average of 4875.51021 milliseconds to analyse each animated challenge of the ReCAPTCHA with standard divination of 1351.230094 milliseconds. The lowest attacking time was 2622 milliseconds, while the highest attacking time was 9669 milliseconds.

6.6 Defence

One point that could be improved to make the CAPTCHA stronger is the warping, which includes the local warping. Such an improvement could make the connected characters more difficult to segment through the vertical line or extract without extracting any part of the other characters with it. The distortions used in the attack could also be helpful, as they

can create too many versions of one character and make them difficult to analyse or segment through the automated attack.

On the other hand, the CAPTCHA needs to solve another weakness to make it more secure. One major weakness comes from using one font type and size in all CAPTCHA images, which makes the characters in it easy to identify through the recognition attacks. In addition, it needs to make it harder for the attack to distinguish between the two strings included in the CAPTCHA, as the current two strings are very easily distinguished through the attacks.

6.7 Other attacks

There are numerous attacks presented against the earlier versions of the ReCAPTCHA before the one we targeted in our attack. The first attack was presented by Jonathan Wikin [115] as a part of his robustness investigation against the first version of the ReCAPTCHA , which displays the text as characters that are lined up on one level line with only a few characters connected together through their edges. This attack uses a standard image manipulation technique to pre-process the CAPTCHA image. It then sends it to an OCR-program multiple times to recognise it, before it chooses the answer returned repeatedly by the OCR as the right answer for the challenge. This attack achieved a success rate of 17.5% against the samples of the CAPTCHA scheme.

Chad Houck[57] presented another attack in 2010 that achieved a success rate of between 10% and 31% against two early versions of ReCAPTCHA. Those versions connect most of the characters together through their edges and are distorted through a low level of text wrap. The attack used against this CAPTCHA firstly analyses the words distorted through a wave distortion mechanism in the image with a “blanked algorithm”. This algorithm uses the upper and lower contours of a distorted string to estimate its waving parameters through a series of tangents points above and below that string. After that, it uses the analysis to remove the wave distortion from the string. It then divides the string into a series of vertical chunks using an algorithm that searches for the valleys in the upper contours of the string and for peaks in the lower contours. Afterwards, it compares content located between two chunks which is compared with a series of templates that shows the average features of the characters. Then, it draws a vertical line between each valley and peak located in the same x-axis of the string.

Ahmad Al-Ahmed[52] also targeted another version of the ReCAPTCHA that further distorts the character by adding noises around it. His attack is an enhanced version of an attack we developed against the Google CAPTCHA that achieved a success rate of 33% against it. This attack included a pattern detection algorithm that located the individual characters using their shape and their connections. The characters detected under the algorithm were:

- Characters with dots like “i” and “j”.
- Characters containing loop components like “o”, “b”, “0”, “6” and “8”.
- characters that contains three vertically juxtaposed lines such as “S”, “3” and “E”.

Each character is segmented using a vertical segmentation method that aims to find the side of the character based of its shape. The attack segments the rest of the characters through a method that identifies their width to detect the components with a width higher than “w”. The attack analysed those components containing a series of the large pixel counted components with small chunks between them, before segmenting them.

Claudia Cruz-Perez et al.[58] also created another attack against a later version of ReCAPTCHA that adds a simple italicisation method to slant the whole text into one specified angle to the left or the right. This attack includes a step that uses a morphological image to identify a variety of CAPTCHA characters. Those characters are the following:

- characters with circular regions like “a”, “b”, “e”, “g”, “o”, “p” and “q”.
- characters with an occurrence of more than one pixel per column such as “c”, “f”, “k”, “s”, “t” and “z”.
- characters with u-shape, n-shape and r-shape patterns like “u”, “n”, “h” and “r”.
- characters of one pixel per column with slopes like “v”, “x” and “y”, thin characters such as “i”, “j” and “l”.
- double shaped characters like “m” and “w”.

It detects the unidentifiable characters in the previous step through two different methods. The first method adopts a Three-Colour Bar approach, which analyses many circular patterns to identify items that include them. The second method uses the Heuristic

Segmentation to determine the rest of the animated characters not included in our processor. The attack achieved a success rate of 56% in segmenting the CAPTCHA and 40.4% in recognising them. This attack encountered the 2013 version of ReCAPTCHA by increasing the level of CCT segmentation and character distortion. As a result, the CAPTCHA resisted the previous segmentation attacks that used pattern detection algorithms.

Lastly, concurrent to our own work, Bursztein et al.[79] with Google and Stanford University, presented another attack that broke the 2013 version of ReCAPTCHA with a success rate of 22.34% and an average time of 6.22 seconds. This attack also uses the automatic segmentor in addition to the machine learning technique, which segments the text-based CAPTCHA schemes. The first step in this attack uses a cut-point detector to choose cut points between the upper and lower sides of the targeted text. It then starts to slice the text by connecting the points in the upper side with other points in the lower side of that text. Next, it uses a recognition technique that uses the OCR and k-natural-network to generate a number of possible answers. The attack then uses the correct answer with a sequential learning approach in which they provide a score for generated answers to choose the correct answer between them. Compared to our work, this attack uses a method that is more complex against the text-based CAPTCHA schemes. It also combines between two techniques were not included in our attack. The first technique is an advanced machine learning that keep training the engine to improve the result. The second technique is relay attack technique that send the challenge to third party to assist the machine learning by providing it with the correct answer for that challenge.

Yan et al.[80] developed another attack that is based the automatic segmentor strategy in 2015. This attack uses Log-Gabor filters to extract character components from CAPTCHA image alongside information about it. This information includes details about aligned parts of the component including their alignment angles. The attack then uses a colour filling attack algorithm to get the order of the aligned parts of the text. After that, it uses both information about the order of the aligned parts and their alignment angle to supervise a natural network engine in recognising and segmenting those components. This supervision includes generating graph that display the possible answers based on the order and alignment angles of those parts. This attack has achieved a success rate of 10.27 within an average time of 10.46 seconds against the ReCAPTCHA.

6.8 Summary

We have demonstrated the vulnerability problem segmentation resistance mechanism used in the ReCAPTCHA, and the mechanism that hides the main CAPTCHA by adding another string to the CAPTCHA image. This attack was implemented using the automated segmentor technique to create two maps. The first map uses the original technique that analyses the string without segmentation. The second map uses a technique that analyses parts of the string segmented through a method that detects the connection loops and segments the areas between them. The attack uses both maps to create another map, which generates the median results between them and analyses it to identify any common errors inside this map. Our attack achieved a 43% success rate against this CAPTCHA. This result shows the weaknesses of the current version of the ReCAPTCHA against the attacks, including the attacks that use the automating segmentor technique. It also indicates that using another word to confuse the attacks is not effective, as the automated attacks could still identify it.

Chapter 7. The Xu's version of NuCAPTCHA

In this chapter, we present our attack against an "emerging image" version of the NuCAPTCHA scheme created by Y. Xu. This CAPTCHA scheme is known by its mechanism, which displays rotated characters that overlap and move together horizontally through the animated frames. Our targeted version of this animated text-based CAPTCHA is the most secure version based on its use of the "emerging image" mechanism. This mechanism forms a black and white animated image that displays only the edges of the animated objects in the CAPTCHA challenge in the form of black segments surrounded by noises. Our attack against this CAPTCHA scheme achieved a success rate of 67.5% in recognising the samples of the CAPTCHA, which is the first successful attack against a motion-based CAPTCHA that uses the "emerging image" mechanism. The goal of our attack was to expose the weaknesses of this CAPTCHA scheme and to prove that the text inside it could be identified and to break through the automated attacks.

7.1 Introduction

In this chapter, we present an attack against the "emerging image" mechanism implemented by Y. Xu et al.[10] on the NuCAPTCHA scheme. This scheme was originally known by its first segmentation resistance animated text-based CAPTCHA that used an animated form of a CCT mechanism. This mechanism overlapped floating characters and moved them together in one horizontal direction through an animated video stream. The characters also rotated individually in different directions between two specified angles. Each challenge also contained an animated background to make it more difficult to track the CAPTCHA characters and catch them. Our target also included another mechanism known as an "emerging image" that displays edges of the text and background in the form of noisy black segments in front of a white background.

The "emerging image" mechanism used in this scheme is one of the most challenging mechanisms for attackers. This mechanism relies on the unique human ability to aggregate information from seemingly meaningless pieces. The main robustness feature of this mechanism comes from the way it spreads the segments of the characters and backgrounds through a number of frames. This causes every frame to display different segments of the background and characters than that shown in the other frames around them. These frames also include other noises shown around the characters and background.

The main advantage of the mechanism is its ability to prevent the computer from displaying text and background in one frame. Such a mechanism causes the CAPTCHA challenges to resist the frame-to-frame attacks that analyse each frame individually. It also prevents the attacks from gathering image segments from the sequence of frames without the noises around them. This results in a near black image that is filled with noises instead of one that shows the segments of the image gathered in one frame. These types of problem are not present for humans who can differentiate between the noises and segments of the animated text and background. The cause of this usability advantage are the differences in the look and animation between the segments of text and background. These differences make segments of the text more distinguishable to the human from the other noises around them. The version we targeted also included another feature that makes the segments of the characters visible to the human.

Researchers have shown the robustness of this version of the NuCAPTCHA to current attacks after they targeted it with the same automated attacks used to target the original schemes. Based on their claims, all recent attacks failed in breaking this version of the NuCAPTCHA. The reason behind this failure is that no single frame displays the characters in a way that allows the automated attack to differentiate them from the background. Therefore, we have targeted this CAPTCHA using a new attack that detects the segments of text and removes the noisy background around them. It then gathers the segments to create an image of the characters that is detectable and recognisable by the attacks.

To our best knowledge, the attack presented in this chapter offers a critical analysis of the "emerging image" and gives a clear insight into its robustness against attacks. It also shows the exploitable key vulnerabilities found on the design of the "emerging image" version of NuCAPTCHA.

The sections in this chapter are organised as follows. Section 7.2 provides an overview of the targeted CAPTCHA scheme. Section 7.3 describes our attack against Xu's version of NuCAPTCHA. We discuss our evaluation in section 7.4 followed by the results in section 7.5. Section 7.6 discusses defence from the attacks, while section 7.7 summarises our findings.

7.2 The characteristics of the Xu's version of NuCAPTCHA

The Xu's version of NuCAPTCHA that we targeted is created by an approach that uses two sets of animated frames. The first set includes grayscale images that display the edges of an animated text alongside the edges of the background. The animated text included in the image contains three characters and shares the same animated feature as the original text in the NuCAPTCHA scheme. The second set of animated frames originate from a set of black and white noisy images. This approach used to create the animated scheme combines the two sets of frames and then converts them into binary images to create a challenge that displays the edges of both the text and the background in blurred images.

The results of the method used to create this CAPTCHA can be shown in figure 7.1 which displays an example of Xu's "emerging image" version of the NuCAPTCHA. The features of this CAPTCHA design, as the figure shows, can be represented in the following ways:

- Every frame is a binary frame.
- Each frame includes black segments that represent the animated text and the background around it as small black noises in front of a white background.
- The text in this CAPTCHA contains three animated characters which rotate individually. They also move together from left to right, while floating in two opposite directions, up and down.
- The characters used in our CAPTCHA scheme include capital letters and numbers only.
- The animated noises appear randomly in different areas in the animated image.
- The noises are smaller than segments and have more distance between each other than the segments in the image.
- The few segments of characters appear to be randomly thicker and blacker through the animated frames

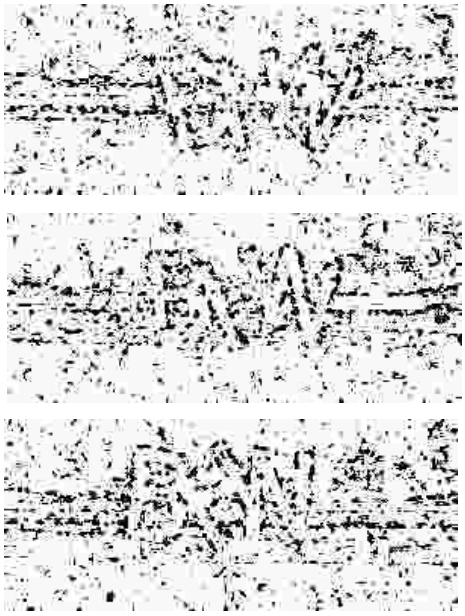


Figure 7.1. Xu's version of NuCAPTCHA

7.3 Our attack against Xu's version of NuCAPTCHA

This attack includes two main steps:

- Tracking/segmentation attack: this step includes method to clean the animated text-based CAPTCHA from noises, and segment the text.
- Recognition attack: this step uses the recognition engine to recognise the segmented text.

7.3.1 *Tracking/segmentation attack*

We have targeted this CAPTCHA with an attack that includes steps that distinguish the text characters through the patterns of their segments, their movements and pace of those movements, before it segments them. The attack includes three steps to clean the background from the noisy elements. It also includes one step to create a recognisable image of the text and one step that segments the characters. We describe these steps as follows:

- Remove the distant noises from the text: this removes the segments of the background image that are located furthest from the text.

- Identify the area where the text characters move horizontally: this step tracks the frames of the CAPTCHA. It then selects the areas where the characters move from right to left and connect noises outside this area.

- Select the exact position of the text segments in the identified area: this step includes a method that marks the exact coordination of the four sides of the animated text by using them to draw a rectangle around the text. It then removes the noisy elements connected to the text that are located outside this rectangle.

- Merge the segments from a sequence of frames and analyse them: this step includes approaches that merge the segments from three sequences of frames into one frame. The resulting image shows the area containing the edges of the text in a near complete image. It also includes another approach that cleans each resulting frame from the noises left around the text.

- Extract and segment the animated text: this step includes approaches that extract the text and segment the animated characters in it.

Remove the distant noises from the text. The first stage of our attack includes approaches that identify and remove most noises and segments of the background that remain far from the animated text. These approaches identify the text based on their thickness and behaviour of the areas where they are gathered. The first approach includes three steps:

- The first step includes a method that lowers the thickness of the noises and segments in the image by removing one pixel from the upper and right side of each noisy element. This method also removes the noises and segments that have the lowest thickness than those analysed in the later stages of our attack.

- The second step includes a method that groups the segments left from the first step that are close to each other within a maximum distance (d), which equals 6 pixels in the case of our attack. This method connects those groups by adding a stroke layer with a thickness of $d/2$ around each of the segments. After that, it determines the size of each group of segments connected through the added stroke layers with the following equation:

$$T = C + \sum_{n=0}^{k} S_n$$

Where S_n is the size of each n segment included in which the total number of those segments are k, and C_i is total size of the area filled with the stroke that connects them.

- In the third step, the attack removes each group of segments that does not reach the limited size of L, which is the lowest size of grouped segments that represent the text in each frame.

The second approach repeats the first two steps from the earlier approach. The difference between these approaches is that the maximum distance (d) in this approach is two times greater than the maximum distance in the previous approach.

The third approach only selects the largest sets of connected segments with strokes those that include the text segments in each frame. This approach benefits from the earlier approaches, as they generate frames that display only areas containing many segments and noises. After that, it determines the total size of the segments and their strokes within every vertical column that have a length of ten pixels. It then cleans the vertical columns where the total size of the segments and their strokes does not reach the limited size determined for a text. This method generates a set of frames that show the text and other areas that contain huge numbers of thick noises and segments near it.

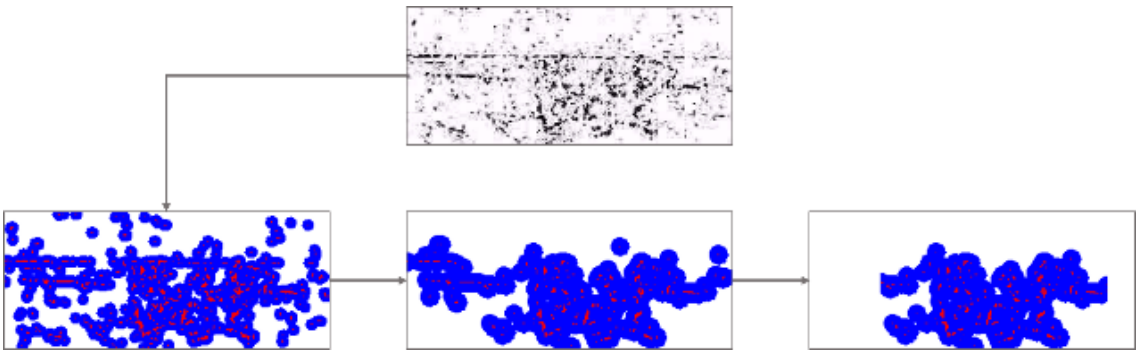


Figure 7.2. Remove noises and background segments

Figure 7.2 shows an example of this process where section A displays the way this CAPTCHA analyses areas. It then identifies small noises and removes them. Section B displays the image created from the second approach to remove the rest of the noises and small segments of the background. Section C displays the final image after using the approach in the challenge.

Identify area where segments move horizontally to one direction. The next step in our segmentation attack uses a tracking method. This method firstly generates many frames that include only huge coloured areas, which include segments and very thick stroke areas generated around them. The thickness of the strokes around the segments in our attack is determined by the movement direction and speed of the text. They also include emerging behaviours that cause the temporary disappearance of the text sides. This thickness can be calculated using the algorithm:

$$Thick = S_m N_t$$

In which S_m is maximum speed of the animated text, N_t the maximum thickness of noisy segments of the characters that could be hidden in each frame.

After that, we compare each combination of segments and stroke layers with the other combinations in the previous and next frames. We then use the results to identify the segments that move alongside the animated areas in one direction, which is from right to left in our target. After that, we create a new frame image that shows the position where segments and stroke layers of the three sequences of frames meet together. We ran this tracking method to analyse all frames in two sequential directions:

- The first direction starts from the first frame to the last frame
- The second direction is the opposite direction from the last frame to the first frame.

We kept running this method until we generated frames that showed only segments of the text alongside background segments that moved in the same direction.

Figure 7.3 shows how this tracking method works.

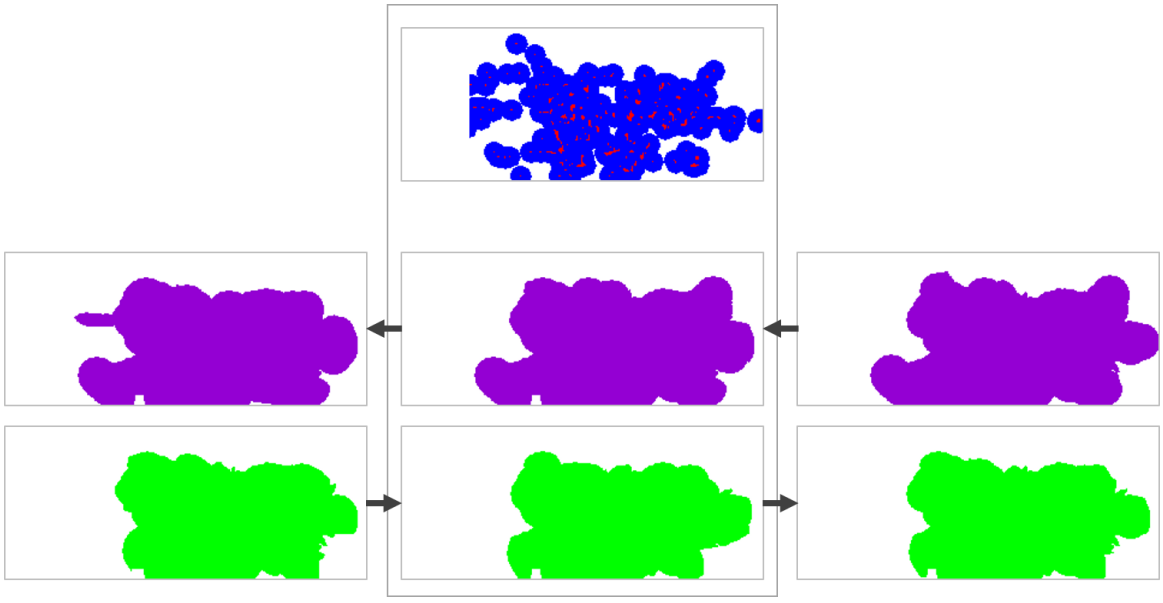


Figure 7.3. Identifying the area where segments move to one horizontal direction: the purple colour shows the scanning from the first frame to the last frame, and the green colour shows the scanning from the last frame to the first frame

Select exact position of the text in the identified area. In this step, we use a method to determine the position of the text inside the area identified in the previous step. This method scans the resulting frames in the previous step alongside the original ones. It then uses the results to create a map of segments and noises appearing within the final area identified in the previous method. Thus, the map divides the segments into three different sets based on their thickness. We marked each set using one unique colour. The first set of these areas includes those with a thickness of only one pixel, while the second set contains those with a thickness of two pixels. The third set contains those with a thickness of more than two pixels.

We then scan this resulting map through an algorithm that identifies the exact position of the text in each frame and mark it using a rectangle that matches the maximum estimated width and height of the text. This algorithm gives a specified score for each pixel appearing in the specified rectangle area based on its colour in the previous map. The algorithm can be represented in the following way:

Algorithm: the rectangle poisition calculation
Input: A(Colour map)
Output: the point P (start cordinate of rectangle)
1: $N \leftarrow 0$

2: $P.x \leftarrow 0$
3: $P.y \leftarrow 0$
4: **for** $x \leftarrow 0$ **to** $width$ of $image$ **do**
5: **for** y $\leftarrow 0$ **to** $height$ of $image$ **do**
6: E(x, y) $\leftarrow 0$
7: **for** $X \leftarrow x$ **to** $140(width\ \text{of}\ rectangle) + x$ **do**
8: **for** $Y \leftarrow y$ **to** $75(height\ \text{of}\ rectangle) + y$ **do**
9: **if** $A(X,Y) = violet$ **then**
10: $E(x,y) \leftarrow E(x,y) + 50$
11: **else if** $A(X,Y) = blue$ **then**
12: $E(x,y) \leftarrow E(x,y) + 5$
13: **else if** $A(X,Y) = Cyan$ **then**
14: $E(x,y) \leftarrow E(x,y) + 1$
15: **end if**
16: **end**
17: **end**
18: **if** $E(x,y) > N$ **then**
19: $N \leftarrow E(x,y)$
20: $P.x \leftarrow x$
21: $P.y \leftarrow y$
22: **end if**
23: **end**
24: **end**
25: **return** P

The cyan colour represents the area with one-pixel thickness, the blue colour represents the area with thickness of two pixels and the violet colour represents the area with a thickness of more than two pixels. Figure 7.4 shows how we created the map. It also shows how we located the exact position of the text in this map.

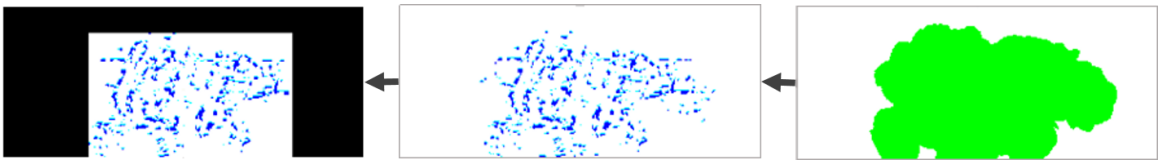


Figure 7.4. Example of the determination of the exact position of the text

Merge the segments from a sequence of frames and analyse them. The attack in this step creates a number of frames that display the edges of the text characters in nearly full image through a method that joins the segments extracted from a sequence of the frames generated in the previous step. The resulting images of the text in these frames are identified as the closest possible images to the original text displayed in the original NuCAPTCHA

image. Thus, the attack can easily track the movements of those edges and distinguish them from any remaining background segments around them.

The method used to clean the remained segments is similar to that used in identifying the area that moves horizontally in one direction. This method uses different levels of thickness of strokes, so it can identify the characters within the text through their horizontal and rotational movement behaviours. The result of this identification is then used to remove all the noises and segments that do not follow the movement of the characters from the area around them. The step also includes an additional method which marks background segments that keep appearing in fixed locations using a unique colour, so it can avoid selecting them as part of the text in the next step. Figure 7.5 shows how we gathered all three frames together in one frame then removed the rest of the noises around the text in that frame using the same tracking method used in the previous step.

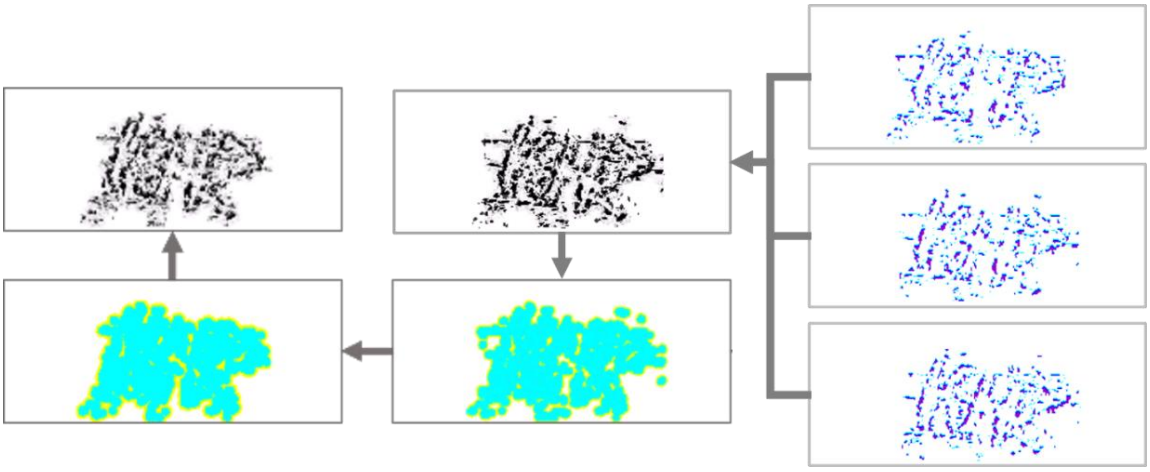


Figure 7.5. Merge and clear segments

Extract and segment the animated text. The final step in our segmentation attack involves a method that extracts and segments the text characters in each frame. This method selects all the frames that show the full image of the main text and then extracts the text into a new sequence of frames.

The method uses a rectangle to analyse each of the vertical columns where each character is displayed in each frame. It then uses the result of the analysis to identify the specified horizontal columns where the characters are displayed in every frame. Finally, it extracts each image of a character from a horizontal column into a folder that includes all other

images of that character. Figure 7.6 shows how we segmented the characters “K”, “H” and “Z” in this step.

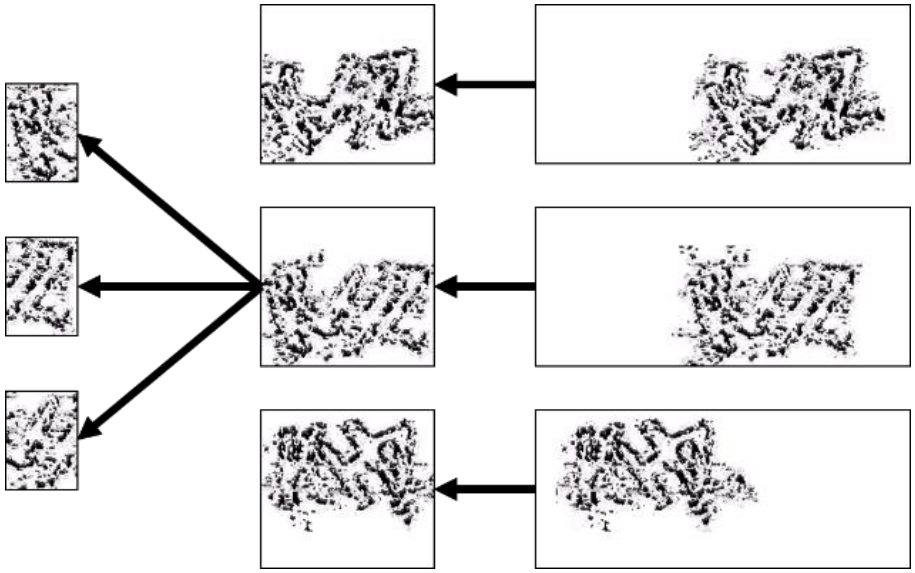


Figure 7.6. Segmenting the characters

7.3.2 *Recognition attack*

In this step, we used the Convolutional Neural Network (CNN)[110] engine to recognise the extracted characters as in our attacks in previous chapters. We trained this engine to recognise characters inside the CAPTCHA images by creating a training set of 10098 sample images of characters (197 sample images for each character) and testing sets of 6153 sample images of characters. After that, we used two datasets to train our engine 50 times and the result was the following:

- The engine achieved a 0.3% error rate in training after it trained 10069 sample images of characters from the training dataset successfully.
- It achieved a 0.2% error rate in testing after it recognised 6138 sample images of characters from the testing dataset successfully.

Figure 7.7 shows how the training and testing accuracy rates were improved during the training task.

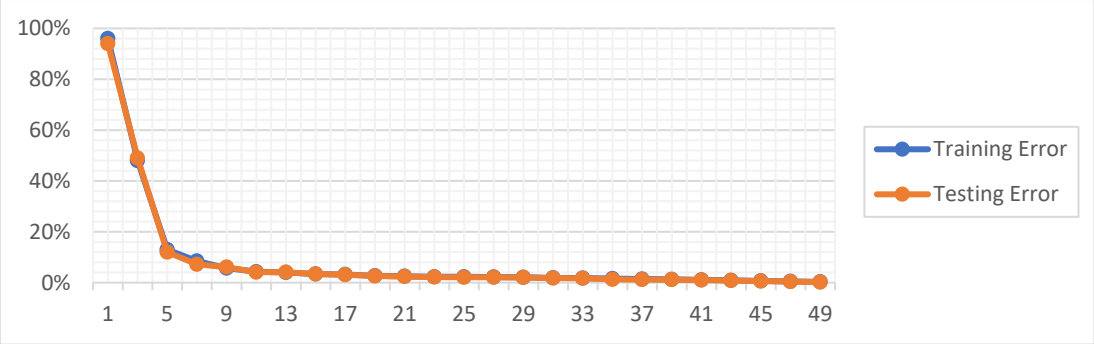


Figure 7.7. Training and testing accuracy rate during the training of the CAPTCHA

After that, we used this recognising engine against each set of images generated for each CAPTCHA character. We then collected the result and used it to determine the most recognised character for every set of images taken for every CAPTCHA character from the frames inside the CAPTCHA challenge to find the right answer.

7.4 Evaluation

We discuss the results of our attack in terms of segmentation and recognition using the same method included in the previous chapters. Our evaluation in this chapter was based on an examination of 300 random samples of this CAPTCHA. We obtained some of those samples through an automated screen capture designed using C# between January and February 2015 from Y. Xu's website[116]. We generated others through automated tools by implementing the following instructions in Xu's paper[10]. We used this tool to generate many samples including many "emerging image" versions of NuCAPTCHA that are different in background and noise levels. We used 100 of these samples in designing and training our attack. We saved the other 200 samples locally to be used as text set in the final examination of our attacks. We did not include any sample used during the design of the attack in the evaluation process of our attack.

7.5 Results

Success rate. We firstly tested our attack against a sample set of 100 CAPTCHA challenges used for training purposes, and it achieved a success rate of 77% in recognising them and an 84% success rate in segmenting them. After that, we tested our attack on 200 random test samples of Xu's NuCAPTCHA, and it achieved an overall success rate of 67.5% (135 out of 200) segmenting and recognising the samples of this CAPTCHA. These results include a success rate of 76% (152 out of 200) in segmenting the samples of this CAPTCHA

and an accuracy rate of 88.82% (135 out of 152) in recognising the successfully segmented samples.

The use of tracking methods in scanning all frames included in the animated image is essential to achieving our recorded accuracy and success rate against this CAPTCHA scheme. The tracking methods should also be used in scanning the sequence of frames for one round in two directions. The increase of scanning rounds only helped to increase the accuracy rate of our attack to a maximum rate of 1.97%. The results showed that the standard deviation of the increase in overall success rate is 1.005%, until it reaches a maximum accuracy rate of 92.11% in the third round. This increase in accuracy rate could only increase the overall success rate to a maximum rate of 1.5% with a standard deviation of 0.7638%, until it reaches a maximum accuracy rate of 70% in the third round. Figure 7.8 shows more details of this increase.

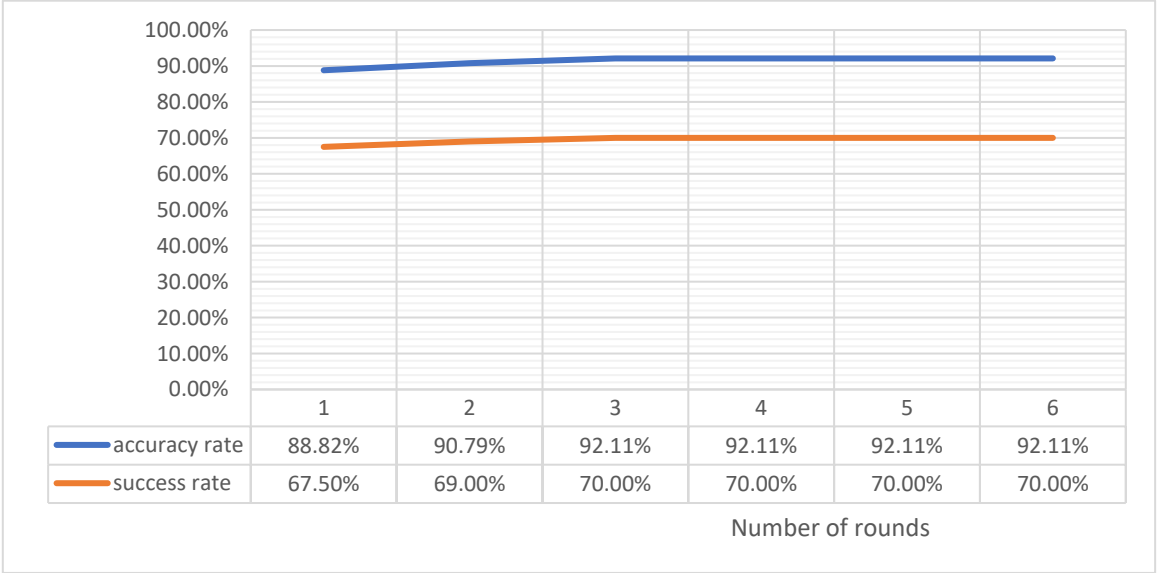


Figure 7.8. The increase in accuracy/success rate per scanning round

We also identified the following failures in our attack:

Failure 1: removing the noises. The most common problem we faced in our attack was in fully removing the noise near the characters, especially in the areas totally filled with noises and background segments. The main cause of this problem was the difficulty in distinguishing the characters from the other segments beside it, as the merging technique could make those noises appear as part of the characters. Figure 7.9 shows an example of the failure in removing the noises.

Figure 7.9. Examples of noises removal failures

The only solution for this problem is to create a grayscale colour level for the thickness, because the CAPTCHA characters always have parts that are thicker than other parts of the image to make them clear for the human.

Failure 2: segmenting the characters. This problem happens occasionally when one of the characters on the sides is very wide, like "W", and the attack could not identify the width of those characters due to the noise. This problem could also occur when the sides of characters keep blurring and disappearing, as it is difficult to identify the space taken by them when they are rotating. Figure 7.10 shows two examples of this problem.

Figure 7.10. Failure in segmenting the characters

This problem could be solved by implementing more a complex segmentation method that tracks the rotation of each animated character, such as the method used by Y. Xu et al. [10]. However, such a method will take a long-time due to the complexity of the emerging technique which is required to scan every segment of the characters unlike the original version, which was attacked by Y. Xu et al.

Figure 7.11 shows rate of failure in our attack against the Xu's version of NuCaptcha.

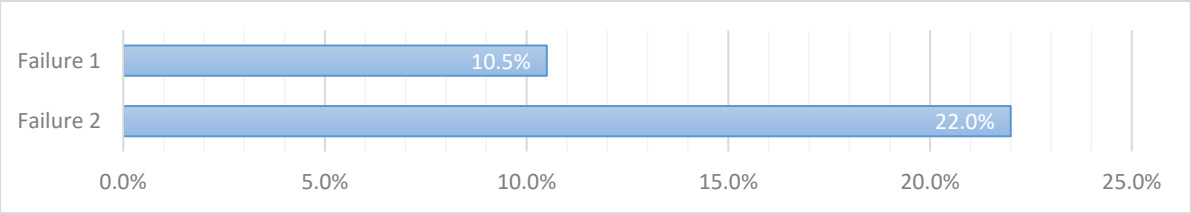


Figure 7.11. Failures rate in our attack against the Xu's version of NuCaptcha

Attack speed. Our CAPTCHA attack was implemented using JAVA, MATLAB, and C# programming languages, and tested on a computer with an Intel(R) Core(TM) i7-4700MQ CPU 2.40GHz and 8.00 GB ram. The results show that the attack took an average of 40569.7452 milliseconds to analyse each animated challenge taken from the Xu's version of NuCAPTCHA with standard deviation of 598.71481 milliseconds. The shortest attacking time was 39572 milliseconds, while the longest was 41649 milliseconds.

7.6 Defence

The use of "emerging image" has the potential to make the CAPTCHA secure. However, it still suffers from robustness problems related to usability. One of these problems occurs when CAPTCHA characters have a unique look and animation behaviour, since the computer can always distinguish the characters from the other elements in the image through those behaviours. This could make those CAPTCHAs vulnerable to attacks and detectable by them.

One solution for this problem is the techniques that display the characters in the same behaviour as the noises and the background elements behind them. These techniques can include animated backgrounds that can be moved in the same direction and at the same speed as the animated characters behind them. In addition, they can include techniques that make the characters move randomly to prevent the attacks from identifying their positions through their movement. However, these techniques should not include any method that could display the elements of the characters in unique thickness or unique overcrowding behaviour as those two behaviours can be detected by the computer.

7.7 Summary

We have developed an attack against the Xu's version of NuCAPTCHA. This attack uses tracking techniques to detect the segments of the characters and gather them through a specified sequence number of the CAPTCHA frames. Our attack achieved a 67.5% success rate against this CAPTCHA, after it was tested against 50 samples of the CAPTCHA animated challenges. As result, our attack has shown the vulnerability of CAPTCHA schemes that use the "emerging image" mechanism against attacks, since the usability of those CAPTCHAs depends on the behaviour of their segments and noises. Therefore, the attacks can detect the segments of the CAPTCHA elements and differentiate between them and the noises through their animated behaviours.

Chapter 8. Kund’s CAPTCHA

In this chapter, we present an attack against the Kund's animated text-based CAPTCHA. This CAPTCHA includes a “noisy lines” mechanism that displays the main object and background in the form of small lines. Our examination of this CAPTCHA involved two attacks. The first attack included an automatic segmentor whereas the second did not. The first attack achieved a 38% success rate against this CAPTCHA and took an average of 150.924 seconds to solve the challenges. The second achieved a 26% success rate and took an average of 97.733 seconds to solve the challenges.

8.1 Introduction

In this chapter, we present an attack on “noisy lines” deployed by the Kund’s CAPTCHA, which is an animated text-based CAPTCHA created by Ivo Kund at the University of Manchester. This CAPTCHA contains small lines that appear in one specified size in each CAPTCHA challenge. We divided the small lines into two types of moving particles. The first type included groups of moving particles that represent the CAPTCHA characters. Each group of particles belonged to a character and showed only a few parts of it in each frame. The particles in each group also moved together in random directions through the animated frames. The second type comprised only noises of the background which could appear or move randomly in a separated group. The version of this CAPTCHA investigated contained many animated designs. Each design shared the same number of characters in the same font size.

The robustness features of “noisy line” mechanisms are like the “emerging image” mechanism in the Xu’s version of NuCAPTCHA. These features spread the particles of the characters and backgrounds through the animated frames. This causes every frame to display different segments of the background and characters than its previous and next frames. The features of a “noisy line” depend more on the movement behaviours of the particles rather than their shape. This feature could increase human ability to distinguish between the particles of characters and the noisy background without showing any unique patterns of the characters. Such features cause the Kund’s CAPTCHA to be more challenging than our previous target that uses the “emerging image” mechanism.

However, we were still able to break the Kund’s CAPTCHA that uses this mechanism with an attack that uses the tracking system to distinguish the noisy lines that form the

background from the particles of the main text. Our goal in this chapter is to identify the strengths and weaknesses of the “noisy line”. We achieved our goal with approaches that differentiated the particles of characters from the noisy lines that belonged to the background. They also segmented the characters and sent them to a recognition engine. This attacking approach achieved a success rates of between 26% and 38% in segmenting and recognising this CAPTCHA scheme. The success rates were high enough to prove the vulnerability of our target against the tracking attack systems. The results of our research helped show the limitations of the mechanism and to make suggestions on how to improve the robustness of the CAPTCHA against future attacks.

The sections in this chapter are organised as follows. Section 8.2 provides an overview of the targeted CAPTCHA scheme. Section 8.3 describes our attack against Kund’s CAPTCHA. We discuss our evaluation in section 8.4 followed by the results in section 8.5. Section 8.6 discusses defence from the attacks, while section 8.7 summarises our findings.

8.2 The characteristics of the Kund’s CAPTCHA

This CAPTCHA includes separated characters. These characters are displayed through their edges as groups of moving particles. It also includes other particles that belong to the noisy background. All the particles have different movement behaviours than the particles of the characters. Figure 8.1 presents the features of this CAPTCHA as follows:

- All characters and the noises displayed through the small particles are in the shape of tiny lines. Those characters share the same colour and length in each challenge.
- There are three types of particles. The first type is shaped from tiny lines and has a length of six pixels, the second has a length of eight pixels and the third type has a length of twelve pixels.
- All characters are capital letters which share the same font type and font size in each challenge
- The characters could move and rotate individually through the animated frames.
- All the fonts used in the challenges are from the san-serif typography design.
- The particles that represent the CAPTCHA characters are close to each other.

- The particles are grouped to constitute the edges of the characters.
- The particles to each character are either aligned all together in one direction or aligned partially in a way to display the exact directions of the edges of the characters.
- The particles can vibrate in few challenges. They could also rotate individually or all together in one direction or remain in zero rotation though the animated frames.
- Each group of the particles related to one character move all together through one unique, looped circle.
- However, each pair of characters may also connect with each other through a sequence of frames.
- Each particle of each character can be randomly invisible for a random sequence of frames of the CAPTCHA challenge.
- The background noises can be more separated than the particles of the characters in few animated designs.
- The noises can also move more separately in different directions and could appear more quickly than the particles of the characters.
- We can divide the designs of this scheme into three categories. Each category has a unique number of characters, particles length and font size as follows:
    - The designs of first category display four characters that appear in small font size. The particles used to shape the edge of those types of character alongside the noisy background appear in the shape of six-pixel lines. Figure 8.1 (A) shows an example of this category.
    - The designs of second category display three characters that appear in medium font size. The particles used to shape the edge of those types of character alongside the noisy background also appear in the shape of eight-pixel lines. Figure 8.1 (B) shows an example of this category.

- The designs of the third category display two characters that appear in big bold font. The particles used to shape the edge of those types of character alongside the noisy background appear in the shape of twelve-pixel lines. Figure 8.1 (C) shows an example of this category.

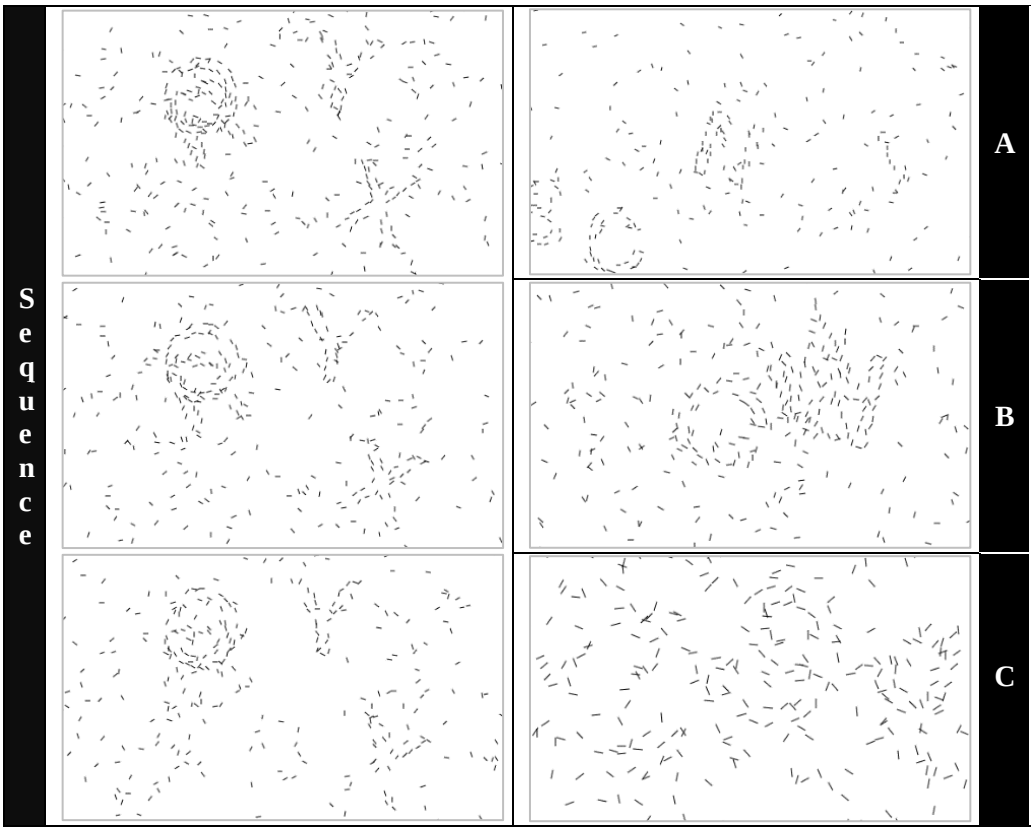


Figure 8.1. The Kund's CAPTCHA

8.3 Our attack against the Kund's CAPTCHA

This attack includes two main steps:

- Tracking/segmentation attack: this step includes method to clean the animated text-based CAPTCHA from noises, and segment the text.

- Recognition attack: this step uses the recognition engine to recognise the segmented text.

8.3.1 *Tracking/Segmentation attack*

Our tracking/Segmentation attack includes the following steps:

- Pre-processing: this step modifies the animated frames from the animated challenge for the next steps of our attack.

- Removing the noisy background: this step identifies the noisy particles that are not close to the characters. it also determines the way they appear and move through the animated frames.

- Marking and segmenting the animated characters in the animated image: the attack in this step uses the colour filling attack to mark the components of the segmented and connected characters. After that, it scans the components of connected characters in each frame to determine the strength of their connection. It then segments them by marking characters in a different colour in every frame in the animated image.

- Extract and characters: the attack in this step extracts all the characters. It also segments the connected characters that are identified in the previous steps.

- Create a full image of the characters: the attack in this step combines the segments of the characters to create a recognisable image of it.

Pre-processing. In this step, we create a mapping frame for every frame extracted from the animated challenge. We generate this mapping frame through a method that merges the particles of the three sequences of frames into one mapping frame. This mapping image displays the particles of merged frames using a binary-based RGB colouring model. This model uses values ranging from 0 to 1 for each of the RGB primaries of the colour. We use each of the three primary colours as a coding colour for one of the three merged frames. We can determine the colour of the noises of the mapping image of the frame $[F_i]$ as follows:

- the particles of the first (previous) frame $[F_{i-1}]$ are marked using green colour (G=1).

- the particles of the second (main) frame $[F_i]$ are marked using red colour (R=1).

- the particles of the third (next) frame $[F_{i+1}]$ are marked using blue colour (B=1).

It also shows the coordinated points at which the noises of two or more merged noises are located using combined colours of those components. For example, the coordinated points where the particles of the previous frame meet the particles of the middle frame are marked in yellow (R=1, G=1, B=0).

The image combines the colours of particles of different frames that meet in one pixel. Figure 8.2 shows an example of the new frames.

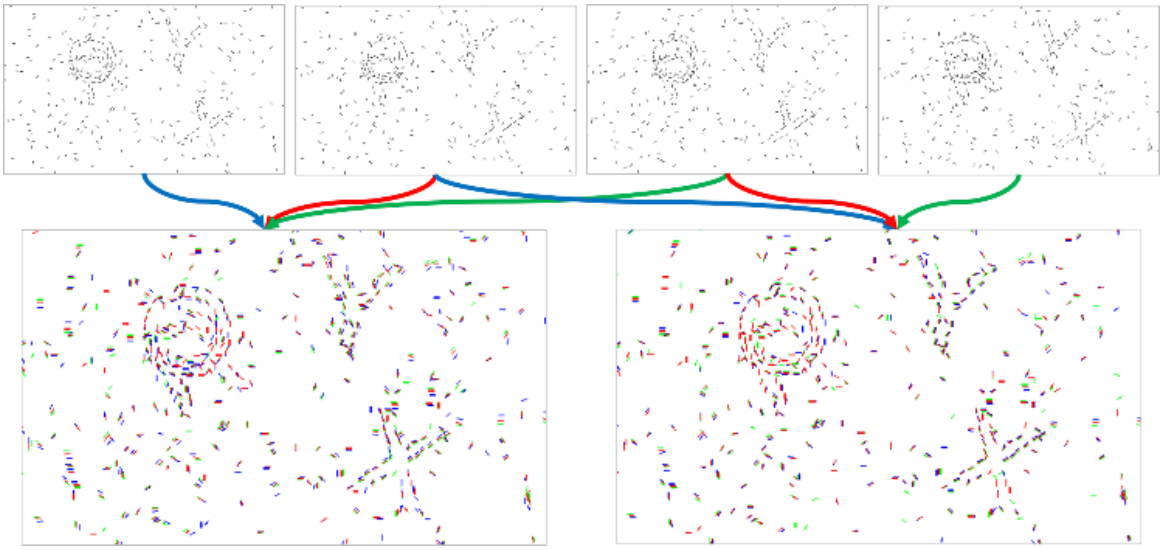


Figure 8.2. The new sequence of frames

Removing the noisy background. This step involves steps to clean the animated frames from noises around animated characters. These steps are as follows:

- Remove the group of noises that disappear or move too fast
- Remove components that do not reach the size limit or number of particles required to constitute a character
- Remove the components that do not include a set of particles that reach the required intensity to constitute a character
- Select the area where particles of the characters match its movement behaviour.

Remove the group of noises that disappear or move too fast. In this step, we scan each mapping frame to determine the positions of the merged particles from three frames in the mapping frame. We then calculate the distance between the merged particle from the

middle frame (red) and its previous (green) and next (blue) frames. We then remove the red particles that are not located within a distance of $D(\mathfrak{C})$ from the nearest particles of both previous (green) and next (blue) frames. We also remove the particles of previous (green) and next (blue) frames that are not located within a distance of $D(\mathfrak{C})$ from a particle of the main (red) frame. We could then determine the distance $D(\mathfrak{C})$ through the following equation:

$$D(\mathfrak{C}) = M + \left(\frac{2\,\pi\,A}{360}\,r(\mathfrak{C}_{\mathcal{P}})\right)$$

Where both M and A are constant numbers. The M is the maximum movement speed of the main characters expressed in pixels per frame, and A represents the maximum angular speed of those characters in degree per frame. The $r(\mathfrak{C}_{\mathcal{P}})$ is the maximum radius of a rotating character. we can calculate through the equation:

$$r(\mathfrak{C}_{\mathcal{P}}) = \mathfrak{C}_{\rho}\,W_{c\to e}$$

where $W_{c\to e}$ is the distance between the centre of character "W" and $\mathfrak{C}_{\mathcal{P}}$ is the font size decided by the design category of the CAPTCHA challenge. We could determine the design category of the animated image $\mathfrak{C}$ by measuring the shared length of the noisy lines ($\mathcal{P}$).

Figure 8.3 shows resulting map of one frame after this step.

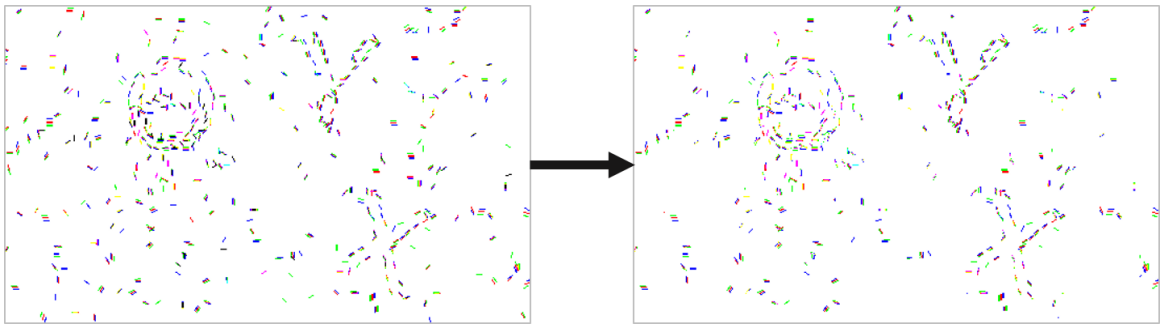


Figure 8.3. The removal of noises that disappear or move too fast

Remove components that do not reach limited surface area or number of particles required to constitute a character. In this step, we form components by selecting all particles located within a distance of R from other particles. We can describe the distance as the maximum gap between the particles of one character and is identified based on the average distance

between the noises. We connect these particles by adding a stroke border around each of them. This border has a thickness of $R/2$ around each of the particles. After that, the we calculate the surface area of each component formed by connecting the groups of particles (G) through the following equation:

$$S_t(G) = S_b(G) + \mathcal{P}\, T(G)$$

Where the $T(G)$ is total number of particles that belongs to the three merged frames within the area of G in the mapping frame, $\mathcal{P}$ is surface area of each of those particles (shared length of the noisy lines multiplied by one), and S_b is total surface area of stroke border that connects them.

We then remove each component that does not reach the minimum surface area of the animated character (S_t of character "I") in each frame. We also remove other components that do not contain a minimum number of O merged particles within their areas. We can determine the value of O as a minimum number of particles required to form a character. We keep components that pass the two conditions alongside the components to be located beside the edge to be analysed in the next step.

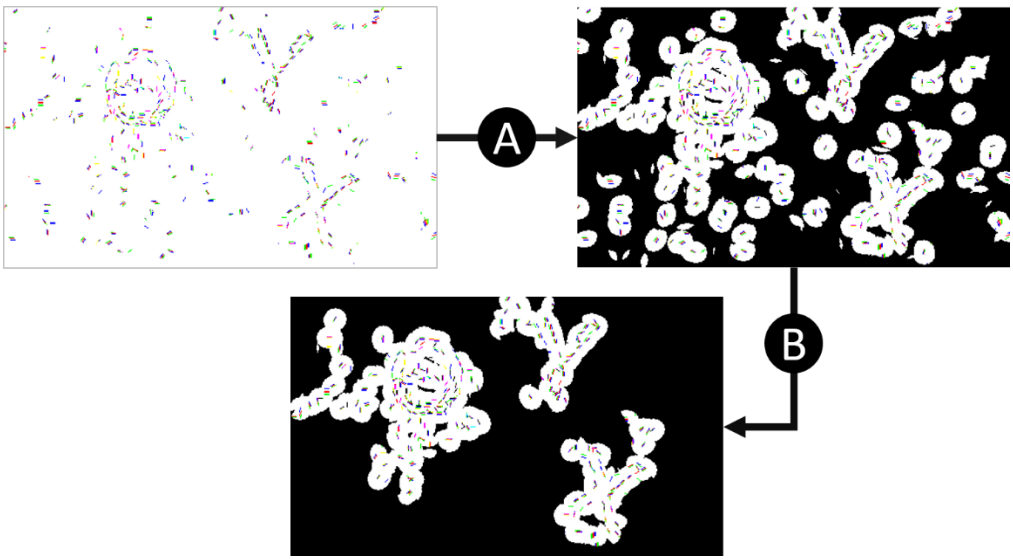


Figure 8.4. The removal of noises using the first limitation method

Figure 8.4 shows an example of our step. Figure 8.4 (A) shows how we formed the component by connecting the groups of particles together. Figure 8.4 (B) shows how we removed components that do not reach a limited surface area or number of particles required to constitute a character.

Remove components that do not include a set of particles that reach the required intensity to constitute a character. In this step, we determine the number of particles that remained within a close distance from particles within each component. Each of those particles needs to match the following condition: $D_a \leq L * M$, where the D is the distance between the centre of the particle a and the centre of the closest particle to it, L is the length of the lines that shape the particles, and M is the maximum distance between particles, which equals a static number (2.5). We then keep only the components that have a high number of particles to achieve a state for the final step. Figure 8.5 shows a map of one frame before and after this step.

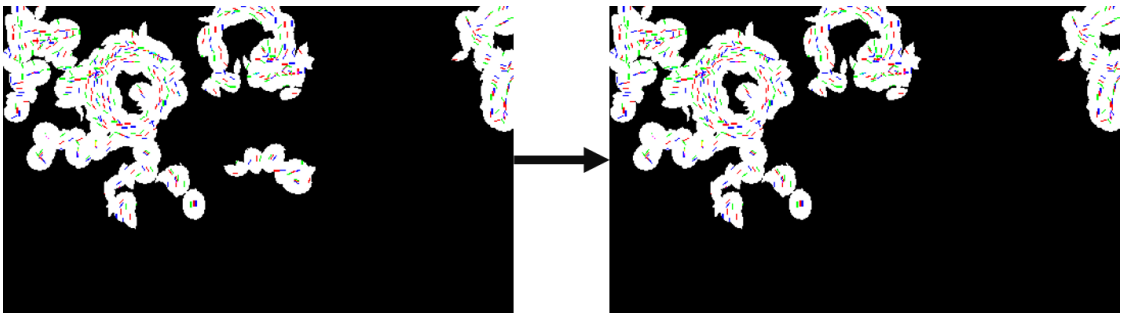


Figure 8.5. The removal of noises using the second limitation method

Select the area where particles of the characters match its movement behaviour. In this step, we use a tracking method to identify the CAPTCHA characters and clean them from noises. This tracking method is similar to the one included in the Xu's version of NUCAPTCHA that used the movement behaviour of the CAPTCHA characters to differentiate them from the noises. This step of our attack against the Kund's CAPTCHA includes a method that adds a stroke border around every component in each frame. We calculate the thickness of this border through the same algorithm used in our attack against Xu's NuCAPTCHA which is

$$Thick = S_m N_t$$

In which S_m is the maximum speed of the animated text, N_t is the maximum thickness of the noisy segments of the characters that could be hidden in each frame.

Our method then divides components formed in each frame into two groups. The first group includes all components that are fully located inside the frame. It also includes the most

components located beside the edge of the frame except those that are fully located within a close distance from the edges. We can determine this distance through the equation

$$D = S * (N + 1)$$

Where the S is the maximum movement speed of particles per frame. N is the number of the original frames merged in each of the analysed frames, which are three as stated earlier.

The tracking system analysed each of those two groups of components using two different methods. We present these methods as follows:

- The first method is processed against the first group of components. This method includes the same tracking algorithm to that used in Xu's NuCaptcha. This algorithm determines the coordination of all pixel points located in one frame. It then identifies the pixel points that share in the same coordination with other pixel points located in both the previous and next frames.

- The second method analyses the second group of components formed from the particles that appear within close range of the edge of the image. This method used a similar analysis process to the one used in the first method. It then identifies the pixel point that share the same coordination with other pixel points that are located in either the previous or next frames and belong to the first group of components.

After that, it selects those pixels to the next stage.

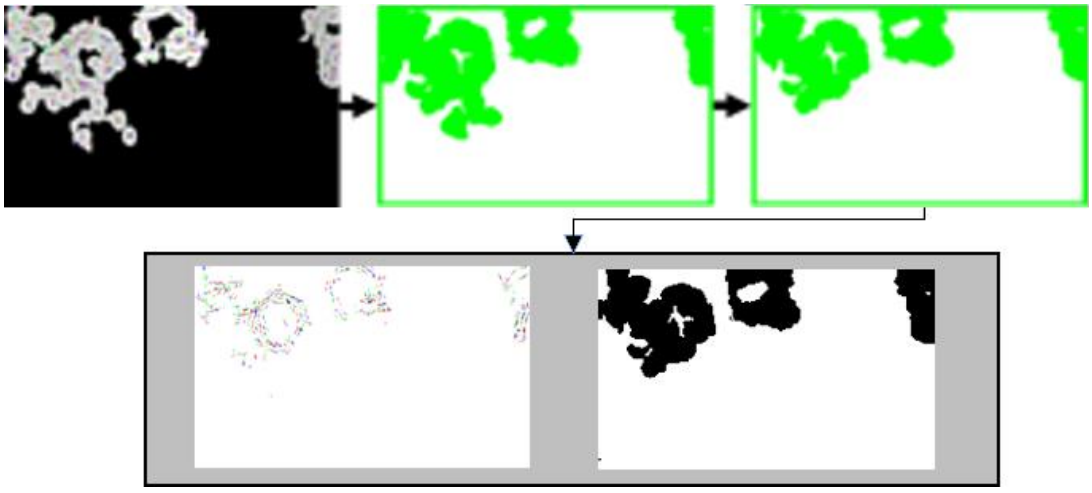


Figure 8.6. The removal of noises using tracking method

The attack uses both tracking methods to analyse the same of set of frames in two opposite animated directions - forwards and backwards. The attack then uses the results of those methods to create a new set of frames. Each of the frames shows the selected pixels through the previous methods. Figure 8.6 shows an example of this step in which the resulting pixels of the two methods are marked using a green colour after each run of the methods.

Marking and segmenting the animated characters in the animated image. In this step, the attack marks each group of connected pixel points resulted from the previous step as animated character components. The attack then tracks this group to determine if this component contains more unsegmented characters. It then uses the colour-filling algorithm that fills each animated component with one unique colour, namely the ID colour of that component, in one frame.

The attack then uses a method that links every component identified in one frame with the other images in the following frames, through a method that fills the other images of that component with the same ID colour of that component. This method identifies the closest A component to the component B filled using another unique colour in the next frame, by determining if most of the pixel points that belong to component A coordinated within the area covered by pixels of component B. It then links them together as one animated component by filling component A with the same colour ID of the component B.

After that, the attack uses another method that determines if any of the components identified in the frame is formed from connected components. This method determines if the pixel points of any component E is coordinated in the same area of components A and B in the previous or next frames. After that, it identifies the colour ID of the component E as an ID colour of a component that is formed from the connection of two components that have the colour IDs of A and B.

Figure 8.7 shows an example of this step. The two images of one component in two sequences of frames are linked by filling them using a blue colour, which is the colour ID of that component. The figure also shows a pink component in the current frame that is formed from a connection between two components linked with it in the next frame, which are the green and the red components.

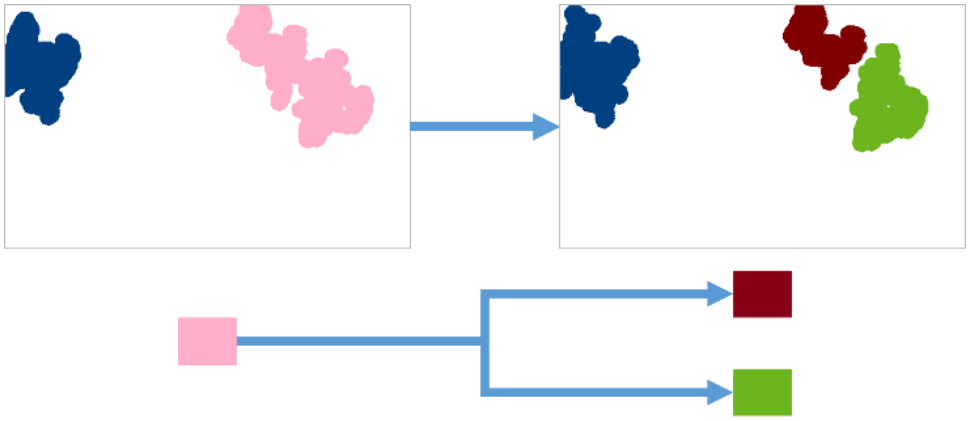


Figure 8.7. Marking and linking components together using a colour-filling method

Then, the attack uses a segmentation method that determines the movement direction of this segmented character using the equation of the parabola. This equation is calculated from the three different points identified in the centre of the animated component from three different frames. We use the equation to identify the position of the segmented character after it connects with another component. The equation calculated for a segmented character before it passes the connected area is similar to the equation for this segmented character after it passes that connection area. The equations can be also opposite to each other, if the segmented character is going backwards, after it passes the connection area. The attack then uses the equation to track the movement of two characters in each frame when they are linked to each other. It also uses the equation that determines if the two characters are not highly connected, so it can segment them by marking each of the segmented areas in a different colour.

It also uses a method to search for the components that contain connected characters that are undetectable in previous steps. This method analyses each component identified as a segmented character through the animated frame. It then uses the results to identify if each component keeps changing in shape and size through the animated frames. It then marks this component as one formed by connected characters that stay connected through the animated frames. We segment the component using an automatic segmentor, as it is difficult to segment through the previous steps. Figure 8.8 shows an example of the use of the tracking system in a segmentation attack, where it segmented two characters as it detected two other characters connected to each other through the animated frames using a black colour.

Figure 8.8. The use of tracking in segmentation attack

Extract the characters. In this step, the attack selects the pre-processing generated frames that display particles of the original CAPTCHA frame alongside its previous and next frames using three different colours, and then removes the particles which are outside the areas as selected character components through the tracking system in the previous step from that image. After that, it extracts each group of particles that appear within each area identified as one character component through the colour filling attack into a new image that displays it separately from the other groups of particles. It also groups the components identified as unsegmented connected characters to be segmented through a special (optional) step that uses the automatic segmentor.

Figure 8.9 shows that the attack extracted the two characters "Y" (blue component) and "X" (red component) separately before adding each of them to a folder that groups the components identified by their ID colours, while adding the components that contain the connected characters "C" and "R" to a special folder to send it to the automatic segmentor.

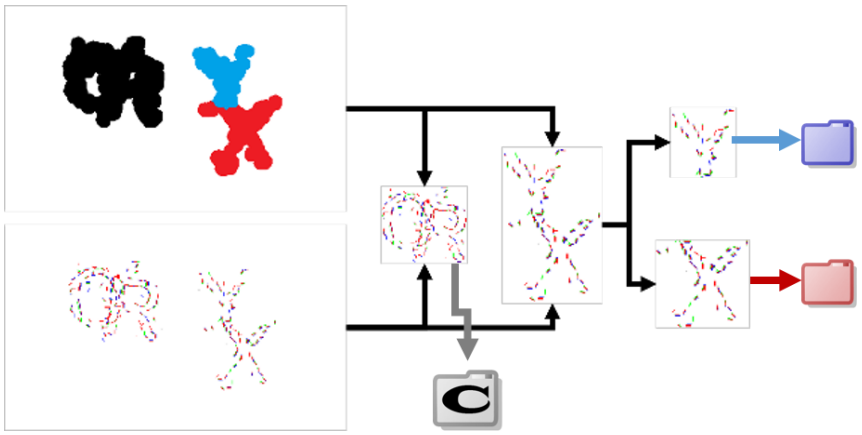


Figure 8.9. Extracting and segmenting characters

Create recognisable images of the characters. In this step, the attack uses a new approach against each extracted image that contains particles of a component identified as a single segmented character and analyses it to create a new image that shows the particles of the

edges of the characters nearly connected. This method that analyses the sets of particles that belong to the three emerged frames that are shown in different colours in the extracted image to identify the lowest and highest coordinates for each set over the x- and y-axis. After that, it uses the results of the identification to determine if the alignment of the lowest coordinated points of the three sets are the same as the alignment of the highest coordinated points of those sets over each of the two axis lines.

Afterwards, the method selects the set of particles that are aligned in the middle between the other sets of particles. It then shifts the other sets into the middle of the selected set. The distance of the shift is determined horizontally and vertically through an algorithm that identifies the shortest vector distance between particles of the middle set and particles of the other two sets over the x-axis and y-axis. This method should produce a new image of the character that shows its edge nearly connected and sends it to the recognition engine.

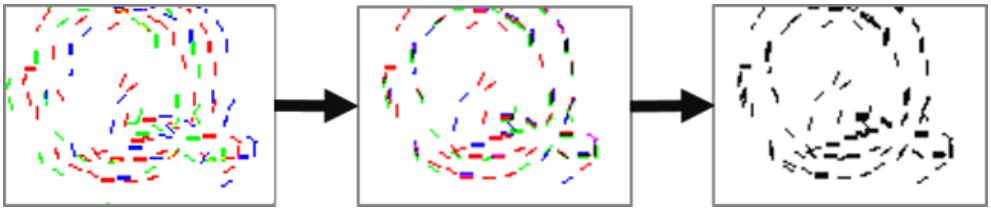


Figure 8.10. Creating a full image of the character

Figure 8.10. shows an example of this step in which our method identified a blue set of particles for the character "Q" on the right side of the image, a green set of particles on the left side of the image and a red set of particles in the middle. After that, it shifted the green particles to the right and the blue particles to the left using the equations mentioned before. As a result, they create a clear image of the character "Q", in which both green and red particles of the character appear in the same position of the red particles in the middle.

8.3.2 *Recognition attack*

In this step, the attack used two approaches against the images of this CAPTCHA. Both approaches use the same Convolutional Neural Network (CNN) engine as in our attacks in previous chapters. We trained this engine to recognise characters inside the CAPTCHA images by creating a training set of 7800 sample images of characters (300 sample images for each character) and testing sets of 4578 sample images of characters. After that, we used two datasets to train our engine 50 times and the result was the following:

- The engine achieved a 0.8% error rate in training after it trained 7736 sample images of characters from the training dataset successfully.

- It also achieved a 1.2% error rate in testing after it recognised 4521 sample images of characters from the testing dataset successfully.

Figure 8.11 shows how the training and testing accuracy rates were improved during the training task.

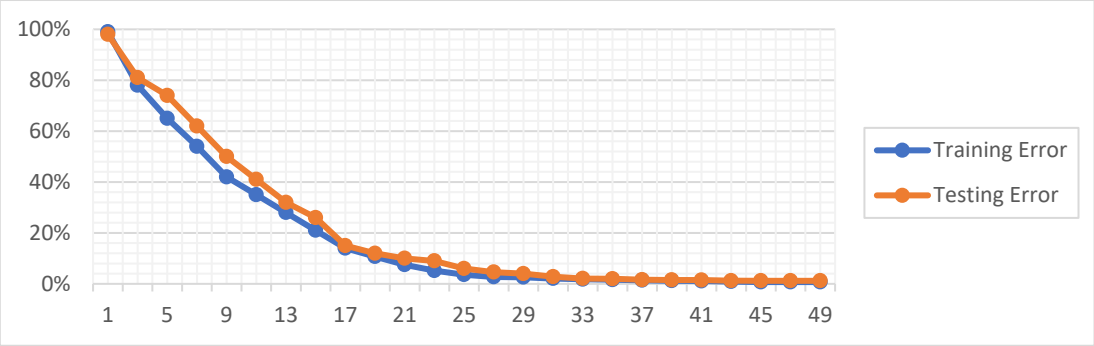


Figure 8.11. Training and testing accuracy rate during the training of the CAPTCHA

The two approaches are the following:

- The first approach: a main approach that includes a simple segmentation attack on extracted images of the segmented characters

- The second approach: an optional approach uses an automatic segmentor approach on the extracted images of the connected characters

This first approach. This approach selects ten extracted images for each segmented character identified through the previous approaches. The attack randomly selects those ten images from a group of extracted images that have a width and height that is higher than the average of all the extracted images for the segmented character. The attack analyses each selected group of extracted images of the segmented characters and then selects the most recognised character chosen from the group of extracted images to decide the right answer.

The second approach. It analyses the set of extracted images of the connected characters that could not be segmented through the previous approach using the automatic segmentor

(which we have used in our attack against ReCAPTCHA). The approach selects three images chosen randomly between five selected images that have the highest square size (width x height) between the set of extracted images for the connected characters.

It then generates a number of screenshots for each of the images using sliding windows. The size of each of the sliding windows is mainly determined by the average size of the segmented characters we already targeted in the previous approach. The approach uses sliding windows to catch the whole area of the connected characters with a method that divides this area into a set of points. Each of the points is identified individually within five pixels square and has a static distance horizontally and vertically between the other points around it, which is 10x10 pixels. Each of the points is used as a position for a sliding window to catch and extract the screenshot of the area around them. The approach then analyses the four edges of the screenshot (up, down, left and right) to check if the screenshot part of the component touches the four edges of the screenshot, before it is included in the group of screenshots in which the approach is used to generate the heat map in the second step.

In the second step, the approach sends all the screenshots associated with the data that shows its position in the extracted image in addition to its size and rotation to an automatic segmentor that is linked to a convolutional neural networks recognition engine [110]. This automatic segmentor then uses the recognition engine against each screenshot to identify the confidence rates for the CAPTCHA characters. Each of these confidence rates is the accuracy rate by which the image displays a specific character. The attack then calculates the average of the confidence rates for each pixel point from the images used to generate the confidence rates for each character through the following algorithm:

$$P = \frac{(\sum_{i=0}^{a} A[i]) + (\sum_{i=0}^{b} 10\ B[i]) + (\sum_{i=0}^{c} 100\ C[i])}{(a + 10b + 100c) * F}$$

Where A is the set of confidence rates of one character for the image where the character achieved a confidence rate that is lower than the other characters, while the B and C are the sets of confidence rates of the same character for the images where the character achieved a confidence rate that is the highest of all the characters. However, the confidence rates included in C are higher than 75%, unlike the confidence rates included in B.

The attack then gathers the result of the calculation for all pixels for each of the extracted images to create a heat map for each character that displays the average confidence rate for the character in each pixel inside the image. Then, it uses those heat maps to create another map that shows the position of the characters through the ID colour of the character that has the highest average confidence rate within the marked area of the text. The attack then uses the set of maps resulting from analysing extracted image of the analysed component to identify the connected characters that appear to achieve the highest average confidence rate in them. The number of the selected characters is mainly identified through the equation:

$$C = M - S$$

In which M is the maximum number of characters in each challenge identified through the average font size for the segmented character and the average length noisy line. For instance, the maximum number of characters is four if the font size used is small and the average length of noisy lines is six.

Figure 8.12 shows an example of the two recognition approaches

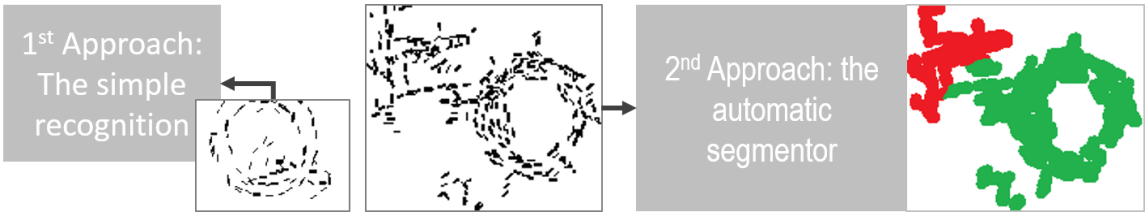


Figure 8.12. The two recognition approaches

8.4 Evaluation

We discuss the results of our attack in terms of segmentation and recognition using the same method included in the previous chapters. Our evaluation in this chapter was simply based on an examination of 300 random samples of this CAPTCHA from the Kund's CAPTCHA website [100]. We obtained the samples through an automated screen capture designed using C# between January and February 2015. We used 100 of the samples in designing, training and processing our attack. We saved the other 200 samples locally to be used in the final examination of our attacks. We did not include any sample used during the design of the attack in the evaluation process of our attack. We targeted this CAPTCHA

using two approaches. The first attacking approach included all segmentation and recognition approaches, and the second approach did not include all approaches except the automatic segmentor recognition approach.

8.5 Results

Success rate. We firstly tested our attack against a sample set of 100 of the CAPTCHA challenges used for training purposes, and it achieved a success rate of 59% in recognising them, after it achieved a 66% success rate in segmenting them. After that, we tested our attack against 200 random samples from the Kund's CAPTCHA. The results showed the attack achieving a success rate of 75.5% (151 out of 200) in segmenting those samples of this CAPTCHA, and a success rate of 38% (76 out of 200) in recognising them. Those results included an accuracy rate of 40.94489% in recognising 127 samples segmented successfully without the automatic segmentor (52 samples out of 63.5% of the overall 200 samples). It also included an accuracy rate of 33.33% in recognising 73 samples (36.5% of the overall 200 samples) analysed using automatic segmentor.

The use of all methods in our attack is essential to achieving the success and accuracy rates of our attack. The tracking/segmentation attack should be used on at least 120 frames (extracted from each challenge using screen-capture at rate of 30 FPS) to achieve the least possible success rate in segmentation attacks without the use of an automatic segmentor. Our examination showed that an increase in the number of frames could also increase the success rate in recognising the text without the use of an automatic segmentor. The average rate of the increase per 30 frames was 1.33% with a maximum success rate of 30% recorded in segmenting and recognising the samples of this CAPTCHA, after we increased the number of frames to 210. The increase in the number of frames aided the automatic segmentor slightly in segmenting the connected characters, as the overall success rate of our attack increased to 43%, after we increased the number of frames to 270. The average rate of the increase per 30 frames was 1%. Figure 8.13 shows more detailed results of our examination.

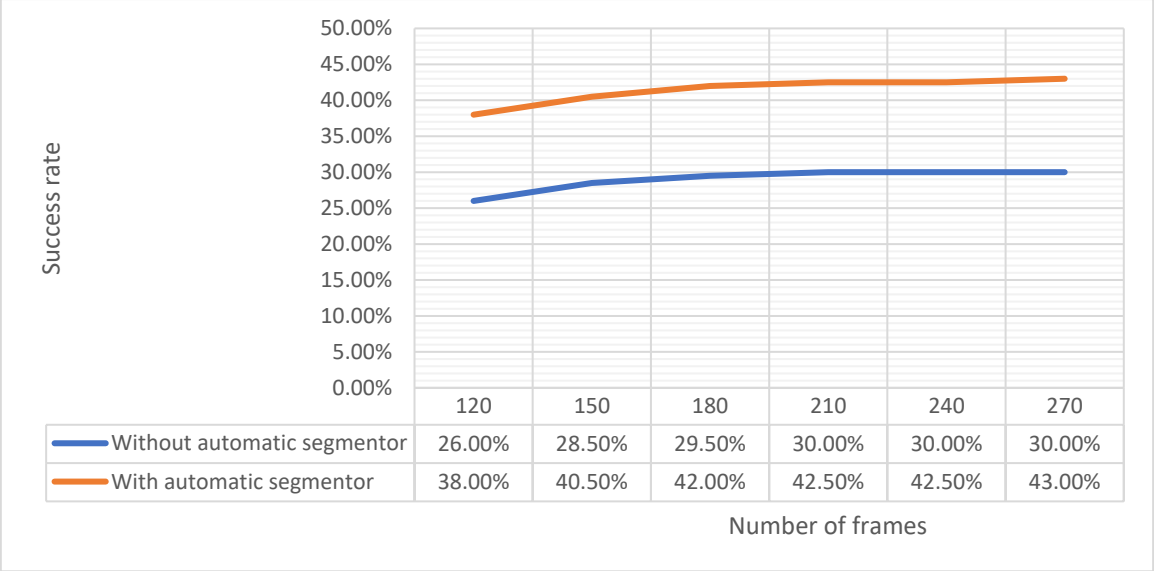


Figure 8.13. The success rate in comparison with number of frames used in the attack

We also identified the following failures in our attack:

Failure 1: Distinguishing the particles of characters from noises, when they share the same behaviour. This failure happens in CAPTCHA particles of characters and the noises when they have the same intensity and animation behaviour that makes the particles of the character hard to differentiate from the noises through the steps included in our attack. This can lead the attack to identify parts of the character as noises instead of the actual noises in the CAPTCHA. Figure 8.14 shows an example where the attack failed to distinguish the particles of character "M" from the other noises around it.

Figure 8.14. Error in distinguishing the particles of the characters and noises

A possible improvement in our attack to counter this kind of problem is to use another approach that draws lines that connect particles through their edges. This approach then fills each empty area within those connected particles to determine the places of the characters in each frame.

Failure 2: Extracting the segment of the characters that keep moving through the edge. This common failure is caused when the characters are not displayed in a full image in any frame that complicates the ability of the segmentation attack to provide the recognition engine with a recognisable image of the characters. Figure 8.15 shows an example of this CAPTCHA in which the character "D" keeps moving through the upper edge of an animated challenge

Figure 8.15. An example shows a character D moving through the upper edge

This can only be countered through the recognition engine itself by improving its ability to recognise those characters

Figure 8.16 shows the rate of failure in our attack against the Kund's CAPTCHA.

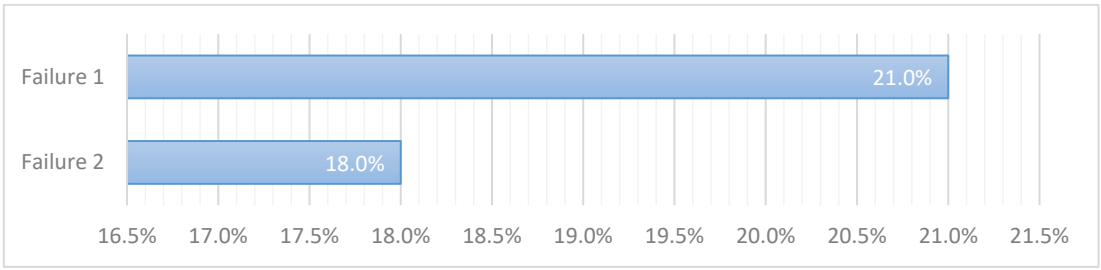


Figure 8.16. Failure rates in our attack against the Kund's CAPTCHA

Attack speed. Our CAPTCHA attack was implemented using JAVA, MATLAB, and C# programming languages and tested on a computer with an Intel(R) Core(TM) i7-4700MQ CPU 2.40GHz and 8.00 GB ram. We have tested our attack on all the samples in the test sets, including 200 samples. The result shows that the attack took an average of 97733.0275 milliseconds (standard deviation of 3808.5446 milliseconds) in both segmentation and recognition steps against each animated challenge of this CAPTCHA without the automatic segmentor. It also took an average of 150923.685 milliseconds (standard deviation of 66832.9974 milliseconds) against each animated challenge of this CAPTCHA with the automatic segmentor. The table 8.1. includes detailed results of our attack.

Table 8.1. The speed of the attack against the Kund's CAPTCHA in milliseconds

	Average time	Lowest time	Longest time
Without automatic segmentor	97733.0275 ms	89621 ms	108706 ms
With automatic segmentor	150923.685 ms	90384 ms	279928 ms

8.6 Defence

Although the mechanism has potential for the future security of the animated text-based CAPTCHA, it still currently suffers vulnerability problems against the tracking attacks. The main problem stems from the way it displays particles of the characters through the animated frames. This makes those particles distinguishable in most cases from the other noisy lines that represent the background. This problem can also simplify the reconnection of the particles through the animated frames.

However, these problems can be still avoided in future designs of the CAPTCHA schemes by adding a number of new features. These features could include animated methods used to distort the particles of characters so they can prevent the attacks from detecting those particles and connecting them. An improvement to the segmentation resistance mechanism is also important, as the current mechanism is still vulnerable to the segmentation attack. For example, we could implement a method that prevents the connected characters from moving separately, as those movements can help the attacks in segmenting the characters. We could also improve the resistance of this CAPTCHA by displaying the characters using multiple font sizes in every challenge.

8.7 Summary

We have exploited the vulnerability problem of the "noisy lines" mechanism implemented in the Kund's CAPTCHA through an automated attack that achieved a success rate of 38% in both the segmentation and recognition steps of our attack. This attack tracks the particles of the characters and noises around them in order to differentiate between them. It then removes the noisy lines from the animated frames and analyses the particles of characters to identify the particles that belong to a number of connected characters in each frame and segment them through the tracking method and the automatic segmentor. The results were used to discuss the vulnerability problems and to find the proper solution to protecting the future build of the CAPTCHA against current tracking attacks.

Chapter 9. Yahoo CAPTCHA

In this chapter, we have developed an attack against the most recent commercial animated text-based CAPTCHA created by Yahoo to be used on their websites. This CAPTCHA uses an animated mechanism that includes animated characters in two forms. The first form is the main challenge object that includes a number of animated characters, which move in a random circle through the animated frames. The second form consists of the noisy objects that move horizontally in one direction. Both forms share the same font size and animated rotation method used to distort the animated characters. Still, they are different in their font types, rotation angle, movement and rotations speeds. Our attack against the Yahoo animated CAPTCHA achieved a 78% success rate after we tested it against 200 samples of this CAPTCHA. The attack proved the weaknesses of this animated scheme by showing the vulnerability of its security mechanism to tracking attacks.

9.1 Introduction

In this chapter, we present an attack against the "noisy characters" mechanism implemented by the most recent animated text-based CAPTCHA created by Yahoo to be used on its websites. This animated text-based CAPTCHA includes text challenges displayed as a group of animated characters that move together in a looped direction up and down. Each of these characters can rotate in opposite directions and move horizontally within small vertical columns between two other characters on its side. This CAPTCHA also includes a number of noisy characters that move horizontally together in one direction from left to right. The noisy characters also include features used in the main characters such as rotation. However, these features are random in their speed and direction, with start and end times unlike the main characters. The main purpose of the "noisy characters" is to confuse the computer between the characters of the main text and the noisy characters. Yet, those two groups of characters are animated using two different methods, so the human can use his ability to identify the characters of the main text through their behaviour.

However, our attack has shown that those features can be still broken, as it achieved a success rate of 78% in segmenting and recognising the CAPTCHA challenges. This attack included tracking methods that locate the main text by detecting and removing the noises around it. It also includes other methods that segment the text using common features such as CL and PDM. The results of this examination proved the high vulnerability of the

tracking-resistance mechanisms to our tracking techniques. The success rate of our attack also showed the efficacy of solving the challenges, as this success rate is very close to human accuracy in solving the challenges. The attack speed recorded in our latest examination reached an average of 13.8547 seconds, which is also very close to the average time needed by a human in solving the challenges.

The sections in this paper are organised as follows: section 9.2 provides an overview of the targeted CAPTCHA scheme. Section 9.3 describes our attack against this scheme. Section 9.4 outlines the evaluation followed by the results in section 9.5. Section 9.6 discusses the defence against our attack, followed by the summary in section 9.7.

9.2 The characteristics of the Yahoo CAPTCHA

This CAPTCHA is an animated CAPTCHA that contains two sets of animated characters. The first set of these characters is the main characters that keep moving in a loop (up and down), while the second set are noises that keep moving horizontally. The key characteristics of this CAPTCHA can be represented as:

- Both the characters of the main text and the noisy characters are displayed using the black colour in front of a white background.
- The characters in each of the two sets are displayed using the same font, which is different from the font used in the other set.
- The main text contains between five and eight characters in each challenge.
- The main characters are displayed in one line that moves up and down in the middle of the image. This line may also move horizontally while it moves up and down.
- Few characters in the main set can be rotated between two angles or move horizontally between the two characters within the text.
- The noisy characters are divided into two sets of characters that are displayed within different lines in the upper and lower halves of the animated image.
- The noisy characters included in each set could move and rotate in different angles.
- All the noisy characters keep moving horizontally from left to right.

- Each of the noisy characters is animated, using either of the two approaches that rotates them slowly or changes the rotation angle randomly between sequences of two frames.

Figure 9.1 shows an example of this CAPTCHA.

Figure 9.1. Yahoo CAPTCHA

9.3 Our attack against the Yahoo CAPTCHA

This attack includes two main steps:

- Tracking/segmentation attack: this step includes method to clean the animated text-based CAPTCHA from noises, and segment the text.

- Recognition attack: this step uses the recognition engine to recognise the segmented text.

9.3.1 *Tracking/Segmentation attack*

The tracking/Segmentation attack includes the following steps:

- Select the noise characters: this step includes methods that identify the noise characters based on their behaviour.

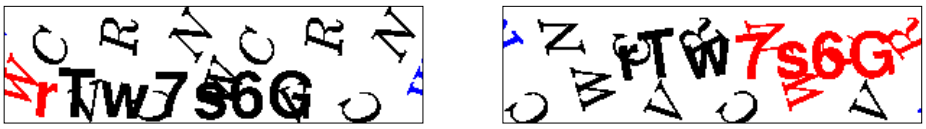
- Tracking and removing the noise characters: this step includes methods that track noisy characters and removes them from the background around the main text.

- Segmenting the main characters: this step includes methods that segment the main text.

Select the noisy characters. In this step, we track the noisy characters using their horizontal movement and rotation speed that could be faster than the main characters.

Select the noisy characters that move through the horizontal edges. In this step, we use a method to detect the noisy characters that keep moving horizontally from left to right. This method scans the horizontal edges of the animated image to locate character components

attached to them through its foreground colour. It then uses an algorithm to check if any of those components represent a piece of the connected characters, as one of those characters could be a part of the main text. This algorithm scans each component by selecting its foreground pixels through a colour filling algorithm. It then uses the scanning results to determine the total surface area, width and height of the component. Then, it selects only the component with a limited total surface area, width and height that do not exceed the surface area, width and height of the largest character "W" for the next stage. Figure 9.2 shows two examples of this method. The first example shows the method marking a character component that has a height that exceeds the largest height of normal characters in the left side of figure 9.2 (A). The second example shows the method marking a component that has a width that exceeds the largest width of normal characters in the right side of figure 9.2 (B).



(A) **(B)**

Figure 9.2. Select noisy characters through the edges

Select the noisy characters that move or rotate randomly. In this step, we used methods to identify the noisy characters that change their positions and rotation angles, so we can prevent obstacles in tracking their movements during the later stages.

The first method in this step scans each frame to locate unmarked objects in the previous step through their foreground colour. It then determines the coordination of the edges and other foreground pixel points within those objects using colour-filling algorithm. We also use the Harris detection algorithm to identify the position of the corners within the edges of the characters. This method helps speed up the comparison process between the animated characters through the shape of their edges alongside their surface areas, which we measured in the previous step.

After that, we use another method that compares the findings of the first algorithm in one frame with the findings of the previous and next frames. This method determines the edges and foreground pixels that are not located within radius of D of the identified edges in the previous or next frames. We could describe D as the maximum distance that the edges of

the main characters can move between two frames. We calculate D based on the maximum rotation and movement speed for the main characters given in pixels per frame through the following equation:

$$D = \sqrt{{S_x}^2 + {S_y}^2} + R \sin A$$

Where S_x and S_y are the maximum movement speed of the main character over x-axis and y-axis given in pixel per frame. Whereas, R is the maximum radius of the circle around the character, and A is the maximum rotation angle per frame. We could set the default value of D in our algorithm as six based of our findings after we run our equation against a few samples of this CAPTCHA scheme. We could use a method that adds stroke layers that have a thickness of D around the identified edges of the previous and next frames to the analysed one. We then join those stroke layers into one mapping image that shows the range where the objects moving at a limited speed are located within. We could determine this range to be within coordinated points where the stroke layers of both the previous and next frames meet. We use this mapping image to mark the foreground pixels that are located outside the range.

We then use a method that determines the total surface area of each object containing marked pixels in its body. This method also compares this object with the other nearest objects in the previous and next frames through the shape of their edges. It then marks the object as a noisy character and removes it, if this object matches the two following conditions:

- The shape of the object matches any nearest objects in the previous or next frames
- The surface area of the objects is less than the surface area of the largest character "W"

The other objects that do not match these conditions are analysed using a different method.

This method determines the foreground pixels located outside the marked range of pixel points in the mapping image between every two pairs of marked edges as noisy pixels. After that, it removes them and then determines the surface area of every group of foreground pixels that are left after the removal of the noisy pixels. We then remove the

group of connected foreground pixels that do not reach the minimum surface area of a main character, while we keep all the other groups that reach or exceed this surface area.

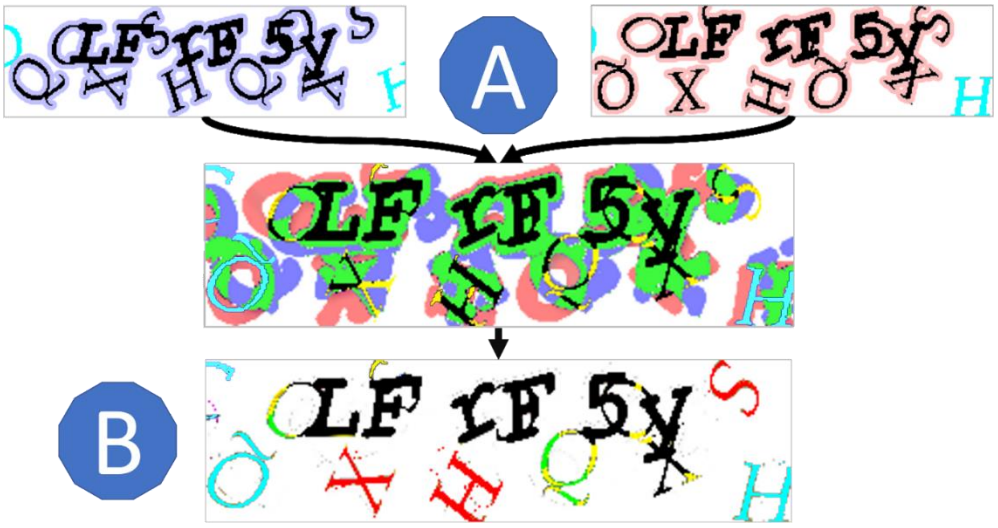


Figure 9.3. Select the noisy characters that have high movement and rotation speed

Figure 9.3 (A) shows how we created the map that shows the locations where the foreground pixels in the analysed frame should be located. Figure 9.3 (B) shows how we used the map in the later stages and identified four objects as noisy objects due to their match with a near component (marked in red). Figure 9.3 (C) shows how we analysed the rest of the objects. We could describe the divided parts of those objects in the figure as follows:

- The yellow parts: the marked parts of the objects, which we removed, before we analysed the other parts of the objects.

- The green parts: we marked these objects as noises and removed them due to their small area surface alongside the yellow parts of the objects.

- The black parts: we identified them as possible parts of the main text, to analyse in the second stage of our attack.

Track and remove the noisy characters. This step includes a few approaches to track the identified noise characters, and remove them. These approaches are:

The first approach works through a method that firstly removes all identified objects as noisy objects. After that, it analyses the previous and next frames to select other character

components in those two frames that are positioned completely or partially within the areas of the removed components. It then compares those components with removed objects. If the results of the comparisons show that any objects have the same shape as the removed noisy character near it, we determine this object to be a noisy character and remove it. Otherwise, we remove only the foreground pixels that have a distance of more than D from the nearest object in the previous and next frames. We also removed the segmented objects that did not reach the lowest surface area of the main character. We could run this approach through the animated frames in many rounds and in two directions forwards (from the first frame to the last frame) and backwards (from the last frame to the first frame). For example, Figure 9.4 (A) shows several characters selected in a previous step, such as "X" and "H" which are removed through the current approach. Figure 9.4 (B) also shows the results after running the tracking method against the animated frame in which it selected the two noisy characters "N" and "X", before it removed them in Figure 9.4 (C).

We could also remove the other noisy characters without the need to use the tracking systems, as we know that the noises move in one direction at a constant speed. This method selects the noisy characters that move steadily and at a constant speed in one horizontal direction. After that, it calculates the movement speed of those characters per frame.

It then uses the results to create an equation for every noisy character identified in the previous step. We use this equation to determine the position of the noisy characters in each frame. We also compare the original image of the noisy character and the image of each object located by the equation by using the same comparison method as in the first step. The equation used in this approach is represented as follows:

$$f_{N,I}(i) = \begin{cases} (I\ M) + \big((S\ N) - E_i\big)\ if (N \geq R) \\ (I\ M) + \big((S\ N) + M - E_i\big)\ if (N < R) \end{cases}$$

In which S is the movement speed for all the noisy character components over x-axis per frame. N is the analysed frame number. E_i is the position of the character in the first frame. R is the frame number where the animated character starts to pass through the left edge of the image, and I is the current round that the character reached through our calculation. Meanwhile, we can identify M as the highest position over x-axis the character i could go for one round through all the animated frames meaning we can calculate through the equation:

$$M = F * S$$

Where F is the number of frames in the animated image and S is the character speed (as we mentioned before). The method used in this step removes the noisy components that do not exceed the maximum pixel size, width or height of the character "W". Figure 9.4 (C) shows an example of this method in which it located "X" and "D" and matched them with the other characters that appear in the left image in figure 9.4 (A) after using our equation.

The remainder of the noise characters around the main characters are removed through a simple tracking method that identifies the whole body of the main text that keeps moving within the animated image. This method identifies the position of the text line that keeps appearing and moving within the animated frames. This method firstly selects the edges of the animated object that are placed within one horizontal column with a constant height based on the font and font size of the animated text in the image. The text in this horizontal column moves up and down at a steady vertical-speed of S_y between two horizontal lines of Y_u and Y_d. We could describe Y_u as a horizontal line located below the upper edge in the upper halves of the animated image. We could also describe Y_d as another horizontal line located at a distance greater than half of the image height $(Height_{img}/2)$ above the lower edge in the lower halves of the animated image. We then scan the moving objects in horizontal columns in every animated frame to determine their horizontal speed. We calculate this speed by determining the horizontal distance between similar corners that have the same horizontal distance $D_x(F_a \rightarrow F_b)$ in each two sequences of frames. We then extract the animated objects through a method that draws a rectangular sliding-screen to catch the text that includes those corners in each frame. The method then scans the horizontal and vertical sides of the text within the rectangular sliding screens to locate and remove the noisy characters that move in and out of this text. After that, it uses the results of the scan to correct the heights and widths of the sliding screen to be the same in each animated frame. It then extracts the text in those sliding screens into another set of frames. Each frame in this new set shows only the main characters animated inside horizontal columns. Figure 9.4 (D) shows an example of the animated frames after we apply our current approach against them. Figure 9.4 (E) shows another example of newly extracted frames after we finish removing the noises around the characters.

We then use another approach against the newly generated frames to clean the noises between the characters. This approach uses a pixel delay mapping image (PDM) created originally by Vu Duc Nguyen et al.[92], which shows the lengths of time each pixel's colour, distinct from the background, appears in each pixel area in the animated image. We use this map to identify the pixel's area where its colour is distinct from the background for a very short time that is less than a time calculated through the equation:

$$T = \left(\sum_{x=0}^{w} \sum_{y=0}^{h} f(x,y) \right) \Big/ 2wh$$

in which w and h are the width and height of the animated image, and $f(x,y)$ is the length of time the distinct colour from the background takes to appear in the coordination. Figure 9.4 (F) shows an example of this approach where the image on the left in the PDM map and the image on the right is the one newly animated frame after we apply the final cleaning approach against it.

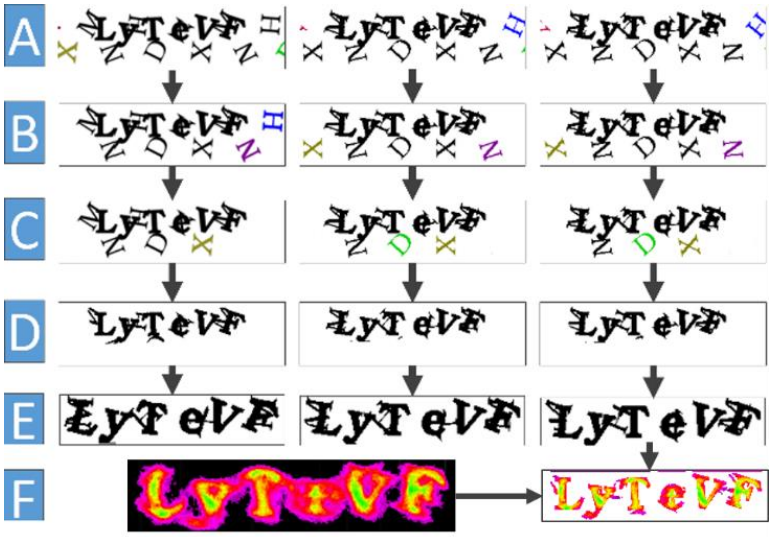


Figure 9.4. Tracking and removing the noisy characters around the main characters

Segmenting the main characters. In this step, we create a new mapping image that combines two mapping features. The first is a PDM map which we used in our attack against the KillBot CAPTCHA and CAPTCHANIM. The second feature shows the vertical columns where the main characters appear to be separated vertically for a specific length of time through the animated frames. We use both features in the map to identify vertical

columns where characters of the text are located. We could describe those positions of the vertical columns using the two mapping features as follows:

- The position where foreground pixels are displayed in a PDM map for longer than the average time through the animated frames.
- The position where the vertical segmented areas appear for less than the average time using the second mapping feature.

Our segmentation method uses both mapping features alongside a vertical histogram algorithm that segments each two pairs of characters by drawing a vertical line between them. This method uses the vertical histogram algorithm to locate and segment the character components vertically separated in each frame.

After that, it selects the segmented components that match the maximum widths of the animated characters marked in the map created by the previous approach. It then segments those components through a simple algorithm that identifies the vertical columns where the characters are separated. This algorithm uses both two mapping features to identify the weakest vertical column between the two pairs of characters in the component. It then segments the two pairs by drawing another vertical line between them.

Figure 9.5. The segmentation step

Figure 9.5 shows how all the characters are segmented using the first segmentation attacking approach except “d” and “f” which are segmented using the second approach.

9.3.2 *Recognition attack*

In this step, we used the Convolutional Neural Network (CNN) [110] engine to recognise the extracted characters. We trained this engine to recognise characters inside the CAPTCHA

images by creating a training set of 17304 sample images of characters (412 sample images for each character) and testing sets of 8350 sample images of characters. After that, we used two datasets to train our engine 50 times and the result was the following:

- The engine achieved a 0.1% error rate in training after it trained 17291 sample images of characters from the training dataset successfully.
- It also achieved a 0.1% error rate in testing after it recognised 8341 sample images of characters from the testing dataset successfully.

Figure 9.6 shows how the training and testing accuracy rates were improved during the training task.

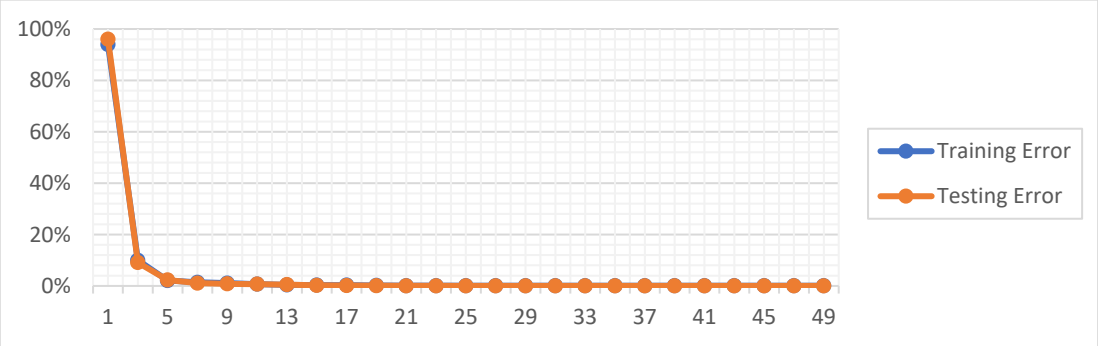


Figure 9.6. Training and testing accuracy rate during the training of the CAPTCHA

We then determined the score number as the most recognised character using a calculation based on the confidence rate recorded for every character. The attack then calculates the average of the confidence rates for each pixel point from the images used to generate the confidence rates for each character through the following algorithm:

$$f(n) = \frac{(\sum_{i=0}^{a} A_n[i]) + (\sum_{i=0}^{b} 10B_n[i]) + (\sum_{i=0}^{c} 100C_n[i])}{(a + 10b + 100c)}$$

Where the A is the set of confidence rates of character n for the image where the character achieved a confidence rate that is lower than other characters. The B and C are the sets of confidence rates of the same character for the images where the character achieved a confidence rate that is the highest between all characters. However, the confidence rates included in C are higher than 75%, unlike the confidence rates included in B. We then chose the character that achieved the highest average rate as the right answer.

9.4 Evaluation

We discuss the results of our attack in terms of segmentation and recognition using the same method used in previous chapters. Our evaluation in this chapter was simply based on an examination of 300 random samples of this CAPTCHA. We obtained some of the samples between June and July 2013 from the Yahoo website[117]. We used 100 of these samples in designing and training our attack and saved the other 200 samples locally to be used as text set in the final examination of our attacks. We did not include any sample used during the design of the attack in the evaluation process of our attack.

9.5 Results

Success rate. We firstly tested our attack against a sample set of 100 of the CAPTCHA challenges used for training purposes, and it achieved a success rate of 95% in recognising them, after it achieved a 100% success rate in segmenting them. After that, we tested our attack on 200 random test samples of the Yahoo CAPTCHA, and it achieved an overall success rate of 78% (156 out of 200) segmenting and recognising the samples of this CAPTCHA. Those results include a success rate of 83.5% (167 out of 200) in segmenting the samples of this CAPTCHA, and an accuracy rate of 93.41% (156 out of 167) in recognising the segmented samples.

The use of tracking methods in scanning all the animated frames is essential to achieving the lowest success and accuracy rates of our attack. The lowest round of this scan should be four (two times forwards and two times backwards). Our examination also showed that an increase in the scanning rounds does not increase the success or accuracy rates. As result, we concluded that the use of additional computational effort does not increase the success rate of our attack on this CAPTCHA scheme.

We also identified the following failures in our attack:

Failure 1: finding the segmentation spot between the animated characters. This failure is mostly due to limitations in our attacking algorithm when it encounters characters that keep connecting to each other through the animated frames. Such a limitation causes the attack to fail in identifying the correct position to segment them. This failure also occurs when the attack fails to remove noises between two animated characters that are close to each other. This failure was the cause of all the segmentation errors (73% of the overall errors) in our

attack. Figure 9.7 shows an example of this failure where the attack failed to segment “G” and “e”.

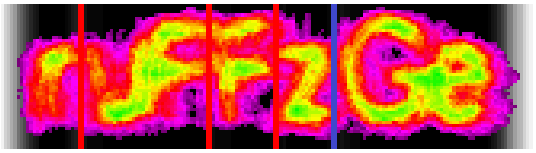


Figure 9.7. Failure in segmenting characters

We could counter this problem by implementing another mechanism that tracks the exact movements of two connected characters instead of finding any frame where they are segmented.

Failure 2: extracting a clean image of characters. This problem is caused by an error in the identification of the noisy characters between the main characters in cases when the main characters rotate and move separately in two horizontal directions. This problem can be more effective in cases where the noisy characters use a similar font to that used in the main text. Such a failure contributed to 27% of the overall errors in our attack. Preventing this failure is achieved by analysing multiple versions of the extracted images of the main characters through the animated frames. Figure 9.8 shows an example of characters extracted with noises around them due to the fault in cleaning the characters.

Figure 9.8. Failure to clean noises between the animated characters

Figure 9.9 shows the rate of failure in our attack against the Yahoo CAPTCHA.

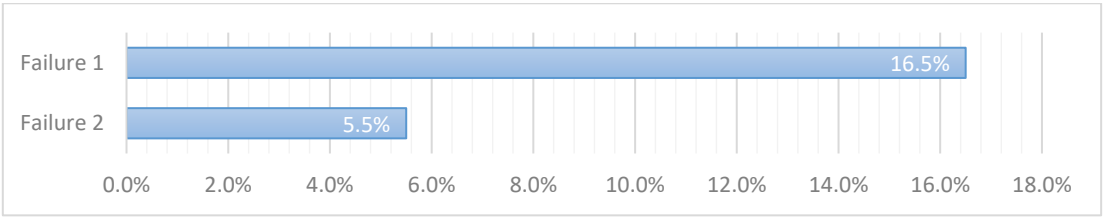


Figure 9.9. Failure rates in our attack against the Yahoo CAPTCHA

Attack speed. Our CAPTCHA attack was implemented using JAVA, MATLAB, and C# programming languages and tested on a computer with an Intel(R) Core(TM) i7-4700MQ CPU 2.40GHz and 8.00 GB ram. The result shows that the attack took an average of 13854.72 milliseconds to analyse each animated challenge of the Yahoo CAPTCHA with standard divination of 1871.504 milliseconds recoded between the timings of the attacks. The shortest attacking time was 12034 milliseconds, while the longest was 19166 milliseconds.

9.6 Defence

The current animated version of the Yahoo CAPTCHA is quite predictable, as it has too many features that make it highly vulnerable to attack. The most vulnerable features are the usability feature of the noisy characters that cause them to move in one horizontal direction from left to right. This feature assisted our attack to detect those characters within the horizontal edges of the animated image. The randomised movement behaviours of noisy characters also had a negative impact on the robustness of this CAPTCHA. The use of behaviours exclusively on the noisy characters caused them to be more detectable by the tracking attacks.

Most of the vulnerability problems in this CAPTCHA can be resolved by removing the unique animated behaviours of the noisy characters. This could prevent attacks from those behaviours detecting them. The CAPTCHA also needs to include other animated features to be used in both the main and noisy characters. For example, we could animate wrapping behaviour to prevent attacks on their images in the animated frame and include other features that decrease the visibility of the main text, such as the “emerging image”. These features could decrease the accuracy rate of the recognition engines in recognising the main text and prevent attacks from those behaviours that detected them

9.7 Summary

We have developed an attack against the animated version of the Yahoo CAPTCHAs. This attack uses tracking techniques in addition to the PDM mapping to detect the main characters and segment them. Our attack achieved 0% against this CAPTCHA, after it was tested against 10 samples of the CAPTCHA animated challenges. As a result, we have proved that the computer can pass approaches used in the current animated text-based CAPTCHA schemes in a way that could even beat the human in recognising the characters.

Therefore, a new mechanism that uses a stronger cracking algorithm needs to be designed for animated text-based CAPTCHAs instead of the current mechanisms.

Chapter 10. Discussing our examinations against CAPTCHAs

10.1 Introduction

We examined the robustness of animated text-based CAPTCHAs that relay in recognition-resistance mechanisms. We also examined animated CAPTCHAs that use tracking-resistance mechanisms. These mechanisms hide the details of the CAPTCHA challenges using noises that are spread through a series of frames. In addition, we examined standard text-based CAPTCHAs related to our research that used segmentation-resistance mechanisms. Some of the most important CAPTCHA schemes discussed in our attacks were Yahoo, the Kund's CAPTCHA and the Xu's version of NuCAPTCHA alongside the animated recognition resistance methods such as the KillBot professional CAPTCHA and CAPTCHANIM.

Our thesis also includes research on the segmentation resistance mechanisms used in text-based CAPTCHA schemes. These mechanisms include the CCT mechanism of ReCAPTCHA, and the Wikipedia mechanism.

We implemented each of our examinations with attacking methodologies aimed at exploiting the vulnerability problems in those CAPTCHA schemes. We summarise those attacks and the results in table 10.1.

Table 10.1. Summary of our attacks against text-based CAPTCHA schemes: the broken criteria is explained in the latter paragraph

	Characteristics	attack	results	Broken*
Wikipedia CAPTCHA	• Non-animated segmentation resistance scheme • Uses grayscale image • Text contains two connected words • Distorts and connects the characters by using lines as clutter + empty holes	• Segmentation attack • Removing: lines + empty holes	Success: 70% A-Time: 17.67 ms	Yes

KillBot professional CAPTCHA	• Animated colourful scheme • Includes animated noises • Text contains five characters • Each character animated using one of four methods: vertical movement, scaling, blurring/fading and rotating	• Character extraction process	Success: 71% A-Time: 5161.82 ms	Yes
CAPTCHANIM	• Animated colourful scheme • Includes two varieties: o characters stay in one fixed location o characters move horizontally (L→R or R→L) • Two methodologies for character distortion: vertical scale + horizontal deformation methodology	• Character extraction process	Success: 85% A-Time: 64.41 ms	Yes
ReCAPTCHA	• Non-animated segmentation resistance scheme • Uses grayscale image • Text contains two segmented strings: o "string known to server": CCT + high distortion o "string unknown to server": contains less distorted + blurred/faded characters	• Segmentation attack: uses the automatic segmentor	Success: 43% A-Time: 4875.51 ms	Yes
Xu's version of NuCAPTCHA	• Animated tracking resistance scheme	• Two-steps tracking attack:	Success: 64% A-Time: 40569.7 ms	Yes

	uses “emerging image” mechanism • Text contains three characters uses animated CCT segmentation resistance: characters highly crowded + rotate individually	○ removes noisy background ○ tracks the main characters • Animated CCT segmentation attack		
Kund’s CAPTCHA	• Animated tracking resistance scheme uses “noisy lines” mechanism • Characters could rotate + move into random direction at random speed	• Two-steps tracking attack: ○ removes noisy background ○ tracks and segments the main characters • Optional: automatic segmentor	Success: AS 37% AS 26% A-Time: AS 97733 ms AS 150923 ms	Yes
Yahoo CAPTCHA	• Animated tracking resistance scheme uses “noisy characters” mechanism main characters: move up and down noisy characters: move horizontally	• Two-steps tracking attack: ○ remove noisy characters ○ track and segment the main characters	Success: 78% A-Time: 13854.7 ms	Yes

By applying our methodologies to exploit vulnerability problems in these CAPTCHA schemes, we have identified various security flaws in their tracking and segmentation resistance mechanisms. These security flaws aided us in breaking our targets with high success rates that were close to the human success rate. The results also passed the observations made by Bursztein et al. [59] in which the CAPTCHA scheme is deemed broken if it can be automatically solved more than 1% of the time within the first 100 schemes. We also considered the recorded speeds of our attacks to be practical, as they were within the duration needed by a human solve a CAPTCHA. Still, we can consider our attack against the Kund’s CAPTCHA as an exception, if we compare its speed with a human's speed in solving a CAPTCHA. However, this exception is still arguable in comparison with other attacks against simpler CAPTCHAs. For example, Xu’s attack against the NuCAPTCHA recorded an average time of 250 seconds, which is still slower than our attack.

However, we still do not claim that any of the segmentation and tracking resistance principles were overturned based on our results. Instead, we have discussed various improvements to the defence mechanism and which have helped us to learn lessons associated with each mechanism as described in previous chapters. Therefore, we have improved our understanding of how we can design better segmentation and tracking mechanisms.

In this chapter, we discuss the various lessons learned from our investigations of the CAPTCHA schemes including the robustness and usability issues found in their design and resistance mechanisms. We also discuss our design guidelines based on those lessons, which we will use to design a new animated text-based CAPTCHA scheme in the next chapter.

10.2 General lessons and security analysis of the examined schemes

In this section, we discuss the lessons learned from our examination of each defence mechanism adopted by our targeted schemes. We also discuss the implications for the security and usability of the CAPTCHA based on these lessons.

The NuCAPTCHA and the "emerging image" mechanism. The technique used in the Xu's version of NuCAPTCHA is designed to counter the tracking attacks that relay on frame-to-frame analyses. The main reason is that the CAPTCHA shows animated objects and the background using noisy black noises instead of using full and colourful images. This kind of technique hides the edges and silent features of the main objects in each frame that are used by the tracking attack mechanisms to segment and recognise those objects. Although the noises used to display the text is thicker than those used to display the noisy background, they remain undetectable in most of the attacks. The main reason is that the noisy points near the boundaries of the text keep changing their sizes and densities through the animated frames. This feature causes some of the boundary points to appear in a similar shape to the one used for the background.

However, we still proved that the attacks can break this mechanism despite those advantages. We also proved that this mechanism has many vulnerability problems in the current design, one of which is centred around the intensity issue. A cause of this issue is the limited distance between noises that is restricted those used to display the boundaries

of the text. This intensity issue aided our attack in removing most of the background noises that remained in the distance from the text.

The other noises around the characters are identifiable by the method that tracks the movement of the text. The main cause is the movement behaviour of the noises that are different than behaviour of noises related to text. We selected those noises through various tracking methods that followed the movement of the text. These methods could also be assisted by other methods that determine the position of the text based on its width and height and the number of thick particles used to display it. The main success of methods is still dependent on the specified movement behaviour of the text objects of NuCAPTCHA, in which the characters steadily move from left to right, as well as up and down. As a result, we could still present an unsolvable problem relating to the use of more complex methods in animating the challenge text.

We can also present another issue in the segmentation attacks against this CAPTCHA scheme which is centred on the tracking attack methods that detect the edges of text and which could fail against this scheme. The main cause of these problems is the difficulty in distinguishing the characters and the other close noises due to similarities in their shapes. This problem could also be caused by approaches that blur and hide few boundary points of the text. We could address this problem by using other approaches that determine the vertical columns where the text is located in each frame. Still, these approaches are also highly dependent on the font type and size of the text.

As a result, we learned that the current design of the “emerging image” mechanism is highly exploitable by tracking methods that identify the position of the animated text in every frame. This issue will remain, if the behaviour of the animated text is identifiable and distinguishable from the noisy background and could provide a clear insight into the robustness of the “emerging image” mechanism. It also shows how this robustness is dependent on the level of difference between the animated behaviours of the text and the noises around it. The unique movement of the animated text could also play a major role in the capability of those tracking attacks to break the CAPTCHA. Therefore, it is highly advisable to randomise the movement behaviours of the animated text, so it could prevent the attacks from predicting the position of the text in every frame.

Another lesson we learned from this attack is that the emerging technique can still improve the resistance of the animated CAPTCHA against segmentation attacks. This lesson is based on the current results of our segmentation attacks that failed to track the boundaries of the characters. This major failure lowered the level of vulnerability caused by animating the characters in the text individually towards the current segmentation attacks. It also showed the level of difficulty for those attacks to extract the animated character without showing the sides of any other characters beside it.

The Kund's CAPTCHA and the "noisy lines" mechanism. This CAPTCHA uses a similar technique to that used in the Xu's version of NuCAPTCHA and is centred on the use of noises to display the pattern of the main characters and background. It is also centred in the separation of noises through the animated frames that resist one to one frame attacking techniques. The types of noises used in this CAPTCHA could also increase its resistance against the attack used on CAPTCHA schemes that use the "emerging image" mechanism. These noises include similar sets of tiny lines that share the same length and size of the animated frames in every CAPTCHA animated image.

However, we could exploit and break this CAPTCHA through various attacking methods which distinguish the particles of the characters from most of the noises of the background through their intensity. The success of this method is dependent on the limited distance between the particles of characters that make them visible to the human. This exploitable behaviour is even detectable in the frames where 50% of the particles are invisible. We could still exploit this invisibility issue in which almost 80% of the particles (90% of the total number of particles) remained invisible for a maximum sequence of two frames. This limitation aided our attack in countering the invisibility issue by creating another sequence of frames. Each of the frames shows particles of characters that emerged from sequences of the three original frames combined.

These exploitable features also contributed towards providing the recognition engine with more recognisable images of the characters. We generated the images through a method that merges the particles that appeared in different positions in every frame. This contribution is more apparent in challenges where the alignments of the borders of the particles and characters are the same. The alignment states helped us to generate an image of the character that was very close to the original images. However, this kind of approach

could still present a problem when additional animated distortion is used against the particles of the animated CAPTCHA characters as the distortions could cause the particles to change their positions and rotation between every sequence of frames. As a result, they could increase the difficulty in generating a recognisable image of the characters. Still, this approach could also produce a high usability issue in which CAPTCHA characters becomes undetectable and unreadable by the human an issue is present, especially where characters are displayed in small fonts. This could also cause a severe robustness failure in our attack and lead to the human success rate in answering the challenges matching that of our computer program.

Nevertheless, our examination showed a difficulty in differentiating between the particles of the characters and the background when both share the same intensity and movement behaviour. This shared behaviour could play a role in encountering tracking attacks that target this scheme. This role is centred in slowing down those attacks by forcing them to run heavy approaches to find the particles of characters. These approaches include an analysis that aims to find every detail of the set of noisy lines and the empty areas between them in every CAPTCHA frame. However, they could still have usability issues that lower the human's ability to locate the particles of the character. These issues may cause humans to look at every set of noisy lines through the CAPTCHA frames to determine whether they belong to the animated character or the noisy background. Still, the ability of the computer program would not be matched with the human’s ability to solve the challenges. As such, the human eye would be able to catch those challenges more quickly than any program and such an analysis would need to analyse every particle in every frame to find the proper answer.

The main lesson learned is that the current form of noisy line mechanism could not resist attacks even with the use of the high distortion methods, the main cause being the features used to animate both the characters and noisy backgrounds. Based on our study, the use of these features in this form would not prevent the computer programs from matching the human success rate. Still, these vulnerability problems would not be enough to confirm that the “noisy lines” mechanism is entirely broken considering that there is scope for improvement in this design which could provide multiple solutions to the robustness issues in this CAPTCHA. Therefore, we regard this kind of mechanism as having the potential to make the CAPTCHA more resistant to current tracking attacks and this is mainly dependent

on the ability to generate usable and secure animated challenges. These challenges should help the human to distinguish between the characters and noises without the need to use a different type of noise for them. Such a challenge could lead to a more robust, or at least a more usable, scheme than the ones that use the "emerging image" mechanism.

However, this potential could still not be realised if the current usability features make the particles of the characters behave differently than the noisy backgrounds. Otherwise, it leads the attacks to break the CAPTCHA scheme that uses the mechanism through their animated distinguishable behaviours. We proved this point through our robustness examination against the Kund's CAPTCHA. This examination showed successful attempts in gathering the segments of the character alone without the noises around them to create a set of recognisable images of the character from every sequence of frames.

Yahoo CAPTCHA and the "noisy characters" mechanism. In principle, the use of a "noisy characters" mechanism can increase the resistance of the CAPTCHA against the tracking attacks that directly identify the text and track it through the animated frames. However, as implemented in the Yahoo CAPTCHA, this mechanism could also be exploited by attacking methods that distinguish the noises from the text through their behaviours. We have demonstrated the vulnerability of this mechanism in our examination of the Yahoo CAPTCHA. This examination includes an attack that succeeded in identifying the noisy characters through their movement from left to right and identified the main text through its movement in a random circle through the animated frames. This attack also distinguished the main texts from using the noisy characters in a few challenges by the difference in their movement and rotation speed. These included the challenges where the noisy characters change their position or rotation angle randomly through the animated frames.

This mechanism may also suffer from usability problems that could affect human accuracy in solving its challenges. These problems are linked to the ability of the human to differentiate between the main characters and the noisy characters. This problem occurs mainly when the principal and noisy characters are displayed using a similar font type and size. Figure 10.1 shows an example of those cases in which the difference between the main and noisy characters is their movement behaviour. The CAPTCHA in this case can easily

confuse the human and cause them to add together both the main and noisy characters in the answer box.

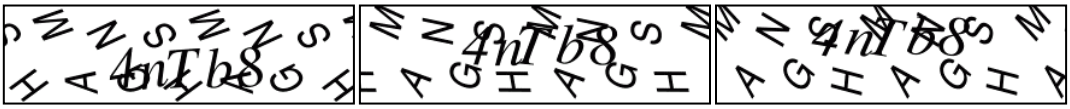


Figure 10.1. Confusing Yahoo CAPTCHA

As result, we have shown that the current designs of the "noisy characters" may only impact negatively on the balance between usability and security. These designs can only assist the computer program in beating the human to solving this CAPTCHA. Therefore, we recommend that future designs include other distortion mechanisms to avoid such a negative impact in those designs.

The CAPTCHANIM/ KillBot professional CAPTCHA and the "animated distortion" mechanism. The animated character mechanism used in the CAPTCHANIM and KillBot professional CAPTCHAs do not improve the security of those CAPTCHAs. Instead, they could lower the ability of humans in recognising them. We have shown this particular problem through our attacks that broke those CAPTCHAs with a very high success rate. This attack included approaches that analysed the animated characters to catch them when they are displayed completely in the middle of the frames. These approaches can also catch animated characters distorted through the animated approaches in their least distorted images. These animated approaches include complex ones such as the rotation approach and vulnerability problems can lead to those CAPTCHAs being less secure than the traditional, image-based CAPTCHAs.

However, the approaches used as part of the targeted mechanism could still slow our attacks, the main cause being the length of time needed to analyse frames from the animated challenge and to identify the frames that show the correct forms of the animated characters. Still, this does not mean that the human can do better, as the human also takes time to solve the animated challenge. This includes the time taken looking at the animated image to find the frames that display each character correctly. This length of time could be also increased, when a human misses those frames, causing them to wait until they appear again. In addition, a few animated challenges can be more difficult for the user to solve than the current automated attacks such as the confused rotated characters, which can be difficult

for the user to recognise. This issue is more apparent in the characters which rotate in different directions, while they appear as rotating towards their correct direction for a short length of time. However, we proved the ability of the automated attacks to assist the recognition engines in recognising those confusing characters by catching them when they are rotated in the correct direction.

As a result, we learned that the use of the animated mechanism against the individual characters could not assist in fooling the recognition attacks. Instead, they could lead to vulnerability problems against customised attacks, which can solve them better and faster than the human. We showed this issue in our attacks against CAPTCHANIM and the KillBot professional CAPTCHAs. This attack strengthens the need to adopt both segmentation and tracking resistance mechanisms to improve the resistance needed against the attack. This adaptation can be more effective than relying on the "animated distortion" mechanism that adds only the time dimension to the CAPTCHA schemes.

The ReCAPTCHA and the "crowding characters together" mechanism. In general, the use of this mechanism did not provide significant resistance against the attacks, which could still exploit them by identifying the characters from their shapes. For example, we could identify the characters "i" and "j" through the dot point above them, as we showed in our attacks against the ReCAPTCHA. The connected characters can be also exploitable in their connection areas by the segmentation attacks. One example of this is when the characters "d" and "h" are connected to each other. This connection can leave a sharp looped area with can be used by the attackers to segment those characters.

The distorted characters that could resist attacks identified through their shapes are still identifiable by machine leaning attacks such as the automatic segmentor. We have shown this issue in many cases during our attack on the latest version of the ReCAPTCHA that includes too many distortion methods in its designs. Still, we have shown some cases where the attack could lead to errors in recognising the connected characters. These include cases where the attack identified connected characters as different characters, due to the similarities between them. Nevertheless, the feature behind this problem could still fail to provide an option of distinguishing between the human and the computer. This failure is caused by human inability to avoid the same mistake as the computer in attempting to solve the same CAPTCHA challenge. For example, the computer could fail in distinguishing

between “rn” and “m” in the ReCAPTCHA challenges due to the high distortion mechanism. Humans also encounter these problems, especially non-native English speakers, as the words used in the ReCAPTCHA were originally selected from an English text.

As a result, the use of a CCT mechanism, alongside the distortion algorithms, is not enough to provide the necessary balance between the security and the usability of the CAPTCHA system. Instead, it can increase the level of human confusion each time it increases the level of security. This proves the need to include other mechanisms to empower the segmentation resistance of the CAPTCHA system, such as tracking resistance mechanisms.

The Wikipedia CAPTCHA and the “lines as clutter” mechanism. Relying on noises to improve the robustness of the CAPTCHA against segmentation and recognition attacks does not seem to be sufficient. We proved this point in our attack that removed noises around the characters and in their connection area with other characters. Our attack against the Wikipedia CAPTCHA in chapter 3 has clearly shown how to deal with this mechanism. This attack succeeded in identifying the noises through their low thickness around the characters and their connecting areas between the characters. This identification aided our attack in segmenting the characters that are connected through those noises. It also helped us to clean those noises from the area around the characters before we sent the characters to the recognition engine. As a result, our recognition engine achieved a very high success rate in recognising the CAPTCHA challenges and proved the vulnerability of this mechanism against the attacks.

It should be noted that the CAPTCHA uses a distortion mechanism that causes a few parts of the characters to be removed from the main image. It also uses the CCT mechanism to connect a limited number of characters in few CAPTCHA challenges. Both mechanisms could survive our simple attack against Wikipedia CAPTCHA as they need a more complex attacking method to deal with them. The survival of those mechanisms proves that they are more effective in defending the CAPTCHA against simple attacks than a “lines as clutter” mechanism. Noises generated by the mechanism around the characters were easily identified and removed from our CAPTCHA challenges, as we explained before.

10.3 On the design guideline of animated text schemes and associated mechanisms

In this section, we discuss the attributes found in the design of both the animated and standard text-based CAPTCHA schemes and their defence mechanisms. We also discuss the trade-offs between the robustness and usability of those CAPTCHAs, as well as the design guidelines which we can provide, based on our research into them.

The attributes of the text (main characters). The attributes of the main characters play a major role in defining the robustness and usability of the animated and non-animated text-based CAPTCHA schemes. We discuss this major attribute for both standard and animated text-based CAPTCHA in the following terms:

The size of character sets/text length. The text attributes that determine both the robustness and usability of the CAPTCHA are mostly defined in two terms. The first is the number of characters included in the character set, and the second is the text length (the number of characters used in each challenge). Robustness issues are mainly apparent in standard and animated schemes that use small text lengths and small sizes of character sets. The use of small text length can play a role in assisting the segmentation attacks, as the lower text length means a decreased chance of the attacks segmenting it wrongly. The small size of the character set could also play a role in helping the segmentation attacks as it will decrease the number of detectable patterns. In addition, it could also play an enhanced role in assisting the recognition attack, as the attacks would have to choose between fewer numbers of characters to solve challenges, which increases the recognition rate of those characters. The use of small text length and character set sizes in the animated schemes could also help the tracking attacks solve the challenges by simplifying the objective of the attacks in determining the position of the characters in each frame used in the animated challenges.

That does not necessarily mean that the use of a greater text length or character set size can be regarded as a solution. This point is proved by many researchers who found other problems that could cause the balance between usability and security in the text-based schemes to be lowered. These issues are present in the recent animated schemes such as the KillBot CAPTCHA, NuCAPTCHA and the Kund's CAPTCHA. One of the major issues is the text length in animated challenges, which played a major role in breaking the schemes that contain them. For example, the fixed number of characters used in each challenge

category in the Kund's CAPTCHA helped us to identify the maximum number of characters needed to be tracked and segmented in each challenge. On the other hand, the fixed number of characters still plays an important role in providing the usability of the character. This fixed number helps the user identify the correct number of characters needed inside the CAPTCHA challenges. It can also reduce the chance of misspelling the connected characters. This usability advantage is more present in the animated scheme where the animated characters are highly connected and distorted, such as Xu's NuCAPTCHA.

The choice of characters. Another issue related to the text attribute is the choice of the character included in the character set. This issue is linked to major successes of current attacks on text-based CAPTCHAs. These successes were mainly dependent on a specified pattern shape, although the patterns could significantly increase the recognition rates of challenges by the human. For example, we showed how too many characters were exploited through a certain pattern in their shape. For example, the looped characters such as "O" and "D" in the ReCAPTCHA are easily detected by the shape of the loop pattern inside them. The characters with points above them can also be regarded as prime examples. These characters can be easily identified through the shape of their points, such as "i" and "j" in the ReCAPTCHA scheme.

We have also identified another issue with the character set included in the current CAPTCHA scheme that is related to those sets limited to capital letters only. The main cause of this issue is the common ratio of width to height in most capital letters, such as "K", "P", "T", "N", "B" and "R", in which they share the same font type and size. This issue could aid attacks in identifying the position of other characters of shared height data, even in cases where those characters are rotated. However, these characters can still play a major role in the usability of the CAPTCHA scheme. This role is more apparent in animated schemes that connect them together through animated mechanisms. The usability of these schemes is defined by the shared height of the animated capital letters. This shared height would make animated letters more recognisable to the human, when they are connected together.

The segmentation resistance attributes. In general, the main weaknesses of the current segmentation resistance mechanisms are centred on the exploitable shape of the characters

and their connection areas. These exploitable shapes are mainly present in the standard text-based CAPTCHA schemes that use the CCT mechanism. Those schemes include the most recent ones that failed to hide their exploitable shapes through distortion mechanisms such as ReCAPTCHA. Our recent research on these CAPTCHA schemes showed that they are highly exploitable by attacks that combine novel attacking methods with the automatic segmentor approach.

The use of animated approaches as a part of the CCT mechanisms may present an additional solution to hide exploitable shapes. However, many animated approaches still present another vulnerability problem such as the segmentation-resistance approach deployed by the NuCAPTCHA. This approach overlaps with a number of characters and animates them individually. We could present their problems with attacks that identify the connection areas between the characters animated through their movements, such as our attack against Xu's NuCAPTCHA. Nevertheless, we could still overcome these problems with methods that animate the whole connected characters through the CCT mechanisms. We could also include other animated methods that aim to partially hide the characters through the animated frames.

Other noisy objects could still also play a role in strengthening the segmentation-resistance of the CAPTCHAs. For example, they could act as false patterns of the text to confuse the segmentation attacks between them and that text. These objects mainly need to match the pattern of the characters, otherwise they would be highly exploitable through the automated attacks that would be able to remove them from the areas around the character (e.g. the Wikipedia CAPTCHA). These objects also need to match the animation of the characters in case they are included in the animated CAPTCHA schemes. We could do that by randomising the movement of the patterns of the character and the noisy elements around them.

The use of emerging noises against tracking attacks. The use of the “emerging image” mechanism and its equivalent mechanism (noisy lines) should prevent attackers from gathering pieces of a character segment from the animated frame. Such a feature should not make the segment of the characters distinguishable from the noisy background by a computer program, otherwise it would cause it to extract segments of the characters and

separate them from the noisy background around them. That would simplify gathering pieces of the animated characters and lead to these pieces being recognisable by attacks.

Our research explored many exploitable features. We can identify those features through the intensity, movement direction and speed of the noises that represent the main text, especially our research against the Xu's version of the NuCAPTCHA and the Kund's CAPTCHA. Nevertheless, these problems can still be countered by decreasing the intensity of the text segments and matching their animation behaviours to the other noises around them. In addition, we may need to include other animated distortion methods so that those used against segments of the character in one frame cannot be matched with another segment of the same characters in another frame.

The use of the noisy characters against tracking attacks. The use of animated characters as noises has a subtle effect on both security and usability. For example, the "noisy characters" mechanism has failed to hide the exploitable behaviour of both the main characters and the noisy characters around them. This failure leads our attack to identify and select the main characters, after it had removed the noisy characters around them. These exploitable behaviours are essential to the recognisability and readability of the text by the human. Such distinguishable behaviours need to exist regardless of their format (upper case, lower case and numbers). In addition, as we discussed in 10.1, it seems that in the presence of these behaviours the system could still cause usability issues.

We have explored techniques to exploit the behaviour of the main and noisy characters associated with segmentation technologies. We have also discussed the possible defences against them. However, it still seems to be an open problem to design a CAPTCHA scheme that is secure and usable. We can prove that through our targeted scheme that still depended on one unique behaviour for the animated noises. This can be overcome by using different animations for the noises in each challenge. These different animations should increase the confusion of the attacks in distinguishing the noisy characters from the main characters.

The use of other animated distortions. As we discussed before, the animated distortion would not improve the resistance of the segmented characters against attacks. As we have shown, these attacks can still catch the animated character in their full and most correct form in the animated images. As result, they could still beat the human in solving the animated CAPTCHA schemes that include the animated characters. The results of our

examinations against the KillBot and the CAPTCHANIM are prime examples of how the computer program can catch even the confusing rotated characters in their correct form, including “p”, “q”, “d” and “b”.

Nevertheless, we still think that animated distortion methods could increase the robustness of the CAPTCHAs if they are used against the whole text alongside segmentation-resistance and tracking-resistance mechanisms. Such a combination could make it harder to present a visible image of the text but, even so, could still not prevent the attacks from generating the full image of the animated text, even from a sequence of emerging frames. This issue could aid the attack to exploit these mechanisms, as it could still segment and recognise the full images of the text through a deep-learning attack. Therefore, an implementation of a method that prevent the attacks from generating these images is required to improve the robustness of the animated challenges.

Chapter 11. The design of the new animated text CAPTCHA

In this chapter, we discuss the design of the new CAPTCHA scheme based on the results of our attacks on the previous text-based CAPTCHA schemes. This CAPTCHA scheme includes animated text that moves randomly through the animated frames and other background objects that resemble the noises in our CAPTCHA challenges. The CAPTCHA displays both the text and background using the "emerging image" mechanism. We tested this CAPTCHA scheme in terms of its robustness and usability to develop an understanding of the main problems of the animated text-based CAPTCHA scheme. It also helped us to improve them with an examination of its methodologies not included in our previous scheme.

11.1 Introduction

Through our studies, we have shown how too many animated and standard text-based CAPTCHA schemes failed to repel the automated attacks. Our research led us to identify various vulnerability problems in our targeted schemes and helped us to exploit our targets with tracking and segmentation attacks. This research also helped us to determine proper solutions to remove the exploitable vulnerability problems from the targeted CAPTCHA schemes and increase their robustness against the segmentation and tracking approaches, as we discussed in chapter 10.

The results of our examinations of the current CAPTCHA schemes have helped us to produce a design guideline which we have used to design a new animated text-based CAPTCHA (more details included in section 10.3). This CAPTCHA scheme includes distortion methods that are based on the "emerging image" mechanism. Each of the methods show the objects in the animated image in the form of animated segments that move in front of a white background. The animated objects displayed in the CAPTCHA challenges in this scheme include a main text object which includes animated characters that humans need to write in a specific box to solve the CAPTCHA. They also include other animated objects built from noisy images. All these objects appear and move randomly through the animated frames. Each animated object then moves in random directions at random speeds and keeps changing its movement, direction and speed through the animated frames.

The characters in the main object are animated by a number of distortion approaches including character overlapping and individual character rotation implemented by the NuCAPTCHA. We also included another method that changes the appearance of the animated characters through the animated frames using a twirl/ swirl effect which rotates a whole image around an anchor point[118].We also included an animated method that causes large parts of characters to be hidden through a long sequence of frames. This method could prevent tracking mechanisms from merging segments of the text from sequences of frames. We presented examples of these mechanisms in our attack on Xu's NuCAPTCHA and the Kund's CAPTCHA.

In this chapter, we provide details of the design of this new animated text-based CAPTCHA scheme. We also discuss the main distortion features used in it such as the partial hiding and twirl effect. We also discuss other features used to design the noisy objects and backgrounds and provide a detailed discussion of the robustness of the CAPTCHA scheme. These details include examinations through attacking methods previously employed against other CAPTCHA schemes. In addition, we provide details of the usability examination of our scheme, including the human success rates in solving the challenges and the time needed for humans to solve them.

The sections in this chapter are organized as follows: Section 11.2 discusses the design of the new CAPTCHA scheme, while in section 11.3 we discuss the security of the CAPTCHA scheme based robustness examinations against it. In section 8.4 we discuss the usability test. Section 8.5 summarises the chapter.

11.2 The new animated text-based CAPTCHA design

The AI-problems in our scheme is associated with character recognition in which the main object in each challenge consists of text. As we discussed previously, humans could solve these challenges by recognising the text and writing it in a specific box. Each challenge in our scheme displays the text with other animated objects that share the same movement behaviour of the text. We generated those objects by using animated images such as clouds, mountains and trees.

We distorted our challenges using the “emerging image” that displays the edges of the text and background with animated noises. We generated the challenge by using Xu et al.'s technique[10] that was implemented to generate his version of NuCAPTCHA. This technique

includes generating grayscale frames that show the edges of the text and background in bright colours in front of a black background. The method used to generate those frames includes calculating the norm of the derivatives of the image. The grayscale frames are then combined with a set of noisy images generated by Gaussian distribution. This combination results in a new set of grayscale frames. The value of each pixel in those frames is decided through the following equation:

$$I(x,y) = I_m(x,y) * \exp(\frac{I_n(x,y)}{const})$$

The variables in this equation as follows:

- I_m is the value of pixel within the coordination point (x, y) in each grayscale frame that include the text and background
- I_n is value of pixel within the coordination point (x, y) in each generated noisy image
- $const$ is constant value.
- exp is an exponential function.

Xu's technique also includes a constant threshold method that converts the new set of frames into binary frames. This method chooses a random value $t < 0$. It then converts every pixel that has a value greater than t into a white pixel, while it converts the rest of the pixels into black.

All contents included in this animated challenge image are discussed in the following sections:

- The design of the main challenge object. In this section, we discuss the main object and the distortion method used in it.
- The design of the noisy background. In this section, we discuss the noises included in the background.

11.2.1 *The design of the main challenge object*

The main challenge object design includes a text that humans most read and write in a specific box to solve the challenge. The animated movement of the main object is implemented with a method that makes the text start from a random position in random frames and then moves the text to random directions at random speeds. Both the direction and speed then keep changing randomly through the animated frames. The method determines the appearance, position and direction by constructing spine curves. The construction process includes choosing a sequence of random coordination points that are widely separated within the animated image. We connect these points using a periodic interpolating cubic spline curve algorithm created by Eugene Lee[119]. This algorithm returns a parametric variation cubic spline curve passing through the given sequence points. We can explain the algorithm with the following equation:

$$\sum_{i<j} \sqrt{\|points_{i+1} - points_i\|_2}$$

In which two coinciding points belong to a given sequence of points chosen by Eugene Lee's centripetal scheme. After that, the approach uses another method that chooses the random frame that the main object appears in. The movement speed of the object is determined through another method that sets a random speed ranging between two to five pixels per frame. This method also keeps changing the speed to another random speed in random frames.

The design of the animated text challenge included in this main object is based primarily on the animated CCT mechanism of the NuCAPTCHA. This mechanism was originally implemented to overlap the characters in the text, and rotates each of them individually. Our animated CAPTCHA incorporates a different design to make the mechanism more secure than its earliest design in THE NuCAPTCHA. The new design includes a random number of overlapped characters that rotate slowly with a rotation speed that reaches one degree ($\Pi/180$) per frame. The overlapped characters are distorted using two distortion methods. The first method uses an animated wrap algorithm to increase the difficulty of the automated attack to identify the slow rotation of the characters. The second method partially hides characters using a set of animated circles that keep moving to display different parts of the character in each small sequence of frames. Our aim was to combine

both animated distortion and rotation. This combination could help in preventing the automated attack from matching the visible pieces of the characters through the animated frames. Thus, they could resist the segmentation and tracking attacks.

The following sections include a detailed description of the animated distortion approach and partial hiding of the characters approach.

The animated circular wrap (twirl effect) distortion. We designed a technique using an approach that creates a map of image pixel coordinates around a specific point in the image of these characters. Then, we used this map to create an animated warp system where all the pixels of the characters rotate around the specific point of the image. This rotation angle and speed are different depending on the radius distance for every pixel from the centre of the rotated area. Tim Warburton[120] explained the use of this approach against a single image through the following algorithm that was used for each wrapping point against one frame:

Alogrithm: Processor Wrap

Input: $image$, coordinates of the rotation centre $(x_c, y_c), a, b, c, d$

Output: $image$ (after modification)

1: **for each** pixel coordinates x and y from the image of the characters **do**

2: Calculate the radius distance from the rotation centre using the equation:

$$r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$$

3: Calculate the radius angle of line between the point and the rotation centre using the equation:

$$A = \tan^{-1}(y - y_c)/\,(x - x_c)$$

4: Use both radius distance and angle of the original image point to calculate the new angle of the wrap:

$$A_n = A + \left(1 - \frac{1}{1 + e^{b(c-dr)}}\right) a$$

5: Use the results to calculate the new coordination of pixel in the wrap image using the two following equations:

$$x_n = x_c + r\ \cos A_n$$

$$y_n = y_c + r\ \sin A_n$$

6: Copy the pixel (x,y) from the original image into the new coordinates (x_n, y_n) in the wrapped image

7: **end**

8: **return** image

Where b, c and d are static values that can be chosen randomly for every challenge. a is a variable with value that keeps steadily increasing at speed of S until it reaches a value of M. it then starts to go backwards until it reaches the value of $-M$, so it can limit the animated wrap. The value of $M, -M$ and S are limited to protect the usability of the CAPTCHA. This algorithm can run two times against the same animated image. In each run, we choose different the values of $b,\ c,\ d$ and M. We also choose different starting value of a and coordination of the rotation centre (x_c, y_c) . We can explain our animated wrap algorithm as follows:

Alogrithm: Animated Wrap
Input: $AImage$ (Animated Image)
Output: $AImage$ (Animated Image)
1: $W \leftarrow$ $width$ of $AImage$
2: $H \leftarrow height$ of $AImage$
3: $x_c \leftarrow$ Random number between $(W/3)$ and $(2 * W/3)$
4: $y_c \leftarrow$ Random number between $(H/3)$ and $(2 * H/3)$
5: coordinated point of the rotation centre $(P) \leftarrow x_c, y_c$
6: 3: $b, c, d \leftarrow$ Random numbers
7: $M \leftarrow$ Random number between (2.0) and (4.0)
8: $a, A \leftarrow$ Random number between (M) and $(-M)$
9: **for each** $frame$ in the $AImage$ **do**
10: **if** $(A > a\ and\ a < M)\ or\ a = -M$ **then**
11: $A \leftarrow a$
12: $a \leftarrow a + 1$
13: **else**
14: $A \leftarrow a$
15: $a \leftarrow a - 1$
16: **end if**
17: $frame \leftarrow$ Processor Wrap(P, a, b, c, d)
18: **end**
19: **return** $AImage$

Figure 11.1. The animated wrap

Figure 11.1 shows an example of the animated wrap method.

The partial hiding of the characters. This approach uses animated circles to display a limited part of the animated characters, making them difficult to be gathered and identified by the recognition engine. The animated circles move through two opposite hertz cycle lines that are linked through their edges. The position of the link can be in the middle of the vertical area of the animated image in case the circles move horizontally or in the middle of horizontal area of the animated image and in case the circles move vertically. The coordination of the circles in each frame can be determined using the following equation in case the circle moves horizontally:

$$y = \frac{H}{2} sin\left(\frac{2\,\pi}{l} d\; x\right) + \frac{H}{2} \begin{cases} x \geq 0\; and\; x < W \\ d = +1\;\; or - 1 \\ W\; mod\; l = 0 \end{cases}$$

It can be determined by using the following equation in the event the circle moves vertically:

$$x = \frac{W}{2} sin\left(\frac{2\,\pi}{l} d\; y\right) + \frac{W}{2} \begin{cases} x \geq 0\; and\; x < W \\ d = +1\;\; or - 1 \\ W\; mod\; l = 0 \end{cases}$$

In which the W is the width of the image of the main object, the H is its height. The l is the wavelength. This wavelength should be a factor of the width in case the circles move horizontally. It should also be a factor of the height in case the circles move vertically. The d is a value that shows which one of the two opposite hertz cycle lines is crossed by each circle. This value can be determined by identifying the movement direction of the circle; it

would be +1 when the circle move forwards, while it would be -1 when the circle moves backwards. Figure 11.2 shows an example of the partial hiding of the characters.

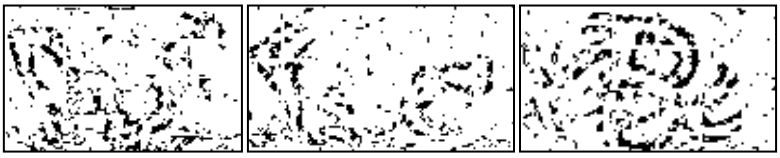


Figure 11.2. Partial hiding of the characters

11.2.2 *The design of noisy background*

The CAPTCHA also includes other noisy objects used in the background image. These are limited in each challenge to three objects created from animated images that are chosen randomly. Each of the objects is re-sized to match the width and height of the main object and animated using the same wrapping method. It also shares the same movement behaviour of the main object in which it appears and moves randomly through the animated frames. The main object still includes more noises than the noisy object. The noises in the main object are darker and thicker than the noises used in the noisy objects. Both the main challenge object and the noisy objects in the animated challenge images are also shown in front of a noisy background. This background is created from an animated loop image chosen randomly for each animated challenge image. Figure 11.3 shows an example of this noisy image.

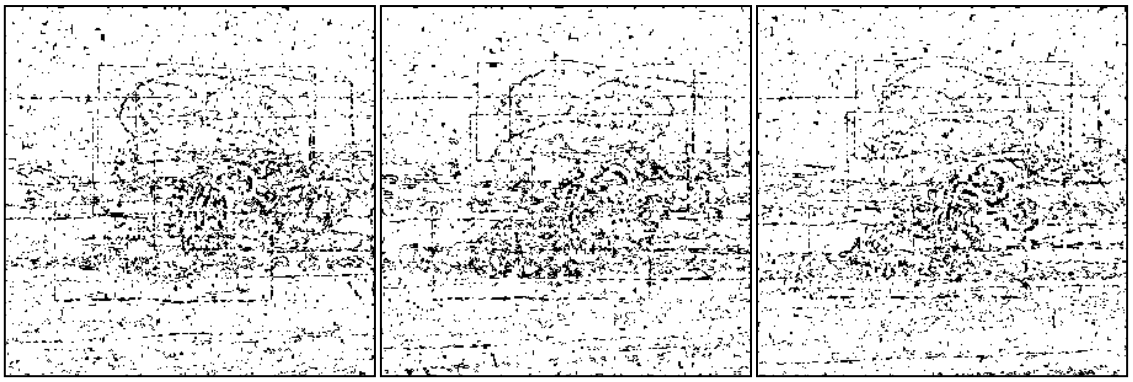


Figure 11.3. The noisy image

11.3 The security resistance of our new CAPTCHA scheme

As described before, the major problems in the recent animated CAPTCHAs are their emerging mechanisms that aim to confuse attacks between noises and main objects. However, our scheme overcame those problems with the new techniques described in the

previous section. These techniques could prevent the segmentation and tracking attacks from breaking this CAPTCHA and could be effective against the recent deep-learning attacks. In this section, we discuss the robustness of our scheme against tracking and segmentation attacks and discuss its robustness against deep-learning attacks.

11.3.1 *The security of the scheme against tracking and segmentation attacks*

We can discuss the resistance of our scheme against tracking and segmentation attacks by including a detailed description of every feature included in our attack:

Circular-wrap feature. The use of this feature could cause the segments of the text to change their positions in every sequence of the frame. As a result, it caused the identified segments of the text within the sequence of frames to be unmergeable. Thus, it could prevent the attacks from creating a full image of the text using the identified segments. Figure 11.4 (A) shows how the text appears, after we merge its identified segment within three sequences of frames. We can present the segments displayed in the figure as follows:

- the blue segments are taken from the right-side frame.
- The green segments are taken from the middle frame.
- The red segments are taken from the left-side frame.

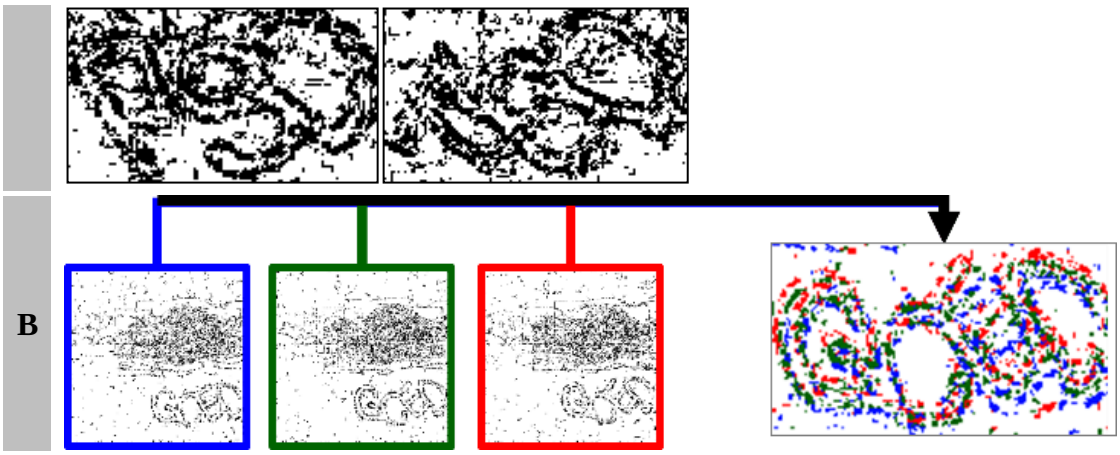


Figure 11.4. Attacking animated CAPTCHA challenges

The circular wrap feature could also improve the resistance of the scheme against vertical-segmentation attacks. The cause of this resistance is that circular wrap could move nearly half of one side of a character into the same vertical column of the opposite half side of the

other character beside it for a limited sequence of frames. Figure 11.4 (B) shows an example of how the animated wrap system could make an animated object that includes three characters “P”, “S” and “D” more resistant to vertical segmentation. The cause of this resistance is that the right side of “P” and the left side of “D” appear on both the right and left side of “S”.

Partial-hiding feature. This feature plays a major part in protecting the CAPTCHA against the attack by showing the human different parts of the character through the animated frames. The parts shown through the animated frames make them difficult to identify and connect together through the current attacking systems. The cause of this issue is the animated wrap that displays each of the parts in a different shape than the parts shown in different frames. The movement effects of text can also prevent attacks that predict the positions of text parts in each animated frame. This method caused the current attack to record a high failure, after we tested it during our research against the CAPTCHA animated images used in this approach.

Noisy objects. The additional features included in this CAPTCHA are the animated objects that move and distort using the same method against the main object that shows the animated characters. These animated objects are only provided to slow the current attacks against our animated CAPTCHA by confusing the attacking programs between them and the main object. However, the animated objects cannot prevent the attacking systems from identifying the main text. The causes of this issue are the usability requirements of the animated CAPTCHA systems which force developers to provide the noises in different shapes than the text. The different shapes make it easy for both human and computer to locate the text. Figure 11.5 shows an example of the attack against the CAPTCHA.

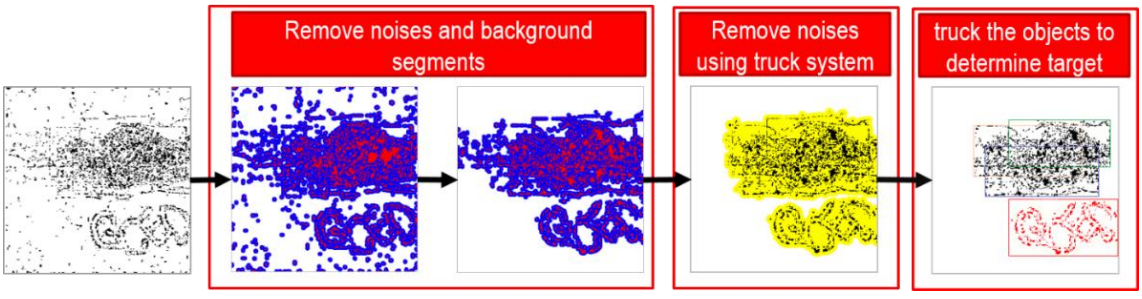


Figure 11.5. Removing the noises

11.3.2 *The security of the scheme against deep learning attacks*

Current deep learning techniques use advanced automatic solvers that segment and recognise the whole CAPTCHA image. These techniques use a highly improved recognition engine that connects neuron collections that process portions of text in the input CAPTCHA image/ frame. They can also be used as part of a generic attack that includes segmentation algorithms to improve the results of the attack. A few examples of these algorithms were provided by Bursztein et al.[79] and Yan et al.[80]. Although we could not examine this scheme using an active deep-leaning attack, we still used similar techniques such as our attack against the ReCAPTCHA. These techniques included a convolutional natural network to develop a basic machine-learning mechanism that could segment the text.

The results of our examination using those attacks showed that the partial hiding can only be a feature that provides full protection against those types of attacks. As mentioned before, such a feature prevents general attacking approaches from identifying the full image of the character and its position in the animated image. The same applies to the current deep-learning attacks which are highly dependent on visibility of the whole text in each frame. The exclusion of this feature can cause the text to become vulnerable to deep-leaning attacks. This vulnerability issue is present even in animated challenges that use the animated wrap feature. Our examinations showed that those attacks could recognise the parts of the emerged texts that were gathered from a sequence of frames. The main cause of this problem is the usability mechanisms that cause the noisy parts of the text to be close to each other. These mechanisms cause the hypothesis of the characters to be visible for the recognition engines to recognise the characters.

Nevertheless, these kinds of attacks could take a long time to exploit the challenges that use animated wrap. The issue is in the design of the wrap method that adds changes to shapes of text in every frame. These changes cause each animated character to take the shape of other characters for a short length of time before returning to their recognisable shape. This feature causes the deep learning attacks to scan multiple frames before they find recognisable shapes of all characters in each challenge. The use of emerging techniques could also help to resist these attacks through similar patterns of noises used to display the text and the background in each frame. However, the attacks could still exploit

this technique by applying tracking mechanisms to clean text before applying the deep learning techniques against them.

We can prove parts of our overall findings with our closest attacking approach to those used in a deep learning attack. This attack uses a machine-learning technique to recognise the patterns of each character in the text. It then connects the results to find the right answer. Our examination showed that the attack analysed at least 13 frames in every challenge that included the animated wrap to achieve a success rate of above 20% (an exact score of 23%) against 100 samples. The attack took an average of seven seconds in analysing each of these challenges. It also needed to recognise 21 frames of each challenge to match the success rate of our attack against the Kund's CAPTCHA, which is 37%. As a result, our attack took an average of 248.375 seconds to analyse each of the challenges, more than 97.451 seconds longer than the average time taken in our attack on the Kund's CAPTCHA.

11.3.3 *The security of the scheme against relay attacks*

The current version of our CAPTCHA scheme was not built to resist relay attacks and, as such, could prove vulnerable to attacks that target the game engines. Still, it could potentially resist some attacks that target the web CAPTCHA schemes through the emerging and wrap features. These features force the relay attacks to capture every frame in the image to make it visible to the solvers that help the attackers. Thus, we could detect these attacks by calculating the time taken to solve the animated challenges. We could also make the same assumption as Van Oorschot et al.[39], Martiyama et al.[36] and Kanich et al.[40] in discussing the security of our scheme, as the effort needed by humans to solve it could prevent them from solving our CAPTCHA challenges at a cheap price. Therefore, the use of relay attack against our scheme could cost the attackers too much money

We could also include other relay prevention mechanisms in the current design of our scheme, for example, a mouse-movement detection mechanism that is based on Mohamed et al.[43] mechanism. This mechanism requires the user to click on an animated image of the text or other noisy image that display an animal to solve the challenge. It could also require the user to click on another animated image after solving the challenge. Such a strategy could simplify the relay prevention against the CAPTCHA challenges.

11.4 The usability study of our new CAPTCHA scheme

We have studied the usability of the CAPTCHA in a user study of 75 participants. We conducted the study through a website that provided each participant with five random challenges for each of the two different versions of our animated CAPTCHA challenges. The first version used in our usability study does not include the animated partial character hiding approach, while the second version includes all the distortion approaches together. Each challenge included in both versions is in the form of text challenge of three letter challenges that the participants must write in a specific box. This text challenge is also provided with a ratings box from 1 to 5, in which 1 is the lowest rating and 5 the strongest. The website then records details of the participants' answers, including the number of correct answers, errors, skipped challenges, the time taken to solve each CAPTCHA challenge and the rating. In this section, we present detailed results of our examinations, before we discuss those results and the lessons learned from our examination.

11.4.1 *The answer accuracy analyses*

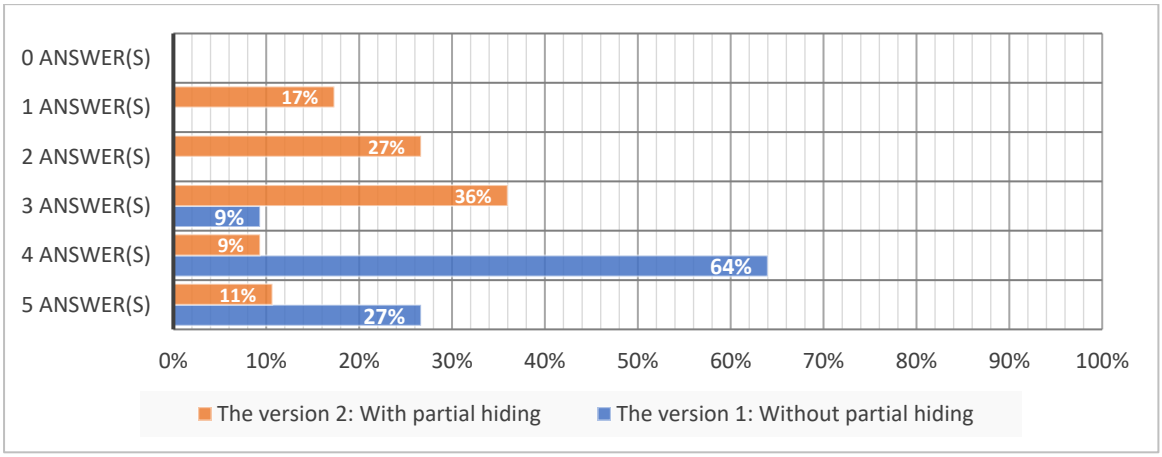


Figure 11.6. The number of correct answers

We firstly analysed the accuracy analyses per user. The results showed the first version that did not include the partial hiding method which was easy for the participants to solve, with an 83.4% success rate. 20 (27%) of the participants answered all the challenges. 48 (64%) solved four challenges correctly out of five. The rest of the participants (9%) solved only three challenges out of five. However, the second version that included the animated partial hiding of the characters was a bit of a struggle to solve with a 54% success rate. The results show that only eight (11%) of the participants provided the correct answer to all the challenges. Seven (9%) of the participants solved four challenges out of five. 27 (36%) of

the participants solved three challenges. 20 (27%) of the participants solved two challenges. 13 (17%) of the participants solved one challenge. Figure 11.6 shows the detailed results of our first analysis.

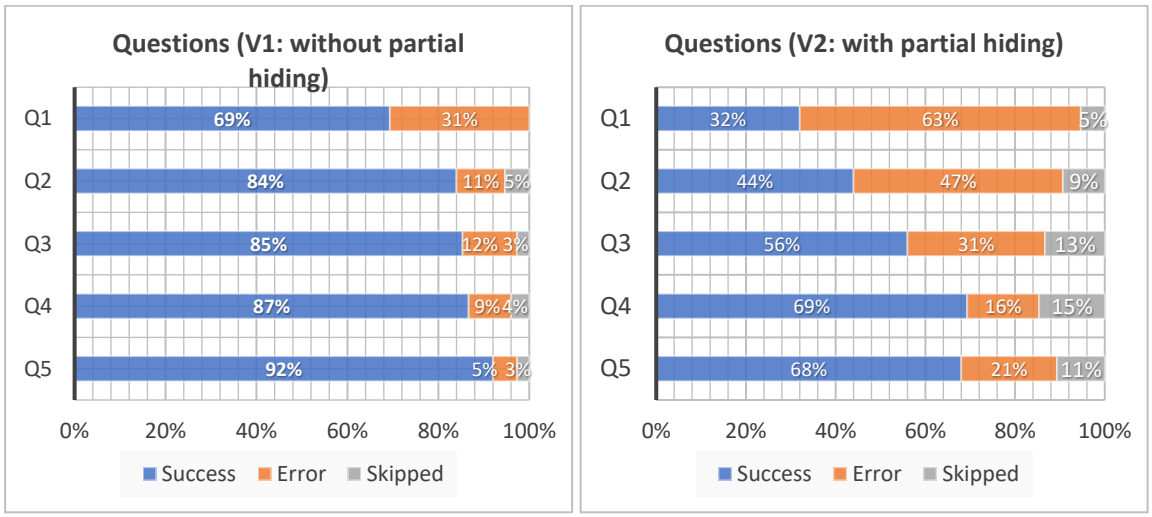


Figure 11.7. Success, error and skipped answers for each of five challenges

We also analysed the results using another chart that provides details of the answers to each of the five challenges provided for both versions in figure 11.7. These details include the following:

- Success, which is the rate of correct responses provided by the participants for each of the five challenges.
- Error, which is rate of responses that do not match the challenge solution.
- Skipped, which is the rate of the skipped responses to each challenge in which the participant clicked a button to receive a new challenge.

Our analysis showed that the lowest rate of correct answers in our examination was recorded during the participants' attempt to solve the first challenges of both examined versions. The answers include 69% of responses to the first challenge of the first version that did not include the partial hiding approach. They also include 32% of responses to the first challenge of the second version that included the partial hiding approach in its challenges.

The results of the examination based on that chart also showed an improvement in solving the animated challenges. Based on these results, the participants provided 92% correct responses to the fifth challenge of the first version, after they solved 69% of the first challenges correctly. They also showed an improvement in solving the challenges between the first and the fourth challenges of the second version. Based on these results, the participants achieved a success rate of between 68% and 69% in solving the fourth and fifth challenges, after they solved 32% of the first challenges correctly. The results show that the CAPTCHA needs to be used a number of times to be understandable by the users. The results also show the inability of a small percentage of users to solve the second challenges as the ratio of skipped challenges reached an average of 11%. This problem was because of the combination of the "emerging image" and the partial hiding that caused some of the animated text challenges to be invisible to the human eye.

11.4.2 *Analyses of text errors*

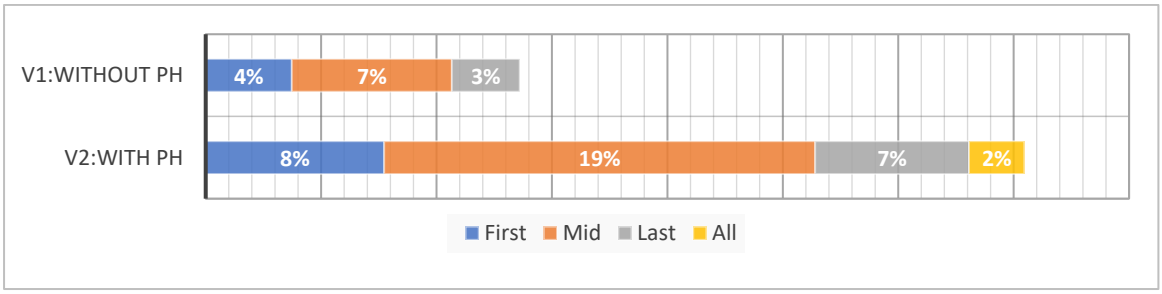


Figure 11.8. Location of errors within code words

Our usability study also included an analysis summarised in figure 11.8 of the distribution of errors within the uncorrected text responses to animated text challenges. The results of this analysis show that most errors were made on the middle character of the text challenge. The errors included 7% of total answered challenges that belonged to the first version (without the partial hiding approach). They also included 19% of the total answered challenges belonging to the second version (with the partial hiding approach). Our examination of these errors showed that the main cause was the similar appearance of the characters to others. These appearances caused confusion between the participants in reading a few characters, including those that changed their appearance for a short amount of time. The most common pairs of characters leading to these errors were "O/Q", "4/A", "R/P", "5/S", "2/Z" and "V/U". These errors were mostly recorded in the examination results of challenges that were solved very quickly by the participants.

11.4.3 *The analyses regarding the time taken to solve challenges*

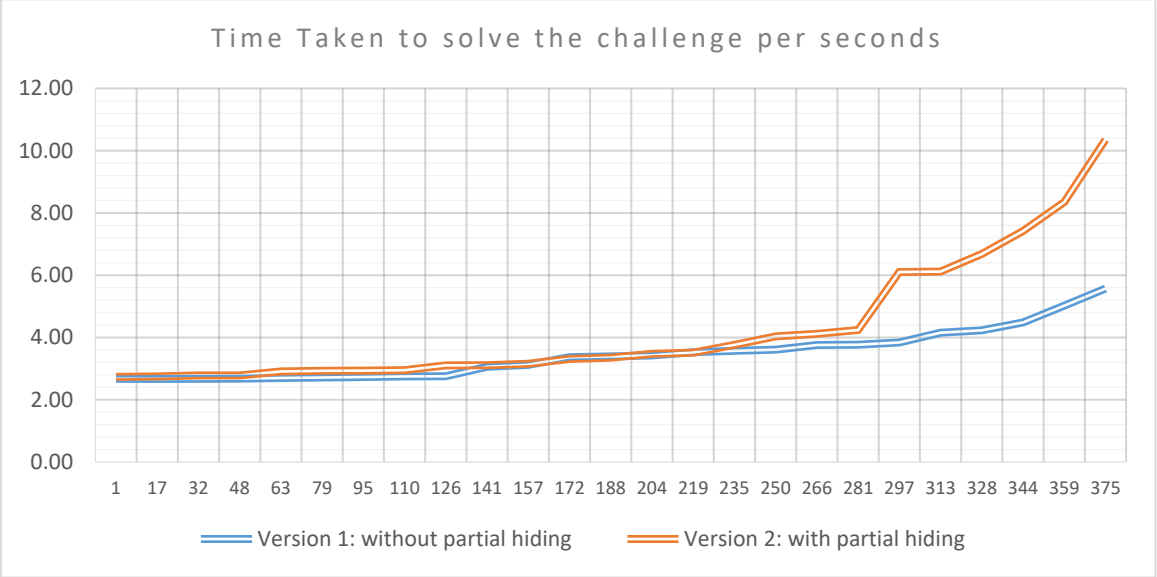


Figure 11.9. The time taken to solve the challenges

Our usability examination of our scheme is also an analysis of the time taken to solve the challenges. The results of our examination included in figure 11.8 shows that participants were quickly solving the challenges presented by both versions. The average time recorded to solve the first version was 3.45 seconds with a standard deviation of 0.77018 seconds. The longest time to solve a challenge was 5.57 seconds, and the shortest time 2.67 seconds. The users took longer to solve the second version with an average time of 4.28 seconds with a standard deviation of 1.99696 seconds. The longest time taken to solve a challenge was 10.35 seconds, and the shortest 2.74 seconds.

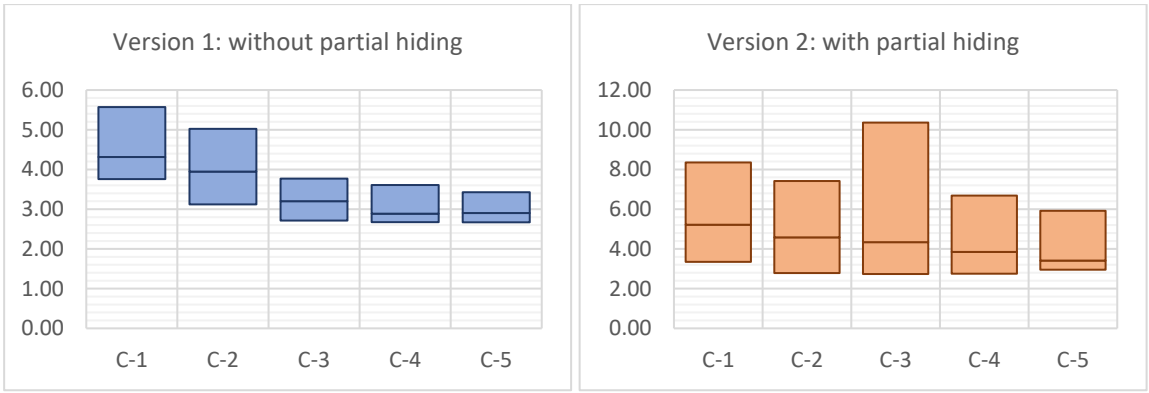


Figure 11.10. The time taken by challenge: Maximum→Average→Minimum

We include an analysis of the time taken per challenge in figure 11.10. Our results based on this analysis showed that participants improved their times in solving the challenges provided to them from both versions. The results showed the average time for solving the challenges of the first version improved from 4.31 seconds to 2.91 seconds. These results also showed that the average time taken to solve the challenges of the second version improved from 5.22 seconds to 3.41 seconds.

11.4.4 *Rating the CAPTCHA challenges*

As we said before, we also asked the participants to rate the CAPTCHA challenge they solved, in which they needed to choose a rating between 1 (the lowest rating that indicates the challenge is awful) and 5 (the highest rating that indicates the challenge is awesome). We present the results of this examination in figure 11.11.

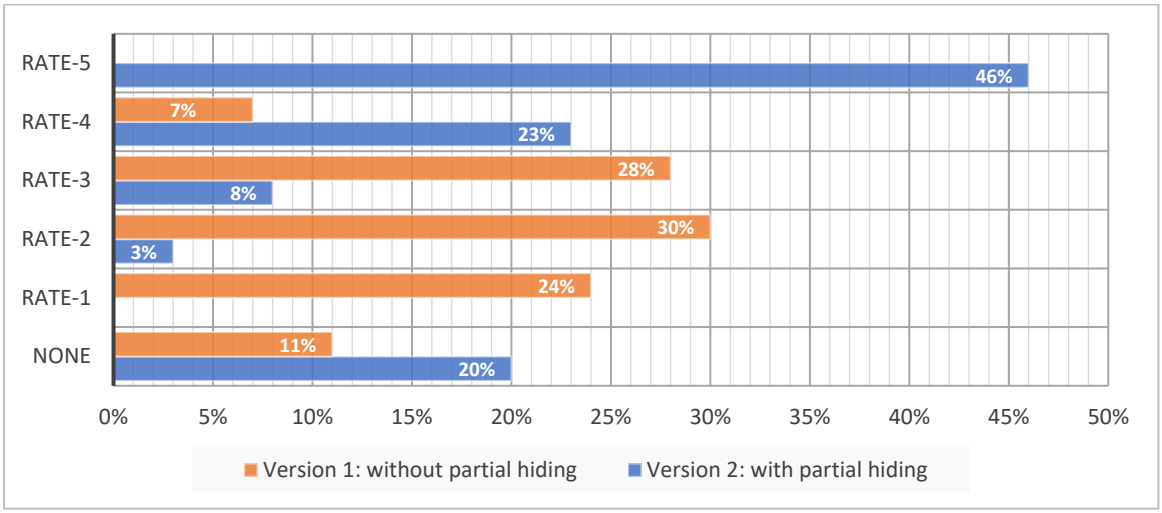


Figure 11.11. The given usability ratio of the challenges by participants

As this figure shows, we found that 52 of the participants gave the first version of the CAPTCHA an average rating of 88%. We could provide more details about the rating as follows:

- 46% of the challenges were given a rating of 5.
- 23% of challenges were given a rating of 4.
- 8% of challenges were given a rating of 3.
- 3% of challenges were given a rating of 2.

Yet, the participants gave the second version a far lower rating than the first challenge with an average rating of 44%. We can provide more details of the ratings as follows:

- 7% of challenges were given a rating of 4.
- 28% of challenges were given a rating of 3.
- 30% of challenges were given a rating of 2.
- 24 % of challenges were given a rating of 1.

11.5 Discussion

Comparing the results of our security and usability examinations against our CAPTCHA scheme with the results of other schemes, we can prove that this CAPTCHA solved many vulnerability problems present in both standard and animated CAPTCHAs. The vulnerability problems include exploitable behaviours that could aid the attacks in extracting, cleaning and segmenting text. We gave various examples in our attacks against KillBot, CAPTCHANIM, NuCAPTCHA, Yahoo and the Kund's CAPTCHA. We have also presented a potential solution for vulnerability problems of the recognition and automatic segmentor attacks as discussed previously in section 11.3.1. The usability study also showed an improvement in the balance between security and usability. The results showed that our attack would need to take 248.375 seconds to solve the CAPTCHA in comparison to an average of 3.46 seconds recorded by humans. The nearest result was in our attack against the Kund's CAPTCHA, in which the attack took 150.024 seconds to solve the CAPTCHA in comparison to an average of 11 seconds recorded by humans.

Looking at our usability examinations, we also learn that the first version of attack that only includes the animated wrap achieved a good usability score in terms of accuracy and speed. Although it is still 7% behind the Chellapilla limited score of 90% for a good CAPTCHA as stated in 2005, we should note that all CAPTCHAs achieving that score are entirely broken, as AI programs improve over time. However, we found that animated partial hiding would still need improvements, as the usability examination showed that users struggle to solve the challenges that included this mechanism. Nevertheless, the results of our examinations still showed that one cause of the usability problem is the extent of learning and understanding. These results were based on the level of improvement in solving the

challenge as this CAPTCHA is still new. We also noted that most of participants were rushing to solve this CAPTCHA rather than taking enough time to solve the challenges. This is a result of humans being more adaptable to solving standard text-based CAPTCHA challenges as opposed to the animated challenges.

11.6 Summary

We have designed an animated text-based CAPTCHA that uses the emerging technique to display overlapped characters that rotate and change their pattern using an animated distortion technique. These overlapped characters move together in random direction at random speed alongside other noisy objects in front of a noisy background. This CAPTCHA scheme includes two versions of animated characters. The first version displays the full overlapped characters, while the second hides overlapped characters partially through the animated frames. We examined the two versions of the CAPTCHA in terms of robustness and usability. These examinations showed that the second version could achieve the highest level of security required against the attack although it suffers from usability problems identified in our test. The first version is more usable than the second version, but suffers from security issues.

Chapter 12. Conclusion

We have presented a set of case studies aimed at evaluating the robustness of the animated text-based CAPTCHA and other standard text-based CAPTCHA schemes. Some CAPTCHA schemes depend on complex animated distortion against the recognition systems. Other schemes rely on tracking resistance principles and segmentation resistance practices. Our studies include a set of examinations based on identified vulnerabilities in the design of each CAPTCHA scheme. The identified problems helped us to develop different approaches that aim to track the characters and determine the connected characters necessary to segment them. The methods used in our attacks mainly rely on technologies that use image processing to analyse the components in every single image or frame. These methods are also supported by tracking techniques that link those components with their other images through animated frames. Our techniques also include several segmentation mechanisms such as the automatic segmentor, shape pattern identifier and vertical histogram techniques. Our aim is to discover the design flows in our targets and understand the reasons for the security mechanism failure in each. Our understanding of these failures also helped us to understand how to progress and avoid them in future versions of the animated test-based CAPTCHA. Thus, it helped provide a design guideline and use it to create a new CAPTCHA. Our study did not include any attempt to examine the CAPTCHA as an open AI problem, as our focus was mainly on the security mechanisms only. This overall research contributed greatly to our work and helped us to answer our main research question, which we will discuss in this chapter.

This chapter is organised as follows. Section 12.1 outlines the contributions made by our research in detail. Section 12.2 provides detailed answers to the questions we asked through our research. Section 12.3 provides details of our future work.

12.1 Summary of contributions

Our research included in our thesis has made many contributions, including the following:

- **A number of segmentation and tracking attacks.** We developed many tracking attacks against animated text-based CAPTCHAs. These attacks include advanced character extraction mechanisms which track the characters individually. We developed these attacks as a part of our investigation into the ability of the computer program to beat the human in recognising the individual animated characters. In

particular, our examination was aimed at the distorted characters through complex methods such as “animated rotating and “horizontal deformation”. We also developed tracking attacks to break tracking resistance mechanisms such as “emerging image”, “noisy lines”, and “noisy characters”. We designed the attacks to target the most resistant mechanisms with current attacks, rather than every new CAPTCHA scheme. Thus, we could have underlined their security requirements as impractical [25]. These CAPTCHAs include animated schemes that represent major tracking resistance mechanisms used currently in the animated schemes. For example, we targeted the Kund's CAPTCHA and Xu's NuCAPTCHA that were based on Niloy Mitra’s “emerging-image” mechanism[11], which is regarded as the most resistant mechanism to tracking attacks. We also developed attacks on Yahoo's noisy letters mechanism that is used in their CAPTCHA. In addition, we also developed various segmentation attacks as part of our examination of standard text-based CAPTCHAs. These attacks included a basic novel attack against the Wikipedia CAPTCHA and an automatic segmentor attack on the 2013 version of the ReCAPTCHA.

- **Toolbox for examining the CAPTCHA scheme.** Every tracking attack designed in our research includes techniques that could break various CAPTCHA schemes. For example, both examinations against Xu’s NuCAPTCHA and Kund's CAPTCHA included a similar distant noises removal process. Other attacking mechanisms also targeted CAPTCHA schemes separately or alongside the tracking attacks. These mechanisms included advanced character extraction and segmentation mechanisms developed for both standard and animated text-based CAPTCHAs. For example, we examined the CCT mechanism as deployed in the ReCAPTCHA through an automatic segmenter and then examined an enhanced version of this mechanism as deployed in the Kund’s CAPTCHA using a similar technique to that used by the ReCAPTCHA. This shows that the standard segmentation attack could even contribute to the tracking attacks against animated schemes.

- **A list of robustness and usability issues in the current CAPTCHA design.** Our examinations of standard and animated text-based CAPTCHAs helped us to learn key lessons about their robustness and usability problems. We discussed every key

lesson, in addition to trade-offs between robustness and usability identified in our targets. We also considered the reasons why the resistance mechanisms failed against our attacks, as well as our defence against them. Based on those examinations, we found that most identified failures in our targeted schemes are due to the main invariants identified in standard text-based CAPTCHAs. Many of those invariants were hidden by the distortions and tracking-resistance mechanisms. These mechanisms that aim to protect the invariants may increase the challenge against the CAPTCHA attacks. Still, they do not present a sufficient level of security to protect the CAPTCHA against the attacks.

- **A better understanding of the CAPTCHA design.** Our study also contributed to understanding issues on the design of both standard and animated text-based CAPTCHAs. Few of those designs included the complex tracking-resistance mechanisms. Others included complex animated distortion methods that remain challenging against attacks. For example, we examined the animated rotation method that is difficult to break when it is used to distort confusing characters and segmentation-resistance mechanisms in both animated and standard text-based CAPTCHAs.

- **A new CAPTCHA design guideline.** Our research on those vulnerabilities also helped us to highlight a set of design guidelines and key attributes. This guideline includes the following:

    i. Recommendations related to the number of characters needed to be included in the text and characters set.

    ii. Suggestions to improve the resistance of animated text-based CAPTCHAs against segmentation attacks.

    iii. suggestions to improve the tracking-resistance mechanisms.

- **A new animated text-based CAPTCHA.** We used our guidelines to design a new animated text-based CAPTCHA scheme that uses the emerging technique. This scheme combines resistance mechanisms that prevent the attacks from gathering text segments. Few of these mechanisms keep animating the shape of connected

characters and hiding parts of them for a limited time. Others randomise the movements, direction and speed of the text alongside other noisy objects. This design contributed to developing our understanding of the main robustness and usability issues of animated text-based CAPTCHAs (see chapter 11).

12.2 Research questions

In this section, we provide detailed answers to the main research questions based on our research in this thesis.

12.2.1 *The first question*

The first question was: *can the animated techniques included in the new CAPTCHA schemes provide the required level of robustness against the attacks?*

As a result of our examinations, we have shown how we can break animated text-based CAPTCHA schemes with tracking attacks. In particular, we broke animated schemes that use complicated tracking-resistance mechanisms. Those mechanisms included “emerging image”, “noisy lines” and “noisy characters”. We have also concluded that the computer could recognise the animated characters better than the human. We showed this result even on animated characters with complex distortion methods such as “animated rotation”.

We can explain our results by looking at the main definition of CAPTCHA, which is a test that is easy for humans but difficult for computer programs to pass. Such tests can be operated as AI open problems under a set of conditions defined by Luis von Ahn. One condition is where the test can be generated automatically followed by an infinite number of tests and so on[2,4]. Many of these tests are broken by security engineering examinations that identify critical vulnerability problems and develop a simple attack to exploit them. These tests led the researchers to design new defence mechanisms that are resistant to previous attacking techniques and their possible enhancements. These include all the resistance mechanisms examined in our previous chapters.

However, our main examination of those mechanisms did not rely on those attacks and their enhancement. Instead, we developed techniques that mainly focused on exposing the weaknesses in those new mechanisms. For example, Xu et al.[10] developed many versions of the NuCAPTCHA scheme based on their examination of the original version of that CAPTCHA. They also examined that version by using the enhanced versions of their own

attack. These examinations led Xu to release a version of NuCAPTCHA that uses the "emerging image" mechanism. This version was the only version that could withstand Xu et al.'s attack on the NuCAPTCHA. Thus, they described this version as the only animated scheme that could resist all attacks against CAPTCHA. However, our examination of this CAPTCHA (see chapter 7) proved that theory wrong by using a new attack. This attack also exposes many vulnerability issues in the "emerging image" mechanism used in this scheme.

Nevertheless, we still consider the tracking-resistance mechanisms as open problems to protect the websites from attacks. Our consideration is based on how the vulnerability issues identified through our attacks are still avoidable in the future design of animated schemes. We provided an example by designing a new animated scheme that could withstand the possible attacks. The initial design of this scheme can still be improved in terms of its usability and security.

12.2.2 *The second question*

The second question was: *can the segmentation resistance mechanism used in the latest standard text-based CAPTCHA schemes still provide the additional required level of resistance against attacks that are not present missed in animated schemes?*

The results of our attacks against ReCAPTCHA and the Wikipedia CAPTCHA showed high vulnerability issues in them. The results included a success rate of 43% against the 2013 version of ReCAPTCHA and a success rate of 70% against the Wikipedia CAPTCHA. Those results also showed that vulnerability issues could also lead the attacks included in our test to compete with the human in terms of speed. For example, the attack against the Wikipedia CAPTCHA took an average of only 17.67 milliseconds to solve its samples. Our attack against the 2013 version of ReCAPTCHA took an average of 4.88 seconds to solve its samples. Based on these results, we can conclude that the mechanisms used in standard text-based CAPTCHA schemes do not provide the required robustness against attacks. They also disprove that animated text-based CAPTCHAs provide a lower level of security than standard text-based CAPTCHAs. As a result, we recommend using the animated resistance mechanisms to improve the security of text-based CAPTCHAs instead of relying on segmentation-resistance mechanisms alone.

12.2.3 *The third question*

The third question was: *how much space is available to design an animated text-based CAPTCHA scheme that could provide a good balance between security and usability?*

We answered this question by designing a new animated text-based CAPTCHA scheme based on our previous research and which includes mechanisms that provide greater required levels of usability and robustness against all possible attacks. Our examinations of this scheme showed the possibility of providing the least practical balance between robustness and usability without failing against attacks. Yet, our examinations showed the usability score of this scheme fall below the stated score for a good CAPTCHA by Chellapilla in 2005[21]. In particular, those examinations showed participants scoring 83.4% in recognising our scheme, which is 6.6% below Chellapilla's score of 90%.

Nevertheless, our previous research showed high vulnerability issues in the schemes that passed Chellapilla's usability argument. Most of these issues are due to exploitable invariants considered to be major usability requirements to reach that score. The success of the attack in identifying exploitable invariants also showed the AI programs matching the human's capacity to solve CAPTCHA problems. As a result, many researches have prioritised the sacrifice of usability over security as result of advice stated by Weir et al.[121]. Meanwhile, other researchers declared the text-based CAPTCHA to be completely solved, including Google's Bursztein et al.[79] in 2014, who encouraged many websites to replace the text-based AI problems with other AI tests. This research led Google to develop a new test in its ReCAPTCHA that detects human behaviour through the mouse or touch screen. This test is also assigned with other AI tests that run when the first test fails to detect human behaviour[122]. However, those tests did not resist the CAPTCHA attacks due to the machine's inability to read human behaviour. The most successful attack on those tests was developed by Suphannee Sivakorn et al.[123] who broke them with a success rate of 81%. This results still shows the importance of the limited usable text-based CAPTCHA until recently.

12.3 Future Work

We propose several tasks that could enhance our current research in the field of the CAPTCHA. These tasks include a series of developments of our new animated text-based

CAPTCHA schemes and other new tasks which we are going to discuss in detail in this section.

Improvement to the new animated text-based CAPTCHA. As we build a new animated text-based CAPTCHA scheme and examined its robustness and usability, we have learned more about the future models of the animated text-based CAPTCHA schemes. However, we still consider that the design of our scheme was not the only possible design to stop the current attacks. This consideration is based on the fact that the area of the animated text-based CAPTCHA is still a new and open problem. Thus, many features are available for experimentation in the field of the animated text-based CAPTCHA scheme many of which could be applicable to the design of our new animated text-based CAPTCHA scheme such as:

- *The animated colour features.* We could add an animated colour feature to improve usability in which the main characters could appear in a different colour (or a combination of colours). This feature could avoid the use of such an attack to track the characters by many design approaches. For instance, we could include an approach that fills different parts of the text and background with different colours or fill the text with a specified colour for a limited time. It could then change the position of that colour to fill another part of the image.

- *Text puzzle challenges.* Instead of using the text as a single challenge object, we could use many animated text objects that are displayed in different fonts. This text could also include different combinations of characters associated with one puzzle question. The user could answer this question by choosing one of the animated text objects displayed in the challenge. For example, we include an animated challenge that displays four text objects such as “CHINA”, “SEA”, “DUCK”, and “PLUTO”. We could assign this challenge with a question: “CHOOSE A PLANET NAME” in which the user should answer “PLUTO”. This challenge complicates the attacks, as they need to recognise all the animated texts before choosing the correct answer. As result, they could also help us lower the level of distortion used against the challenges

- *Use other distortion mechanisms for the animated text challenge.* In our scheme, we presented one design specification of an animated text challenge that included

two methods to distort it. However, we can still offer other mechanisms to distort text challenges and partially hide the animated text. Thee mechanisms could be used as replacements of our mechanisms or in addition to them. Building these mechanisms would help us to find ones that provide the best balance between the robustness and usability in our scheme.

Expanding our research in the field of CAPTCHA. in our current research, we have exploited many standard animated text-based CAPTCHAs. However, the field of the CAPTCHA includes other types of CAPTCHA schemes that were not explored in our research. These other CAPTCHA schemes include the object recognition CAPTCHAs, which display images or visual videos and are associated with different puzzle questions. Such challenges require a specified action by the human in response to the images or the videos to solve. The design of challenges requires different attacking mechanisms. These mechanisms mostly include methods that understand the required action towards the associated images and videos to solve the challenge. They also include other methods that analyse the images or videos associated with the challenges to determine the correct answer. We could also include other studies that aim to solve the audio-based CAPTCHA and study new features that detect the attacks through mouth behaviour, such as the feature developed by Google[122].

Building an attack detector. In our research, we have presented many attacks in our examination of CAPTCHA schemes. These attacks also helped us to identify the main weaknesses of those CAPTCHA schemes against our attacks. Nevertheless, we have still reported a few faults in those attacks during the segmentation and recognition steps. Those reports can assist greatly in building a tool to expose any similar attack by detecting the series of faults. Such a tool could include a combination of approaches that analyse the wrong answers returned from a client's computer. They could then determine if the wrong answers are caused by common errors in computer attacks. For example, we could use an approach that check if the returned answer to the text challenge "mall" is "Inall" or "inall". This approach could then suspect an attack, as those faults are commonly caused by a segmentation approach that segments "m" into "ı" and "n". The tool could also include other methods that check the time taken by the client to solve the answer and check the number of responses returned from a specific IP address or a list of proxy (anonymous) IP addresses per hour.

Reference

1. Pingdom. Internet 2010 in numbers. http://royal.pingdom.com/2011/01/12/internet-2010-in-numbers/. Published 2010.

2. von Ahn L, Blum M, Langford J. Telling humans and computers apart (automatically): or how lazy cryptographers do AI. *Comput Sci Dep Carnegie Mellon Univ*. 2002:149.

3. Schonfeld E. 4Chan Takes Over The Time 100. http://techcrunch.com/2009/04/21/4chan-takes-over-the-time-100/. Published 2009.

4. Von Ahn L, Blum M, Langford J. Telling humans and computers apart automatically. *Commun ACM*. 2004;47(2):56-60.

5. ENGBER D. Who Made That Captcha? Who Made That Captcha? http://www.nytimes.com/2014/01/19/magazine/who-made-that-captcha.html?_r=0. Published 2014. Accessed May 24, 2016.

6. Chellapilla K, Larson K, Simard P, Czerwinski M. Computers beat humans at single character recognition in reading based human interaction proofs (HIPs). In: *Proceedings of the Second Conference on Email and Anti-Spam*. ; 2005:21-22.

7. Yan J, El Ahmad AS. Captcha robustness: A security engineering perspective. *Computer (Long Beach Calif)*. 2010;(2):54-60.

8. Bains P. Video Based Captchas Now Available For Sites And Blogs. NuCaptcha. http://www.nucaptcha.com/press-release/video-based-CAPTCHAS-now-available-for-sites-and-blogs. Published 2011.

9. Parfeni L. NuCaptcha Introduces Video-Based Captchas. http://news.softpedia.com/newsPDF/NuCaptcha-Introduces-Video-Based-Captchas-145824.pdf. Published 2010.

10. Xu Y, Reynaga G, Chiasson S, Frahm JF, Monrose F, Van Oorschot PC. Security and usability challenges of moving-object CAPTCHAs: decoding codewords in motion. In: *21st USENIX Security Symposium Conference*. ; 2012.

11. Mitra NJ, Chu H-K, Lee T-Y, Wolf L, Yeshurun H, Cohen-Or D. Emerging images. In: *ACM Transactions on Graphics (TOG)*. Vol 28. ACM; 2009:163.

12. Turing AM. Computing machinery and intelligence. *Mind*. 1950;59(236):433-460.

13. Baird HS, Coates AL, Fateman RJ. PessimalPrint: a reverse Turing test. *Int J Doc Anal Recognit*. 2003;5(2-3):158-163.

14. Von Ahn L, Blum M, Hopper NJ, Langford J. CAPTCHA: Using hard AI problems for security. In: *Advances in Cryptology—EUROCRYPT 2003*. Springer; 2003:294-311.

15. Yan J. Bot, cyborg and automated turing test. In: *International Workshop on Security Protocols*. Springer; 2006:190-197.

16. Mori G, Malik J. Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In: *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. Vol 1. IEEE; 2003:I-134-I-141 vol. 1. doi:10.1109/CVPR.2003.1211347.

17. An Explanation of l33t Speak. BBC. http://www.bbc.co.uk/dna/h2g2/A787917. Published 2011.

18. Lillibridge MD, Abadi M, Bharat K, Broder AZ. Method for selectively restricting access to computer systems. February 2001.

19. P. A. R. Center. History of CAPTCHA. http://www2.parc.com/istl/projects/captcha/history.htm.

20. Snyder C, Southwell M. *Pro PHP Security*. Apress; 2005.

21. Chellapilla K, Larson K, Simard P, Czerwinski M. Building segmentation based human-friendly human interaction proofs (HIPs). *Hum Interact Proofs*. 2005:173-185. http://www.springerlink.com/index/h9kxwalaguwy5qyd.pdf.

22. Tam J, Simsa J, Huggins-Daines D, Von Ahn L, Blum M. Improving audio captchas. In: *Symposium On Usable Privacy and Security (SOUPS)*. ; 2008.

23. Rob. Why use text-based CAPTCHA logic questions. TextCAPTCHA. http://textcaptcha.com/why. Published 2009.

24. Bigham JP, Cavender AC. Evaluating existing audio CAPTCHAs and an interface optimized for non-visual use. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM; 2009:1829-1838.

25. Elson J, Douceur JR, Howell J, Saul J. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. In: *ACM Conference on Computer and Communications Security*. Vol 7. ; 2007:366-374.

26. Datta R, Li J, Wang JZ. IMAGINATION: a robust image-based CAPTCHA generation system. In: *Proceedings of the 13th Annual ACM International Conference on Multimedia*. ACM; 2005:331-334.

27. Gossweiler R, Kamvar M, Baluja S. What's up CAPTCHA?: a CAPTCHA based on image orientation. In: *Proceedings of the 18th International Conference on World Wide Web*. ACM; 2009:841-850.

28. Kluever KA, Zanibbi R. Video CAPTCHAs: usability vs. security. 2008.

29. D'Souza D, Polina PC, Yampolskiy R V. Avatar captcha: Telling computers and humans apart via face classification. In: *Electro/Information Technology (EIT), 2012 IEEE International Conference on*. ; 2012:1-6.

30. Goswami G, Powell BM, Vatsa M, Singh R, Noore A. FaceDCAPTCHA: Face detection based color image CAPTCHA. *Futur Gener Comput Syst*. 2014;31:59-68.

31. Howard Yeend. Breaking CAPTCHA without OCR « puremango.co.uk. Puremango.co.uk. http://www.puremango.co.uk/2005/11/breaking_captcha_115/. Published 2005.

32. Sharma P, Tyagi N, Singhal D. CAPTCHAs: Vulnerability to Attacks. *Int J Emerg Trends Technol Comput Sci*. 2013;2(2).

33. DeCaptcher. DeCaptcher. https://de-captcher.com/.

34. DeathByCaptcha. DeathByCaptcha. http://www.deathbycaptcha.com/.

35. Freelancer. Freelancer. https://www.freelancer.co.uk/.

36. Motoyama M, Levchenko K, Kanich C, McCoy D, Voelker GM, Savage S. Re: CAPTCHAs-Understanding CAPTCHA-Solving Services in an Economic Context. In: *USENIX Security Symposium*. Vol 10. ; 2010:3.

37. Egele M, Bilge L, Kirda E, Kruegel C. Captcha smuggling: hijacking web browsing sessions to create captcha farms. In: *Proceedings of the 2010 ACM Symposium on Applied Computing*. ACM; 2010:1865-1870.

38. BBC. PC stripper helps spam to spread. http://news.bbc.co.uk/1/hi/technology/7067962.stm. Published 2007.

39. Van Oorschot PC, Stubblebine S. On countering online dictionary attacks with login histories and humans-in-the-loop. *ACM Trans Inf Syst Secur*. 2006;9(3):235-258.

40. Kanich C, Kreibich C, Levchenko K, et al. Spamalytics: An empirical analysis of spam marketing conversion. In: *Proceedings of the 15th ACM Conference on Computer and Communications Security*. ACM; 2008:3-14.

41. Zhu BB, Yan J, Bao G, Yang M, Xu N. Captcha as Graphical Passwords—A New Security Primitive Based on Hard AI Problems. *IEEE Trans Inf forensics Secur*. 2014;9(6):891-904.

42. Truong HD, Turner CF, Zou CC. iCAPTCHA: the next generation of CAPTCHA designed to defend against 3rd party human attacks. In: *Communications (ICC), 2011 IEEE International Conference on*. IEEE; 2011:1-6.

43. Mohamed M, Sachdeva N, Georgescu M, et al. A three-way investigation of a game-CAPTCHA: automated attacks, relay attacks and usability. In: *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*. ACM; 2014:195-206.

44. Gao S, Mohamed M, Sachdeva N, et al. Three-way dissection of a game-captcha: Automated attacks, relay attacks, and usability. *arXiv Prepr arXiv13101540*. 2013.

45. Chellapilla K, Simard PY. Using Machine Learning to Break Visual Human

Interaction Proofs (HIPs). In: *NIPS*. ; 2004.

46. Yan J, El Ahmad AS. Breaking visual captchas with naive pattern recognition algorithms. In: *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*. IEEE; 2007:279-291.

47. Shet V. Google Online Security Blog: Street View and reCAPTCHA technology just got smarter. Google. https://security.googleblog.com/2014/04/street-view-and-recaptcha-technology.html. Published 2014. Accessed May 10, 2014.

48. Sermanet P, Chintala S, LeCun Y. Convolutional neural networks applied to house numbers digit classification. In: *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE; 2012:3288-3291.

49. Moy G, Jones N, Harkless C, Potter R. Distortion estimation techniques in solving visual CAPTCHAs. In: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. Vol 2. IEEE; 2004:II-23.

50. Goodfellow IJ, Bulatov Y, Ibarz J, Arnoud S, Shet V. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv Prepr arXiv13126082*. 2013.

51. Yan J, El Ahmad AS. A Low-cost Attack on a Microsoft CAPTCHA. In: *Proceedings of the 15th ACM Conference on Computer and Communications Security*. ACM; 2008:543-554.

52. El Ahmad AS, Yan J, Tayara M. The Robustness of Google CAPTCHAs. *Comput Sci Newcastle Univ*. 2011. http://www.cs.ncl.ac.uk/publications/trs/papers/1278.pdf.

53. Ahmad ASE, Yan J, Marshall L. The robustness of a new CAPTCHA. *EUROSEC*. 2010:36-41.

54. Huang S-Y, Lee Y-K, Bell G, Ou Z. A projection-based segmentation algorithm for breaking MSN and YAHOO CAPTCHAs. In: *ICSIE'08: Proceedings of the 2008 International Conference of Signal and Image Engineering*. ; 2008.

55. Gao H, Wang W, Qi J, Wang X, Liu X, Yan J. The Robustness of Hollow CAPTCHAs. In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. CCS '13. New York, NY, USA: ACM; 2013:1075-1086. doi:10.1145/2508859.2516732.

56. Gao H, Wang W, Fan Y, Qi J, Liu X. The Robustness of" Connecting Characters Together" CAPTCHAs. *J Inf Sci Eng*. 2014;30(2):347-369.

57. Houck CW. Decoding reCAPTCHA. In: *DEFCON18*. ; 2010.

58. Cruz-Perez C, Starostenko O, Uceda-Ponga F, Alarcon-Aquino V, Reyes-Cabrera L. Breaking recaptchas with unpredictable collapse: heuristic character segmentation and recognition. In: *Pattern Recognition*. Springer; 2012:155-165.

59. Bursztein E, Martin M, Mitchell J. Text-based CAPTCHA strengths and weaknesses. In: *Proceedings of the 18th ACM Conference on Computer and Communications Security*. ACM; 2011:125-138.

60. Hong C, Lopez-Pineda B, Recasens A, Rajendran K. Breaking Microsoft's CAPTCHA. 2015.

61. Deng L, Yu D. *Deep Learning: Methods and Applications*. NOW Publishers; 2014. https://www.microsoft.com/en-us/research/publication/deep-learning-methods-and-applications/.

62. LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature*. 2015;521(7553):436-444.

63. Fukushima K, Miyake S. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In: *Competition and Cooperation in Neural Nets*. Springer; 1982:267-285.

64. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE*. Vol 86. ; 1998:2278-2324.

65. Stark F, Hazırbas C, Triebel R, Cremers D. Captcha recognition with active deep learning. In: *Workshop New Challenges in Neural Computation 2015*. Citeseer; 2015:94.

66. Wang K, Babenko B, Belongie S. End-to-end scene text recognition. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE; 2011:1457-1464.

67. Wang T, Wu DJ, Coates A, Ng AY. End-to-end text recognition with convolutional neural networks. In: *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE; 2012:3304-3308.

68. Novikova T, Barinova O, Kohli P, Lempitsky V. Large-lexicon attribute-consistent text recognition in natural images. In: *European Conference on Computer Vision*. Springer; 2012:752-765.

69. Mishra A, Alahari K, Jawahar C V. Scene text recognition using higher order language priors. In: *BMVC 2012-23rd British Machine Vision Conference*. BMVA; 2012.

70. Rodriguez-Serrano JA, Perronnin F, Meylan F. Label embedding for text recognition. In: *Proceedings of the British Machine Vision Conference*. ; 2013.

71. Goel V, Mishra A, Alahari K, Jawahar C V. Whole is greater than sum of parts: Recognizing scene text words. In: *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*. IEEE; 2013:398-402.

72. Bissacco A, Cummins M, Netzer Y, Neven H. Photoocr: Reading text in uncontrolled conditions. In: *Proceedings of the IEEE International Conference on Computer Vision*. ; 2013:785-792.

73. Alsharif O, Pineau J. End-to-end text recognition with hybrid HMM maxout models. *arXiv Prepr arXiv13101811*. 2013.

74. Almazán J, Gordo A, Fornés A, Valveny E. Word spotting and recognition with embedded attributes. *IEEE Trans Pattern Anal Mach Intell*. 2014;36(12):2552-2566.

75. Yao C, Bai X, Shi B, Liu W. Strokelets: A learned multi-scale representation for scene text recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. ; 2014:4042-4049.

76. Jaderberg M, Simonyan K, Vedaldi A, Zisserman A. Synthetic data and artificial

neural networks for natural scene text recognition. *arXiv Prepr arXiv14062227*. 2014.

77. Jaderberg M, Simonyan K, Vedaldi A, Zisserman A. Reading Text in the Wild with Convolutional Neural Networks. *arXiv Prepr arXiv14121842*. 2014.

78. Jaderberg M, Vedaldi A, Zisserman A. Deep features for text spotting. In: *European Conference on Computer Vision*. Springer; 2014:512-528.

79. Bursztein E, Aigrain J, Moscicki A, Mitchell JC. The end is nigh: generic solving of text-based CAPTCHAs. In: *Proceedings of the 8th USENIX Conference on Offensive Technologies*. USENIX Association; 2014:3.

80. Yan J. A simple generic attack on text captchas. 2016.

81. Athanasopoulos E, Antonatos S. Enhanced captchas: Using animation to tell humans and computers apart. In: *IFIP International Conference on Communications and Multimedia Security*. Springer; 2006:97-108.

82. Cui JS, Mei JT, Zhang WZ, Wang X, Zhang D. A CAPTCHA implementation based on moving objects recognition problem. In: *Proceedings of the International Conference on E-Business and E-Government, ICEE 2010*. ; 2010:1277-1280. doi:10.1109/ICEE.2010.326.

83. Dracon. Dracon CAPTCHA. Dracon. http://www.dracon.biz/captcha.php. Published 2005.

84. AmourAngels. AmourAngels. http://members.amourangels.com/cgi-bin/login.cgi.

85. CAPTCHANIM. CAPTCHANIM. http://captchanim.cs.technion.ac.il/.

86. NotOneBit. KillBot Project. http://www.notonebit.com/projects/killbot/.

87. NuCAPTCHA. NuCAPTCHA. http://www.nucaptcha.com.

88. Atlantis-Caps. Atlantis-Caps. http://www.atlantis-caps.com/eng/7-3-contacts.php.

89. Chow Y-W, Susilo W. AniCAP: an animated 3d CAPTCHA scheme based on motion parallax. In: *CANS'11 Proceedings of the 10th International Conference on*

Cryptology and Network Security. Berlin; 2011.

90. Kund I. NON-STANDARD CAPTCHAS FOR THE WEB: A MOTION BASED CHARACTER RECOGNITION HIP. 2011.

91. Nguyen VD, Chow Y-W, Susilo W. Attacking animated CAPTCHAs via character extraction. In: *Cryptology and Network Security*. Springer; 2012:98-113.

92. Nguyen VD, Chow Y-W, Susilo W. Breaking an Animated CAPTCHA Scheme. In: *ACNS'12 Proceedings of the 10th International Conference on Applied Cryptography and Network Security*. Berlin; 2012.

93. Giasson M, Bailey C, Moravek R, Campeau J, Bains P. NUCAPTCHA & TRADITIONAL CAPTCHA. *NuCAPTCHA*. 2011. http://www.nucaptcha.com/nucaptcha-vs-traditional-captcha-whitepaper.pdf.

94. Bursztein E. How we broke the NuCaptcha video scheme and what we propose to fix it. http://elie.im/blog/security/how-we-broke-the-nucaptcha-video-scheme-and-what-we-propose-to-fix-it/. Published 2012.

95. Lowe DG. Object recognition from local scale-invariant features. In: *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. Vol 2. Ieee; 1999:1150-1157.

96. Harris C, Stephens M. A combined corner and edge detector. In: *Alvey Vision Conference*. Vol 15. Citeseer; 1988:50.

97. Lucas BD, Kanade T. An iterative image registration technique with an application to stereo vision. In: *IJCAI*. Vol 81. ; 1981:674-679.

98. Fredman N, Russell S. Image segmentation in video sequences. In: *Proc. of the 13th Conf. on Uncertainty in Artificial Intelligence. Providence, Rhode Island, USA:[sn]*. ; 1997.

99. Jain AK, Murty MN, Flynn PJ. Data clustering: a review. *ACM Comput Surv*. 1999;31(3):264-323.

100. Kund I. the non-standard CAPTCHA. Kund's CAPTCHA.

http://captcha.ivokund.eu/Tests/. Accessed February 1, 2015.

101. Jakob Nielsen. Usability 101: Introduction to Usability. https://www.nngroup.com/articles/usability-101-introduction-to-usability/. Published 2012. Accessed March 27, 2017.

102. Chellapilla K, Larson K, Simard P, Czerwinski M. Designing human friendly human interaction proofs (HIPs). In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM; 2005:711-720.

103. Yan J, El Ahmad AS. Usability of CAPTCHAs or Usability Issues in CAPTCHA Design. In: *Proceedings of the 4th Symposium on Usable Privacy and Security*. SOUPS '08. New York, NY, USA: ACM; 2008:44-52. doi:10.1145/1408664.1408671.

104. Chew M, Baird HS. Baffletext: A human interactive proof. In: *Electronic Imaging 2003*. International Society for Optics and Photonics; 2003:305-316.

105. Mori G, Malik J. Recognizing objects in adversarial clutter: breaking a visual CAPTCHA. *2003 IEEE Comput Soc Conf Comput Vis Pattern Recognition, 2003 Proceedings*. 2003;1. doi:10.1109/CVPR.2003.1211347.

106. Inaccessibility of CAPTCHA: Alternatives to Visual Turing Tests on the Web. *W C W Gr*. 2005.

107. Converse T. CAPTCHA generation as a web service. In: *Human Interactive Proofs*. Springer; 2005:82-96.

108. El Ahmad AS, Yan J, Ng W-Y. CAPTCHA design: Color, usability, and security. *IEEE Internet Comput*. 2012;16(2):44-51.

109. Brodić D, Petrovska S, Jevtić M, Milivojević ZN. The influence of the CAPTCHA types to its solving times. In: *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2016 39th International Convention on*. IEEE; 2016:1274-1277.

110. D'haene F. Convolutional Neural Network Workbench.

http://www.codeproject.com/Articles/140631/Convolutional-Neural-Network-MNIST-Workbench. Published 2010.

111. Wikipedia. Wikipedia CAPTCHA. https://en.wikipedia.org/w/index.php?title=Special:CreateAccount&returnto=Main+Page.

112. Von Ahn L, Blum M. The reCAPTCHA Project. The reCAPTCHA Project. https://www.cylab.cmu.edu/partners/success-stories/recaptcha.html. Published 2000.

113. Duda RO, Hart PE. Use of the Hough transformation to detect lines and curves in pictures. *Commun ACM*. 1972;15(1):11-15.

114. ReCAPTCHA. reCAPTCHA: Easy on Humans, Hard on Bots. Google. https://www.google.com/recaptcha/intro/. Accessed March 16, 2017.

115. Wilkins J. Strong CAPTCHA Guidelines v1.2. *Retrieved Nov*. 2009;10:1-18.

116. IE-NuCAPTCHA. Xu's NuCaptcha. http://www.cs.unc.edu/videocaptcha/.

117. Yahoo. Yahoo CAPTCHA. https://uk.yahoo.com/. Accessed March 6, 2017.

118. Marques O. Geometric Operations. In: *Practical Image and Video Processing Using MATLAB*. ; 2011:151-170. doi:10.1002/9781118093467.ch7.

119. Lee ETY. Choosing nodes in parametric curve interpolation. *Comput Des*. 1989;21(6):363-370.

120. Warburton T. Morph I. RICE. http://www.caam.rice.edu/~timwar/CAAM210/Morph_I.html. Published 2008. Accessed March 14, 2015.

121. Weir CS, Douglas G, Carruthers M, Jack M. User perceptions of security, convenience and usability for ebanking authentication tokens. *Comput Secur*. 2009;28(1):47-62.

122. Shet V. No CAPTCHA reCAPTCHA. https://security.googleblog.com/2014/12/are-

you-robot-introducing-no-captcha.html.

123. Sivakorn S, Polakis J, Keromytis AD. I’m not a human: Breaking the Google reCAPTCHA.