

Strength Evaluation of Visual Text CAPTCHAs



By
Yousra Javed
2009-NUST-MS-CSE(S)-17

Supervisor
Dr. Syed Ali Khayam
NUST-SEECS

A thesis submitted in partial fulfillment of the requirements for the degree
of Masters in Communication Systems Engineering (MS CSE)

In
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
Islamabad, Pakistan.

(November 2011)

Approval

Certified that the contents and form of thesis entitled “**Strength Evaluation of Visual Text CAPTCHAs**” submitted by Ms. **Yousra Javed** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Syed Ali Khayam**

Signature: _____

Date: _____

Committee Member1: **Dr. M. Murtaza Khan**

Signature: _____

Date: _____

Committee Member2: **Dr. Fauzan Mirza**

Signature: _____

Date: _____

Committee Member3: **Dr. Khawar Khurshid**

Signature: _____

Date: _____

Certificate

Certified that the Scrutinizing Committee has reviewed the final documentation of Ms. **Yousra Javed** Regno **2009-NUST-MS-CSE(S)-17** Student of **MS-CSE2** thesis title **Strength Evaluation of Visual Text CAPTCHAs** and found it satisfactory as per NUST's standard format for Master Thesis.

President

Wg Cdr (R) Muhammad Ramzan

Abstract

CAPTCHAs (Completely Automated Public Turing test to Tell Computers and Humans Apart) are used to prevent attacks by malicious automated programs that try to register for or disrupt online services. The basic requirement of a CAPTCHA is that humans should be able to interpret it easily, whereas computer programs should not be able to interpret it with the same ease. Due to their simple design, CAPTCHAs have found pervasive applications in different web based systems, including sensitive systems such as email services, e-banking systems, and online voting forums.

The most widely-used CAPTCHAs today are visual text based CAPTCHAs, which present the user with a distorted text image and require him/her to enter the displayed text. The text appearing in a CAPTCHA is distorted in such a way that, while a human is capable of understanding it, it is not possible (in theory) for an automated Optical Character Recognition (OCR) algorithm to read it. Many commercial CAPTCHAs have been defeated due to their ease of segmentation and recognition by current OCR systems. Therefore, the strength of current CAPTCHAs (such as those of Google and Yahoo!) lies in distorting and packing the characters so closely that the text is difficult. Therefore, we believe that a systematic way of evaluating both the security and usability of CAPTCHAs is very important. This will not only help designers but also the end user in carefully selecting a CAPTCHA scheme for his/her ser-

vice.

In this thesis, we focus on the semi-automated strength evaluation of visual text based CAPTCHAs. To accomplish this objective, a two phase process is undertaken. In the first phase, we estimate the level of difficulty in defeating contemporary CAPTCHAs (such as Google and Yahoo!) which contributes towards their security strength. For this purpose, we first explore the area of CAPTCHA generation and estimate the procedures followed by a CAPTCHA generation engine. Since the CAPTCHA generation algorithms are not known to public, very little knowledge currently exists about how texts CAPTCHAs are generated. We develop a CAPTCHA generation algorithm which is capable of generating CAPTCHAs similar to state-of-the-art systems.

Using the CAPTCHA generation knowledge, we devise a method to defeat Google CAPTCHAs through reverse engineering. This method achieves 20% correct character recognition accuracy. To the best of our knowledge, no public study has thus far been able to achieve such high character recognition accuracy against Google CAPTCHAs.

In the second phase, we develop two quantitative measures based on various geometric indicators to assess the strength of any visual text-based CAPTCHA. These measures are tested on a non-extensive but diverse CAPTCHA database and achieve over 80% accuracy in classifying a CAPTCHAs' visual hardness by a human user.

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgment has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: **Yousra Javed**

Signature: _____

Acknowledgments

No piece of work is ever exclusively the product of the individuals whose names appear on its title page. Inevitably, the brainchild of the authors bears the imprint of many forces and people.

First of all, I am grateful to Allah Almighty for providing me the opportunity to work in the unique and conducive atmosphere of NUST-SEECs and endowing me with the strength to complete this thesis.

I am indebted to my advisor, **Dr.Syed Ali Khayam**, and co-advisor, **Dr.Muhammad Murtaza Khan**, not only for rendering me their standards of excellence in guiding me throughout the thesis but also for keeping me motivated along the path . What I have achieved holds a huge credit to their apparition and vision.

I am also grateful to my committee members; Dr.Fauzan Mirza and Dr.Khawar Khurshid, for their encouragements and valuable suggestions throughout this thesis.

Lastly and very importantly I would like to thank my family; grandparents, parents and siblings for their love and prayers, friends and colleagues at SEECs (especially the WiSNet Lab) for their help, support and unremitting favours.

Yousra Javed

Table of Contents

1	Introduction and Motivation	1
1.1	Introduction	1
1.2	Contribution	3
1.3	Thesis Organization	4
2	Literature Review	6
2.1	Classification of CAPTCHA Schemes	6
2.1.1	Visual Text based	6
2.1.2	Image Based	10
2.1.3	Video based	11
2.1.4	Math Based	12
2.1.5	Audio Based	13
2.2	Defeating CAPTCHAs	14
2.2.1	Attacks on Text CAPTCHAs	15
2.2.2	Attacks on Image and Audio CAPTCHAs	15
2.2.3	Attacks on other CAPTCHA Schemes	16
2.2.4	Current trend	16
3	Text CAPTCHA Generation Algorithm	18
3.1	Methodology	18
3.1.1	Morphing	18
3.1.2	Packing	23
3.2	Results	24

4	Reverse Engineering of Text CAPTCHAs	27
4.1	Methodology	27
4.1.1	Inverse map calculation	27
4.1.2	Distortion removal from Google CAPTCHAs	28
4.2	Offline attack on Google CAPTCHAs	28
4.2.1	Divide and Conquer	30
4.2.2	Inverse map application	30
4.2.3	Character recognition using OCR tool	30
4.2.4	Voting for the final recognition result	31
4.2.5	Attack Results	32
5	Strength Estimation Measures	33
5.1	Geometric Indicators	33
5.1.1	Shape Compactness	34
5.1.2	Euler's Number	34
5.1.3	Thickness/Boldness	35
5.1.4	Compactness-Length Ratio	36
5.1.5	Euler's Number-Thickness Ratio	36
5.2	Experimental Results	36
5.2.1	User Studies	37
5.2.2	Neural Network Classification	38
6	Conclusion and Future Work	40
6.1	Conclusions and Summary	40
6.2	Future Work	41

List of Figures

1.1	Three hard Google CAPTCHA images: (a) “an- termenct” or “anterrnenct”? (b) “nomorinen” or “nomormen”? (c) “smomtotmno” or “smorn- totrnno”?	2
1.2	Some selected weak e-banking CAPTCHAs bro- ken in [7].	3
2.1	Types of Visual Text CAPTCHAs (a) Gimpy (b) Ez-Gimpy (c) BaffleText (d) Yahoo (e) Microsoft (f) re-CAPTCHA (g) Google	9
2.2	Types of Image CAPTCHAs	12
2.3	Types of Video CAPTCHAs	12
2.4	Types of Math CAPTCHAs	13
2.5	CAPTCHA solving market workflow	17
3.1	Original pixel grid with the original 16 points specified	21
3.2	Morphed characters formed using the maps . . .	22
3.3	Packed Characters	24
3.4	Generated CAPTCHAs using packing first and morphing afterwards	25
3.5	Generated CAPTCHAs using morphing first and packing afterwards	25

4.1	Selected Google CAPTCHAs and their corresponding unmorphed images using inverse bezier transformation	29
5.1	Compactness (Cn) values of an easy and hard Google CAPTCHA respectively	34
5.2	Euler Number (EN) values of an easy and hard Google CAPTCHA respectively	35
5.3	Number of Erosion Steps (ES) of two Google CAPTCHAs	35
5.4	Objective hardness indicators of selected CAPTCHAs of four different CAPTCHA schemes	37
5.5	User study for training the classifier: training data (left) and testing data (right)	39

List of Tables

6.1	Maps for Control points	42
-----	-----------------------------------	----

Chapter 1

Introduction and Motivation

1.1 Introduction

CAPTCHAs (Completely Automated Public Turing tests to Tell Computers and Humans Apart)[1] are now being deployed widely on the Internet to combat malicious activities by automated bots. Such malicious activities include commenting spam on blogs, automated website registration, online polls and dictionary attacks.

The advent of CAPTCHA generation algorithms has been driven by attempts of attackers to develop automated methods to defeat CAPTCHAs. Earlier CAPTCHA schemes e.g., Gimpy, EZ Gimpy, Captchaservice.org, BotCheck, BotBlock and Humanverify [2, 3, 4, 5, 6] were rather easily defeated since the text presented in them could be both segmented and recognized using simple image processing algorithms. In order to increase CAPTCHA security, algorithms have now evolved to the other extreme where the text has become extremely difficult to segment, even for human users. As an example, consider the CAPTCHAs of Figure 1.1 which are used by Google on its sign up page. The text letters in the CAPTCHA image are morphed and closely packed, which increases its security by making character segmentation and recognition difficult for the

attacker. This, on the other hand, induces considerable degradation in the usability of the CAPTCHA as well. For instance, an average user (i.e., neither an expert nor an elderly person with limited computer exposure) will encounter significant difficulty in recognizing the Google CAPTCHA images shown in Figure 1.1. To get around hard Google CAPTCHA images, users often simply refresh the Web page until an easy CAPTCHA image is displayed. In other words, the (stronger) CAPTCHA scheme is reduced to a subset of weaker CAPTCHA images.

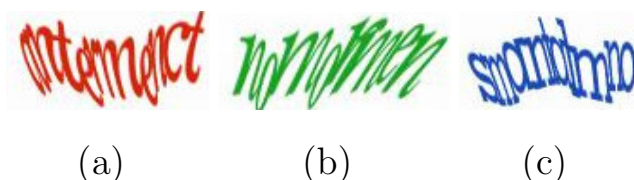


Figure 1.1: Three hard Google CAPTCHA images: (a) “antennent” or “anterrnenct”? (b) “nomorinen” or “nomormen”? (c) “smomtotmno” or “smorntotrno”?

Since it is extremely hard to find a balance between usability and strength, many web sites choose to deploy more usable but less secure CAPTCHAs. For example, consider the E-banking CAPTCHAs used by some Chinese banks [7] as shown in Figure 1.2. It is obvious from the figure, that these CAPTCHAs can easily be segmented and recognized by state-of-the-art Optical Character Recognition (OCR) tools.

The above problems signify the need for evaluating the usability and security of contemporary CAPTCHA schemes to assist the designers in improving them. So far, no work exists on automatic CAPTCHA evaluation. In this thesis, we focus on evaluating the security of visual text CAPTCHAs using semi-automated methods.

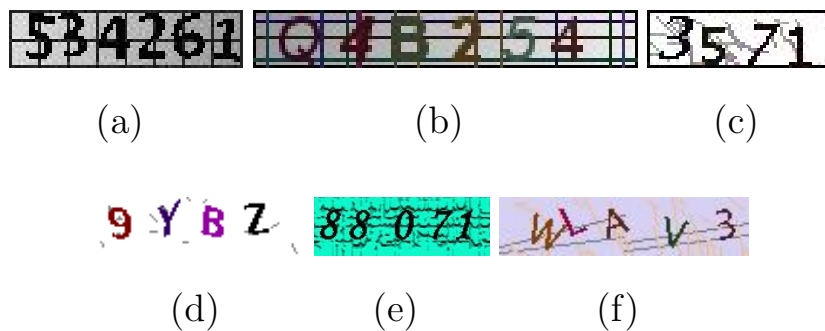


Figure 1.2: Some selected weak e-banking CAPTCHAs broken in [7].

1.2 Contribution

In this thesis, we develop a method for semi-automated strength evaluation of visual text based CAPTCHAs.

Two aspects are investigated:

1. Estimation of hardness in defeating contemporary text CAPTCHAs.
2. Formulation of geometric measures to estimate CAPTCHA strength.

Following are the main contributions of this thesis:

- *Design of a text CAPTCHA generation algorithm:* Since there is no publicly available report on how these CAPTCHAs are generated, we start off with building an algorithm that can generate text CAPTCHAs similar to contemporary CAPTCHA schemes (e.g., Google). This step also gives us an insight into defeating the deployed text CAPTCHA schemes.
- *Reverse engineering of CAPTCHAs:* In this step, we use the modules of our CAPTCHA generation algorithm to analyze if we can remove the text distortion in Google

CAPTCHAs and straighten the letters. We then use a state-of-the-art OCR tool, *Abby FineReader*, to recognize the text. Based on our findings, we launch an offline attack on Google CAPTCHAs and estimate the correct letter recognition accuracy of our approach.

- *Development of geometric strength indicators:* In this step, we automate the hardness evaluation of different kinds of textual CAPTCHAs as judged by an average user with normal eyesight (which is an indirect metric of security). We have formulated five geometric indicators based on which we classify a CAPTCHA as weak or hard, in terms of security.
- *User studies:* In the last phase, we carry out two user studies for rating various text CAPTCHAs on the basis of solving difficulty. The median subjective scores of the CAPTCHAs of one study are used to train an NN classifier. This classifier is then tested on CAPTCHAs of the second user study whose subjective scores are used to analyze the hardness estimation accuracy of our proposed geometric indicators.

All the algorithm implementation and experiments in this thesis have been done using MATLAB software.

1.3 Thesis Organization

The rest of the thesis is organized as follows:

Chapter 2 discusses the background of this domain, types of CAPTCHA schemes and the related work that has been done in this domain.

Chapter 3 provides the details of our text CAPTCHA generation algorithm and the resulting CAPTCHAs of our approach.

Chapter 4 illustrates the reverse engineering approach used in an attempt to defeat Google CAPTCHAs. The results of an offline attack launched in this regard are also discussed.

Geometric strength measures are discussed in chapter 5 with detailed discussion on the proposed geometric strength indicators. The details of user studies conducted to gather subjective scores of CAPTCHAs and the NN classifier used for testing the accuracy of strength estimation are also provided in this chapter.

Chapter 6 finally concludes the thesis and mentions our future work.

Chapter 2

Literature Review

This chapter discusses the various types of CAPTCHA schemes existing in literature and the existing work on defeating them.

2.1 Classification of CAPTCHA Schemes

CAPTCHAs are classified based on what is presented to the user as a challenge. They fall into the following five categories:

1. Visual Text based
2. Image based
3. Video based
4. Math based
5. Audio based

2.1.1 Visual Text based

In this scheme, the user is presented with an image containing a distorted text string and is required to recognize and enter the displayed letters. Some implementations of text based CAPTCHAs are shown in Figure 2.1 and their description is presented below.

a) *Gimpy*

Gimpy is the first ever visual text CAPTCHA implementation, built by Carnegie Mellon University in alliance with Yahoo for their Messenger service. Gimpy[8] forms a challenge by picking ten random dictionary words which are overlapped and distorted. The user is required to enter a given number of the words in the image for verification. The CAPTCHA is dependent on dictionary words making it prone to dictionary attacks. Moreover, the lack of proper distortion lead to its defeat by modern OCR systems.

b) *Ez-Gimpy*

A simplified version of the Gimpy CAPTCHA; Ez-Gimpy was adopted by Yahoo for their sign-up page. Ez-Gimpy[8] randomly picks a dictionary word and distorts it. The user is required to recognize and enter the text. This CAPTCHA has more character distortion compared to Gimpy but is weak due to the ease of segmentation.

c) *BaffleText*

BaffleText, is a variation of Ez-Gimpy. It picks up and distorts random alphabets instead of a dictionary word in order to combat dictionary attacks. The foreground is black and the background is white[9]. It exploits the fact that humans are very good at filling in missing portions of an image while computers are not.

d) *Yahoo*

Yahoo CAPTCHAs use five to eight characters consisting of upper and lower case letters and alphabets. The foreground color is black and the background color is white. Noise is added by placing arcs at random positions. This is a hard to segment CAPTCHA because of the addition of noise to distorted characters.

e) *MSN*

Microsoft CAPTCHAs use eight characters (upper case) and digits. Foreground color is dark blue, and background color is grey. The characters are distorted through warping which results in a ripple effect, making character recognition difficult [10]. Similar to Yahoo CAPTCHA, this CAPTCHA has a large space of characters to choose from and is hard to segment due to the noise.

f) *reCAPTCHA*

The idea behind reCAPTCHA was to digitize scanned books. Scanned words that cannot be recognized by sophisticated OCR techniques form a challenge by combining them with a control word whose content is known. The words are distorted and aligned randomly. reCAPTCHA accepts a solution if the control word is submitted correctly, and the text for the unknown word overlaps substantially with already submitted solutions for the same challenge[11]. Segmentation of reCAPTCHA is hard. Moreover, the OCR systems do not recognize these scanned words. There are legibility issues at times when the scanned words are severely blurred.

g) *Google*

The text CAPTCHAs used by Google display a random number of lower case alphabets. Foreground is red, blue or green and the background is white. The characters are distorted and packed very closely making segmentation hard [12]. Due to the close character packing, the CAPTCHA is hard to segment. On the other hand, this leads to severe usability issues at times as shown in Figure 2.1.

Text-based CAPTCHAs are the most widely deployed CAPTCHA schemes on Internet. Major web sites such as Google,

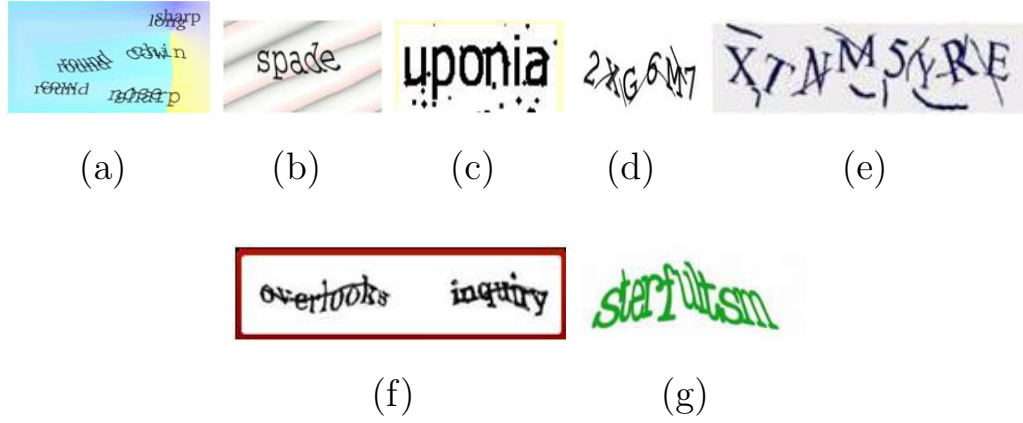


Figure 2.1: Types of Visual Text CAPTCHAs (a) Gimpy (b) Ez-Gimpy (c) BaffleText (d) Yahoo (e) Microsoft (f) re-CAPTCHA (g) Google

Yahoo and Microsoft all have their own text-based CAPTCHAs deployed for years. In this thesis, we focus on text-based schemes since they are the most popular because of the following reasons:

- With as little as 5 characters (case-insensitive letters and digits) one can have $36^5 = 60$ million possible combinations. These combinations are increased further, once the distortion is applied to the characters.
- They are easy to use; the user task is limited to character recognition. Therefore, the response time for an average text CAPTCHA is small. Moreover, they do not have localization issues; can be easily adopted by any language and culture.
- They have a potential for strong security since brute force attack is difficult and there is no database dependency.

2.1.2 Image Based

In this scheme, the user is either required to perform the task of image recognition or annotate some pictures. Various implementations of image based CAPTCHAs are as follows:

a) *Bongo*

The Bongo CAPTCHA displays two series of shapes. User must recognize the difference in the two shape series. For example, in Figure ??, one set of blocks has thick lines and the other has thin [13].

b) *Pix*

This scheme picks an object at random, searches its database for six pictures of that object, and presents them to the user. The user is required to name the object in the displayed pictures [14].

c) *Asirra*

ASIRRA scheme is based on the classification of displayed pictures. The goal is to identify all pictures of cats among the displayed images.[15]

d) *ArtiFacial*

This CAPTCHA displays various facial features in random positions. The user is required to click on 6 of these features [16].

e) *Whats Up*

Whats up CAPTCHA is a Google CAPTCHA which displays a set of images at various orientations. The user is asked to pick the images that are in an upright position. [17]

f) *Imagination*

Imagination is a click and annotate CAPTCHA. The challenge is formed by creating a composite image using various objects from the image database. The user is first required to click the geometric center of one the image. He is then displayed another image which he has to annotate using the sixteen options provided to him [18].

g) *Cortcha*

This CAPTCHA scheme presents the user with an image having a missing object. The user has to find the missing object from the set of object images provided to him. This CAPTCHA doesnt require any database; the challenge is created using a new image each time therefore, it combats dictionary attacks [19].

h) *Confident*

In this scheme, the user is displayed images of various objects. To pass the challenge, the user is required to recognize and click on the picture of a particular object [20].

i) *Yuniti*

This is a 3D CAPTCHA, in which the user is displayed three objects at a particular scale and orientation. User is asked to select the images of the same objects from the set of images at different scales and orientations [21].

2.1.3 Video based

In this scheme, the user is either required to tag a displayed video in one word or input the streaming text appearing in the video [22, 23].

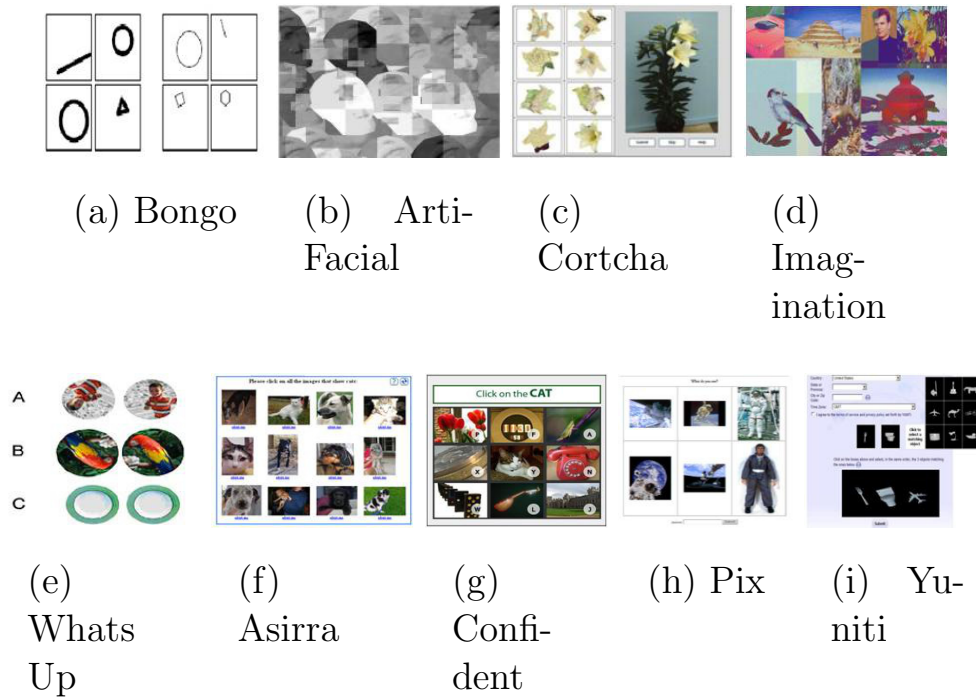


Figure 2.2: Types of Image CAPTCHAs

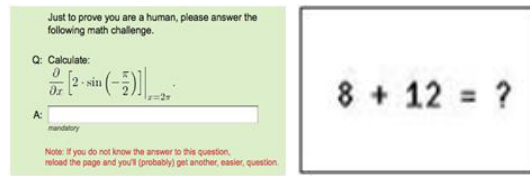


Figure 2.3: Types of Video CAPTCHAs

2.1.4 Math Based

In this scheme, the user is presented with a mathematical problem ranging from simple addition and multiplication to complex calculus i.e., differentiation and integration. The user is required

to solve the math problem [24].



(a) Calculus

(b) Sum

Figure 2.4: Types of Math CAPTCHAs

2.1.5 Audio Based

In this CAPTCHA scheme, a word or a sequence of random numbers is picked and their distorted audio clip is created. The user is required to identify and type the word or numbers. This type of CAPTCHA is mainly meant for users who have difficulty in reading and is presented as an option alongside visual text CAPTCHAs. Various implementations of audio based CAPTCHAs are as follows:

a) *Digg*

Audio CAPTCHAs on digg.com consist of a female voice speaking aloud five letters. There is heavy white noise in the background, and sometimes an empty but louder segment is played between letters.

b) *eBay*

Audio CAPTCHAs on ebay.com consist of the same six digits from the image CAPTCHAs being spoken aloud, each by a different speaker.

c) *Google*

Audio CAPTCHAs on google.com consist of three beeps, a male voice speaking digits aloud, the phrase “once again”, and a repeat of the male voice speaking the digits. In the background, various voices are played simultaneously, and confusing decoy words like “now” or “it” are occasionally interjected [12].

d) *Microsoft*

Audio CAPTCHAs on live.com consist of ten digits being spoken aloud, each by a different speaker over a low quality recording, with various voices playing simultaneously in the background [10].

e) *Recaptcha*

Audio CAPTCHAs from recaptcha.net consist of eight digits spoken by different speakers, with voices in the background and occasional confusing words interjected. This is similar to the live.com presentation, but the digits are delivered much more slowly [11].

f) *Yahoo*

Audio CAPTCHAs from yahoo.com consist of three beeps and then a child's voice speaking seven digits with various other child voices in the background.

2.2 Defeating CAPTCHAs

In this section, the existing work in the domain of CAPTCHA breaking has been discussed. Attacks on all types of CAPTCHAs have been detailed along with the current trend in the area.

2.2.1 Attacks on Text CAPTCHAs

The initial text CAPTCHAs were easily segmentable. In 2003, recognition based attacks were proposed to defeat two initial distorted text CAPTCHA schemes i.e., Gimpy and EZ-Gimpy [2]. Later, several other attacks were proposed for other CAPTCHAs highlighting the insecurity of distorted text CAPTCHA schemes [3, 25]. Figure 5 shows some easy to segment CAPTCHAs. In [5], an interesting finding was reported: if the text CAPTCHA image characters are well segmented, OCR tools can recognize them even better than humans. Therefore, making character segmentation harder is the only way to improve the security of distortion based text CAPTCHAs. In [4] Yan et al proved that some CAPTCHAs at some Web sites can be easily defeated using segmentation and pixel-count based attack. In [6], Yan et al proposed a new attack on some visual text CAPTCHAs, that can segment CAPTCHA images into characters with high accuracy. This lead to the creation of newer CAPTCHAs that are harder to segment e.g., Google and Microsoft CAPTCHAs shown in Figure 2.1.

2.2.2 Attacks on Image and Audio CAPTCHAs

Although most of the existing attacks have been launched on visual text CAPTCHAs. But attacks on image and audio CAPTCHA schemes were also launched. In [26], Golle demonstrated that an attack based on machine learning can achieve a success rate of 10.3% on the image-based CAPTCHA scheme Asirra[15]. In [27, 28], machine learning attacks on some deployed audio CAPTCHA schemes were reported. Audio CAPTCHAs e.g., ebay were broken by similar machine learning attack with high accuracy.

2.2.3 Attacks on other CAPTCHA Schemes

Attacks on other CAPTCHA schemes exploit design and implementation flaws. In [24], a side-channel attack was proposed on a math CAPTCHA scheme based on solving calculus problems. At IFIPTM2007, Caine and Hengartner found out a fault in CAPTCHA protocol design, which facilitated the attacker in obtaining multiple CAPTCHA images for a single text string. The results of all these images were voted for guessing the actual text. In [29] Hindle et al. demonstrated new attacks based on a cracked CAPTCHA generator can be designed using reverse engineering. At iNetSec2009, Hernandez-Castro and Ribagorda pointed out many common problems and design pitfalls of different CAPTCHA schemes.

2.2.4 Current trend

The existing CAPTCHAs are very hard to segment and recognize by OCR systems. Therefore, the attackers have now shifted their focus to human solvers instead of defeating them. These human solvers are paid for every CAPTCHA they solve, therefore, facilitating the attacker [31]. The CAPTCHA solving market workflow is shown in Figure 2.5. In the first step, an automated program attempts to register a Gmail account and is challenged with a Google CAPTCHA. This program uses the CAPTCHA solving plug-in to solve the CAPTCHA at a fixed rate for a certain number of CAPTCHAs. This plug-in queues the CAPTCHA for a human worker paid at a particular rate. The worker enters a solution which is returned to the plug-in. The automated program then enters the solution for the CAPTCHA to Gmail to register the account.

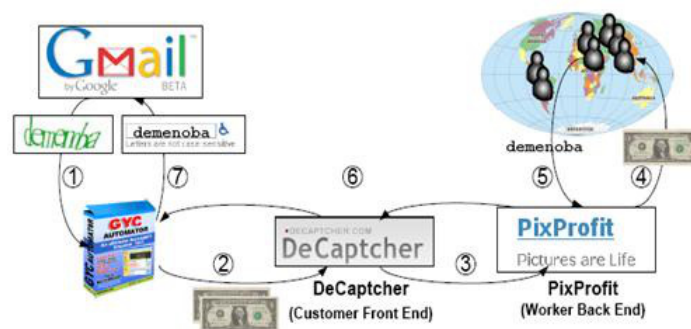


Figure 2.5: CAPTCHA solving market workflow

Chapter 3

Text CAPTCHA Generation Algorithm

In this section, we explain our ideas and how they led us to approximate the actual approach followed in CAPTCHA generation. The algorithm was implemented using MATLAB software.

3.1 Methodology

Based upon the analysis of Google CAPTCHAs, we generate CAPTCHA images by a permutation of two basic steps:

1. Morphing
2. Packing

Morphing is used to add distortion to the text. For this purpose, we implement the Bezier surface approach [32]. Packing is done to bring the characters as close as possible in order to strengthen the CAPTCHA against segmentation, avoiding any extreme overlapping simultaneously.

3.1.1 Morphing

To accomplish text distortion, we test the bezier surface approximation on single character images. Bezier surface determines

how an image would be morphed if the given set of control points in the 2d image space are changed from their initial position. Its implementation for cubic surface gives control over 16 points which we assume should be enough for deforming an image to obtain the morphed characters as they appear in the Google or Yahoo CAPTCHAs. Thus, we choose to use the cubic surface.

Different standard grid maps of these control points (the new location of the pixel) can be formed using the 16 points. These grid maps are applied to an image to get a particular deformation. Following is a step by step discussion of the procedure:

a) *Loading and Cropping*

An image is first loaded and binarized. It is then cropped from all sides so that the image encloses only the text.

b) *Original coordinates and Map Formation*

Using third order Bezier surface we have 16 control points. Firstly we need to save the original position of the control points in the original image. Figure 3.1 shows the position of the original pixels. Let ϕ represent a matrix containing the original pixel locations.

$$\phi_x = \begin{bmatrix} a_x & b_x & c_x & d_x \\ a_x & b_x & c_x & d_x \\ a_x & b_x & c_x & d_x \\ a_x & b_x & c_x & d_x \end{bmatrix} \quad (3.1)$$

$$\phi_y = \begin{bmatrix} a_y & a_y & a_y & a_y \\ b_y & b_y & b_y & b_y \\ c_y & c_y & c_y & c_y \\ d_y & d_y & d_y & d_y \end{bmatrix} \quad (3.2)$$

where ϕ_x and ϕ_y represent the x and y coordinate of the original pixel location respectively.

μ represents the matrix containing the new pixel locations, with μ represents the matrix containing the new pixel locations, with

$$\mu_x = \begin{bmatrix} A_x & B_x & C_x & D_x \\ E_x & F_x & G_x & H_x \\ I_x & J_x & K_x & L_x \\ M_x & N_x & O_x & P_x \end{bmatrix} \quad (3.3)$$

$$\mu_y = \begin{bmatrix} A_y & B_y & C_y & D_y \\ E_y & F_y & G_y & H_y \\ I_y & J_y & K_y & L_y \\ M_y & N_y & O_y & P_y \end{bmatrix} \quad (3.4)$$

where μ_x and μ_y represent the x and y coordinate of the new pixel locations respectively.

Using the original positions we have formed maps which contain the 16 new locations on which the original coordinates are to be shifted. These maps are shown in Table 6.1. In order to finish the dependency on fixed map database, we automated the map calculation process so as to randomly calculate the new location of each control point from the range in which it can move.

c) *Surface Formation*

Using the 16 new locations the new x and y position of every pixel of the original image is computed using

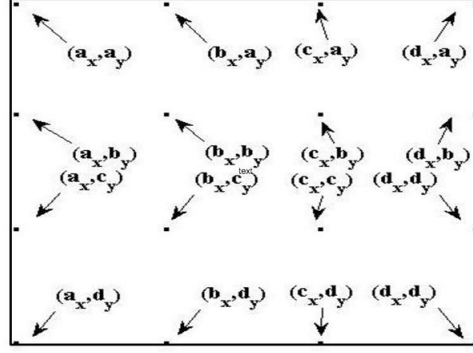


Figure 3.1: Original pixel grid with the original 16 points specified

$$\begin{aligned}
 X(a, c) = & A_x a^3 c^3 + B_x 3a^3 c^2 d + C_x 3a^3 c d^2 \\
 & + D_x a^3 d^3 + E_x 3a^2 b c^3 + F_x 9a^2 b c^2 d \\
 & + G_x 9a^2 b c d^2 + H_x 3a^2 b d^3 + I_x 3a b^2 c^3 \\
 & + J_x 9a b^2 c^2 d + K_x 9a b^2 c d^2 + L_x 3a b^2 d^3 \\
 & + M_x b^3 c^3 + N_x 3b^3 c^2 d + O_x 3b^3 c d^2 \\
 & + P_x b^3 d^3
 \end{aligned} \tag{3.5}$$

$$\begin{aligned}
 Y(a, c) = & A_y a^3 c^3 + B_y 3a^3 c^2 d + C_y 3a^3 c d^2 \\
 & + D_y a^3 d^3 + E_y 3a^2 b c^3 + F_y 9a^2 b c^2 d \\
 & + G_y 9a^2 b c d^2 + H_y 3a^2 b d^3 + I_y 3a b^2 c^3 \\
 & + J_y 9a b^2 c^2 d + K_y 9a b^2 c d^2 + L_y 3a b^2 d^3 \\
 & + M_y b^3 c^3 + N_y 3b^3 c^2 d + O_y 3b^3 c d^2 \\
 & + P_y b^3 d^3
 \end{aligned} \tag{3.6}$$

Where a and c is the location of pixels in x and y coordinate normalized over the height and width of the image respectively. Whereas $b = 1 - a$ and $d = 1 - c$.

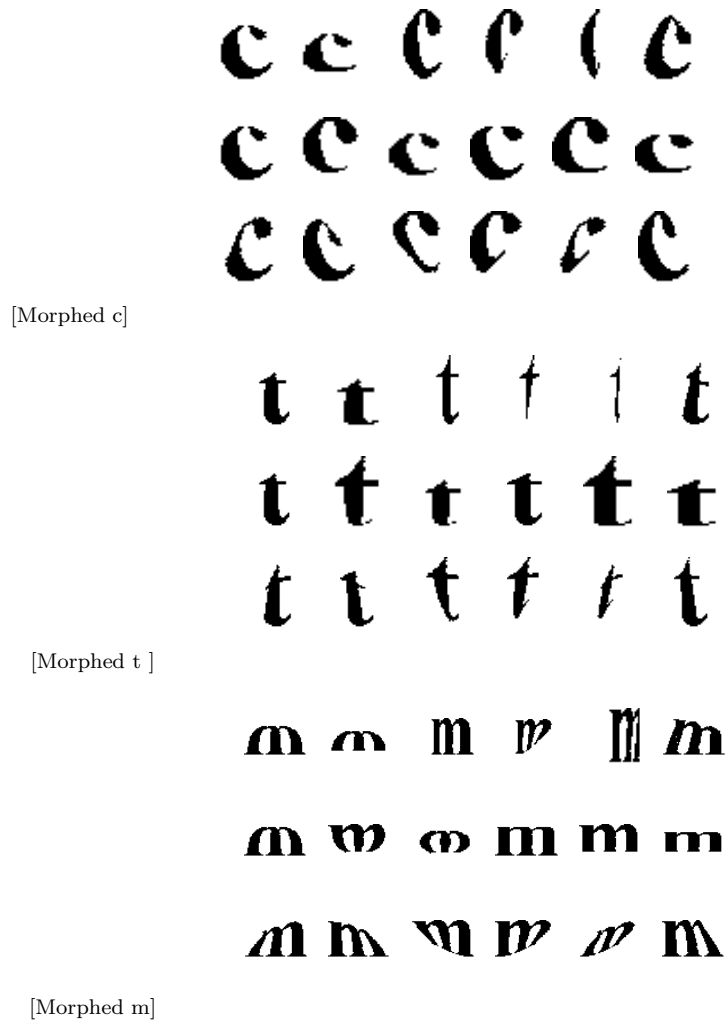


Figure 3.2: Morphed characters formed using the maps

d) *Translation of coordinate points*

Now a new grid is defined, the height of the grid is equal to the maximum value of $Y(a, c)$ and the width is equal to the maximum value of $X(a, c)$. Using the maps shown in Table 6.1 we have generated some morphed characters shown in Figure 3.2.

3.1.2 Packing

In order to mosaic the characters together, we have used the following heuristic. We form polygons of the text characters and slide each subsequent polygon over the already packed ones until we get no overlapping. The details of our approach are as follows.

a) *Polygon formation*

Using each text character's image, we calculate the character's boundary coordinates and join them to form a polygon. A random dc offset is then added to the x coordinates of the polygon to shift it in up or down direction.

b) *Overlapping*

We fix the first character's polygon and starting from the second character onwards, we slide each polygon from the origin, assuming the origin to be the start of the first character's polygon. Taking two polygons at a time, we check if the coordinates of the second polygon lie in or on the first polygon (i.e., do they overlap or not). If there is an overlapping, we increment the y coordinates of the current to-be packed polygon. In other words, we shift the polygon to the right. We repeat this procedure all over again, with the new coordinates until we get no overlapping. The whole process is repeated for each character in the text. Our packing approach is summarized in Algorithm 1.

After packing the polygons, each character's pixels are written to a new image according to its new polygon coordinates. Figure 3.3 shows some characters packed using Algorithm 1.

Algorithm 1: Packing characters

Input: Original Character Polygons P , Number of characters N
Output: Packed Character Polygons P_p

```

1  $i$  is the to-be packed Polygon number
   $j$  is the Polygon number to check overlapping with
   $k$  is the Polygon coordinate number

2 foreach  $i = 2$  to  $N$  do
3   foreach  $j = 1$  to  $i - 1$  do
4     foreach  $k = 1$  to  $\text{length}(P[i])$  do
5       Check whether coordinate  $k$  of polygon  $i$  lie inside polygon  $j$ 
6       if ( $\text{InPolygon}((P_x^k[i], P_y^k[i]), P[j]) == 1$ ) then
7          $P_y^k[i] = P_y^k[i] + 1$ 
8       end
9     end
10    foreach  $k = 1$  to  $\text{length}(P[j])$  do
11      Check whether coordinate  $k$  of polygon  $j$  lie inside polygon  $i$ 
12      if ( $\text{InPolygon}((P_x^k[j], P_y^k[j]), P[i]) == 1$ ) then
13         $P_y^k[i] = P_y^k[i] + 1$ 
14      end
15    end
16  end
17   $P_p[i] = P[i]$ 
18 end

```

Figure 3.3: Packed Characters

3.2 Results

In order to see which method gives better results, we tested the following permutations of the basic steps described above.



Figure 3.4: Generated CAPTCHAs using packing first and morphing afterwards



Figure 3.5: Generated CAPTCHAs using morphing first and packing afterwards

1. *Morphing the characters first and packing them afterwards:* Each character's image is first morphed using a random bezier map. These morphed character images are then packed close together.
2. *Packing the characters first and applying morphing afterwards:* The unmorphed character images are first packed together into a single image. This packed image is then morphed using a single random bezier map.

Figure 3.4 and 3.5 show the resulting CAPTCHAs of both permutations. We observed that the second approach i.e., packing the characters first and morphing them afterwards results in a stronger deformation. Moreover, it requires adding distortion using only one bezier map after packing the character images.

Since the initial characters were small, the effect of morphing was less pronounced on a single character image, whereas for the complete image, the morphing resulted in a desirable distortion. The distortion i.e., movement of each of the sixteen control points may be in different directions due to which a different type of distortion results on each part of the packed image. Therefore, we selected the second approach for our experiments.

Chapter 4

Reverse Engineering of Text CAPTCHAs

After successfully building our own CAPTCHA generation algorithm, the next step is to test its effectiveness in defeating existing deployed text CAPTCHA schemes. We start off with Google CAPTCHAs [12] as they are widely considered the most difficult ones to break because of their extreme morphing and compactness.

4.1 Methodology

Our approach is based on using different inverse transformations assuming that one of these transformations might give us an unmorphed CAPTCHA which might be readable when presented to an OCR tool. Step-by-step details of the procedure are presented below:

4.1.1 Inverse map calculation

In order to obtain a relatively less distorted image from the deformed image, we automated a procedure to create a fixed size 2D inverse mapping matrix for each bezier control map. This mapping matrix was created while forming bezier surface

corresponding to this control map. It was done by storing the new coordinates of each pixel in a matrix at the pixels original coordinates.

The x coordinates of each pixel were stored in the first dimension of this matrix, while the y coordinates were stored in the second dimension. The undistorted image was therefore created by assigning values to the pixels original coordinates corresponding to the shifted coordinate locations saved in the matrix at that index. The size of each mapping matrix was fixed to $[200 \times 70]$, equal to the size of Google CAPTCHA images. We obtained inverse maps for each of the 500 forward bezier maps.

4.1.2 Distortion removal from Google CAPTCHAs

Since, there is no information available on the distortion techniques used by contemporary CAPTCHAs; we tested our inverse mapping module on Google CAPTCHAs to analyze if we can modify them and remove character distortion. Google CAPTCHAs appear to have undergone some type of geometric transformation and this transformation is not constant within an image. It might not be possible to find an exact estimate of the type or model of distortion. Our results showed that our approach was able to straighten most of the characters in a medium distorted Google CAPTCHA. Figure 4.1 shows some Google CAPTCHAs and their corresponding modified images (in binary) resulting from the best inverse bezier transformation.

4.2 Offline attack on Google CAPTCHAs

In order to test the accuracy of our approach on a large database, we launched an offline attack on a database of 1300 Google CAPTCHAs of various lengths downloaded from Googles signup



Figure 4.1: Selected Google CAPTCHAs and their corresponding unmorphed images using inverse bezier transformation

page[12]. In broad terms, our attack contains the following sequential steps:

1. We split a whole CAPTCHA image into two parts by cutting it in the middle.
2. The two parts go through a separate distortion removal (un-morphing) step to get a number of candidate images for recognition.
3. All candidate images are sent to a commercial OCR tool (Abbyy FineReader in our case) for automated recognition.
4. We align the recognition results of all candidate images and vote to get the final result.

The detailed steps are presented in the following subsections.

4.2.1 Divide and Conquer

A Google CAPTCHA may contain up to ten (lowercase) English letters with an average number of approximately 9 letters per CAPTCHA. Our experiments showed that a direct application of any OCR tool on the whole CAPTCHA does not produce any meaningful result even when the tool is augmented by some pre-processing and post-processing steps. Therefore, our attack first tries to reduce the number of letters by splitting a Google CAPTCHA image into two halves and later combines the results of the two halves. The splitting is done by looking for the horizontally thinnest part around the midpoint of the CAPTCHA image after binarization. In other words, we look for the column with the least number of foreground pixels.

4.2.2 Inverse map application

The 500 inverse bezier maps are applied on each half of the CAPTCHA image, and collage files of unmorphed images are created by mosaicing several unmorphed images together separated by sufficient horizontal and vertical space. Each collage contains 50 unmorphed CAPTCHA images.

4.2.3 Character recognition using OCR tool

Each candidate image obtained in the last step is sent to Abbyy FineReader for recognition. Abbyy FineReader is customized to handle only lowercase English letters since this is the requirement of Google CAPTCHAs. We did not try to re-train Abbyy FineReader with any new training samples taken from Google CAPTCHAs however it was observed that Abbyy FineReader is sensitive to noise and hence it may be interesting to re-train Abbyy or develop our own OCR using our own training data set.

We automated this step initially using windows programming by passing each modified CAPTCHA image to the OCR tool one by one but it was too time consuming. Therefore, to make the process faster, we used the automated task feature of Abby FineReader to read a number of files from a location. The recognition results are saved in an text file, from which we chose the best result with the highest number of character matches.

4.2.4 Voting for the final recognition result

After passing the Bezier transformed CAPTCHA images through the OCR tool, we concatenate the recognition results of the two splitted parts of each (possibly incorrectly) recognized CAPTCHA image. Consequently, we obtain n different recognition results, where n is the total number of Bezier maps used for transformation ($n=500$ for our experiments).

We apply majority voting on each character position in the n recognition results to obtain a reliable estimate of each character. We observed that majority voting does not yield good results because incorrect recognitions and character misalignment (e.g., due to extra or fewer recognized characters in the transformed images). We then scanned the unmorphed images bi-directionally before passing them to AbbyFineReader. Boundary voting was then performed to remove bad results at each character index.

We then resorted to a probabilistic voting method in order to improve the results of majority voting. The CAPTCHA dataset was divided into two non-overlapping datasets: a training dataset and a test dataset. From the training dataset, we computed conditional probabilities that an original/input character, O , is recognized as R by the OCR tool. More specifically, we compute: $Pr\{R_a|O_a\}, Pr\{R_b|O_a\}, \dots, Pr\{R_y|O_z\}, Pr\{R_z|O_z\}$. We can also find the marginal character probabilities, $Pr\{O_a\}$,

$Pr\{O_b\}, \dots, Pr\{O_z\}$, from the training dataset by calculating normalized frequencies of each letter in test dataset.

Now in the test dataset, assuming that we recognized the letter ‘a’ (R_a), we computed the inverse conditional probability, $Pr\{O_a|R_a\}$, using the Bayes rule as:

$$Pr\{O_a|R_a\} = \frac{Pr\{R_a|O_a\} \times Pr\{O_a\}}{\sum_{i=a}^z Pr\{R_a|O_i\} \times Pr\{O_i\}} \quad (4.1)$$

Similarly, $Pr\{O_b|R_a\}, \dots, Pr\{O_z|R_a\}$ were computed and the highest probability character was picked as estimate of the original character.

4.2.5 Attack Results

A direct comparison of the recognition results with the actual CAPTCHA strings gave us over 55% correct letter recognition accuracy on a database of 1300 CAPTCHAs. 8 out of these were fully recognized giving a full CAPTCHA recognition accuracy of 0.006%. Using majority voting scheme, we achieved a correct letter recognition of 20%. For probabilistic voting scheme, the dataset was split into training and testing set of 1000 and 300 CAPTCHAs respectively. Inverse conditional probabilities for each letter were calculated from the majority voting results for the training set. The inverse conditional probabilities provided good estimates of four most frequent letters appearing in Google CAPTCHAs i.e., ‘a’, ‘e’, ‘i’ and ‘s’. Using these probabilities, the original characters were estimated from the voting results of the testing set. The above voting scheme achieved a mean accuracy of only 13% correct letter recognition at the correct location. We believe that this percentage can be increased by using a large training set.

Chapter 5

Strength Estimation Measures

This section discusses our work on establishing geometric measures to estimate the strength of text CAPTCHAs. We have attempted to automate the process of evaluating the visual hardness of text CAPTCHAs as judged by an average user with normal eyesight, which is an indirect metric of security since if a CAPTCHA is very hard for human users then it is likely even harder for automated programs. We base our automated evaluation on a number of geometric indicators that can be measured via `regionprops` function and simple binary image processing techniques in MATLAB [33].

5.1 Geometric Indicators

We use the following geometric indicators to capture different aspects of hard CAPTCHAs:

1. Shape Compactness Euler's Number
2. Thickness/Boldness
3. Compactness-Length Ratio
4. Euler's Number-Thickness Ratio



Figure 5.1: Compactness (C_n) values of an easy and hard Google CAPTCHA respectively

5.1.1 Shape Compactness

Packing of characters closely together is being practiced by the designers to enhance the security of contemporary CAPTCHAs. Excessive crowding causes confusion in identifying the characters clearly therefore, reducing usability. We measured the level of compactness (C_n) of a CAPTCHA text as:

$$C_n = \frac{Perimeter^2}{Area} \quad (5.1)$$

The perimeter and area were calculated on the binarized CAPTCHA image. Hard CAPTCHAs have a high compactness value as compared to easy CAPTCHAs. Figure 5.1 shows the compactness values of an easy and hard Google CAPTCHA respectively.

5.1.2 Euler's Number

Increased compactness in a CAPTCHA can create overlaps between adjacent characters resulting in larger fused areas and new holes between them. This results in a large Eulers number (EN) which can be used to differentiate hard CAPTCHAs from easy ones. Euler number can be calculated as:

$$EN = Connected\ Components - Holes \quad (5.2)$$

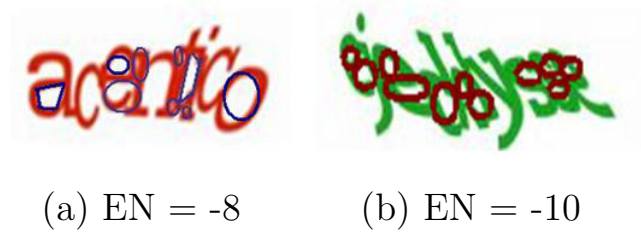


Figure 5.2: Euler Number (EN) values of an easy and hard Google CAPTCHA respectively



Figure 5.3: Number of Erosion Steps (ES) of two Google CAPTCHAs

Figure 5.2 shows the holes and Euler number (EN) values of the CAPTCHAs in Figure 5.1.

5.1.3 Thickness/Boldness

Thickness/boldness of the characters is another aspect of text CAPTCHAs. Thin and closely packed CAPTCHAs with a large number of holes are considered as hard CAPTCHAs. We calculate the number of steps for morphologically eroding all characters in the binarized CAPTCHA image as a measure of the thickness/boldness using square-shaped structuring element of size 2x2 pixels. This measure is called the number of Erosion Steps (ES). Figure 5.3 shows the ES values of two Google CAPTCHAs.

5.1.4 Compactness-Length Ratio

The Compactness-Length (CL) is the ratio between shape compactness (Cn) and the width (Cw) of the text in the CAPTCHA. We observed that highly compact CAPTCHAs have small width. Therefore, using CAPTCHA width along with compactness can be a better indicator of hardness than the compactness alone. Higher CL values correspond to harder CAPTCHAs and vice versa.

5.1.5 Euler's Number-Thickness Ratio

Euler's Number-Thickness (ET) is the ratio between Eulers number (EN) and the number of Erosion Steps (ES) for the CAPTCHA. We observed that smaller ET values correspond to harder CAPTCHAs and vice versa.

The last two indicators are used in our experiments because a combination of them allows us to distinguish easy and hard CAPTCHAs with an acceptable accuracy. The geometric indicators of selected CAPTCHAs are shown in Figure 5.4. It can be inferred from the figure that higher CL and lower ET values are obtained for CAPTCHAs that are more compressed.

5.2 Experimental Results

To test the strength estimation accuracy of our geometric indicators, user studies were conducted to get the subjective scores on non-overlapping training and testing CAPTCHA sets. The subjectives scores for the training set were used to train a Neural Network (NN) Classifier which was then tested on the test CAPTCHA set.



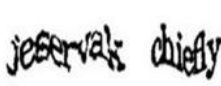
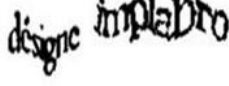




Google CAPTCHA		Google reCAPTCHA	
			
Cn=790, EN=-10, ES=6, Cw=130 CL=6.02, ET=-1.66	Cn=1436, EN=-18, ES=6, Cw=150 CL=9.5, ET=-3	Cn=1728, EN=-13, ES=5, Cw=286 CL=6.03, ET=-2.6	Cn=1557, EN=-11, ES=6, Cw=259 CL=6, ET=-1.83
Microsoft CAPTCHA with Two Rows		Yahoo! CAPTCHA	
			
Cn=1513, EN=-5, ES=10, Cw=188 CL=8.04, ET=-0.5	Cn=1479, EN=-5, ES=14, Cw=200 CL=7.39, ET=-0.35	Cn=1195, EN=-10, ES=8, Cw=140 CL=8.53, ET=-1.25	Cn=1005, EN=-20, ES=7, Cw=143 CL=7.00, ET=-2.857

Figure 5.4: Objective hardness indicators of selected CAPTCHAs of four different CAPTCHA schemes

5.2.1 User Studies

The first user study comprised of 20 users (engineering students and faculty members) who were asked to rate the training dataset on a 5-point scale so that we can easily define the boundary between easy and hard CAPTCHAs:

- 1 = extremely easy
- 2 = somewhat easy
- 3 = somewhat difficult
- 4 = difficult but readable
- 5 = impossible to read

For each CAPTCHA, we used the median score as the average users rating. The CAPTCHA schemes involved in this work include Google CAPTCHA, Google reCAPTCHA, a Microsoft

CAPTCHA scheme with two rows (one of several CAPTCHA schemes used by Microsoft) and Yahoo! CAPTCHA schemes. The second user study comprised of 5 new users who rated the testing dataset of 38 new CAPTCHAs. These subjective scores were used to compare our NN classification results.

5.2.2 Neural Network Classification

The goal of quantitative strength estimation is to predict the hardness given the scores of geometric indicators for a CAPTCHA. We consider this as a binary classification problem: given a feature vector formed by the geometric indicators, a CAPTCHA is classified as easy or hard. A 2-tuple feature vector [CL,ET] was formed and a binary Neural Network Classifier was used for this purpose.

To train the classifier, the median subjective hardness scores from the first user study were used. Figure 5.5 shows the scatter plot of the average users ratings of the 50 CAPTCHAs on the CL-ET plane. One can see that the upper left corner of the plane contains mainly easy CAPTCHAs (green and yellow markers), which implies that the 2-tuple (CL,ET) can be used to get a classifier with acceptable classification accuracy. A 5-fold cross-validation scheme is used for training to avoid any bias due to the random selection of the training and validation sets. We tested the classifier on a testing set with 38 new CAPTCHAs and five new users. We trained the NN approximately 30 times with different random partitions of the training set to test the stability of classification results. Average classification accuracy of the 5-fold cross-validation process is larger than 80% in all cases except one (76.8%) and with a high probability exceeds 85% [33].

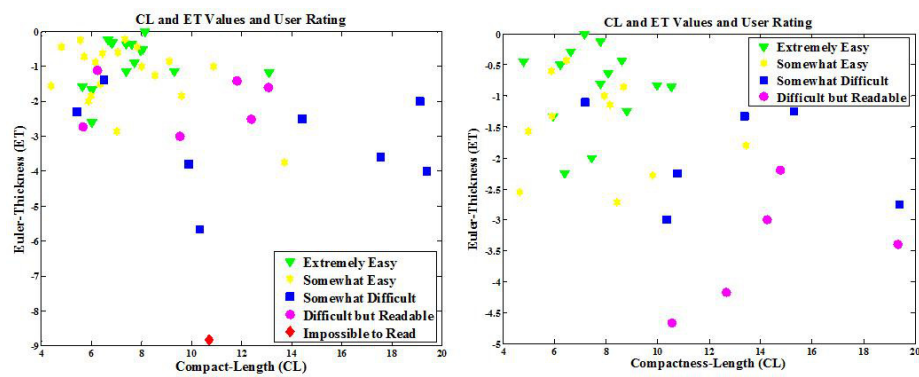


Figure 5.5: User study for training the classifier: training data (left) and testing data (right)

Chapter 6

Conclusion and Future Work

6.1 Conclusions and Summary

In this thesis, we performed a semi-automated evaluation of visual text CAPTCHAs. For this purpose, we 1) Built an algorithm to defeat Google CAPTCHAs through reverse engineering 2) Developed geometric strength indicators to estimate the visual hardness of text CAPTCHAs.

Our experiments on breaking Google CAPTCHAs show that we are able to accurately recognize the most frequent letters i.e., ‘a’, ‘e’, ‘i’ and ‘s’ on a database of 1300 CAPTCHAs. With known CAPTCHA text, we are able to achieve full recognition rate of 0.006% and mean correct letter recognition rate of over 50%. Though, the correct letter recognition accuracy of our current voting schemes lags behind this rate by 30% , we believe that even 20% correct letter recognition accuracy has not been achieved on contemporary text CAPTCHAs especially Google so far. In the next section, we discuss how we intend to improve this accuracy in near future.

In order to automate the visual hardness of text CAPTCHAs, we have formulated five geometric indicators. The Neural Network based classification results on a dataset of 38 CAPTCHAs from 4 schemes show that two of these proposed geometric indi-

cators namely Compact-Length ratio and Euler-Thickness ratio are able to accurately estimate a text CAPTCHA as easy or hard with 80% accuracy.

6.2 Future Work

In near future, we wish to explore the following tasks to improve the correct character recognition accuracy of our voting schemes:

- Employment of more enhanced distortion removal (e.g., de-waving and global obliqueness correction).
- Addition of user training sets to OCR tools to improve character recognition.
- Combination of several OCR tools for recognizing unmorphed images and voting their results.
- Improving estimate of recognition probability during voting by using lines detected by Hough transform to eliminate false matches.

Regarding our strength measures, our next direction is as follows:

- Collection of more subjective user data i.e., user studies on a larger and diverse CAPTCHA dataset with more users.
- Run new tests on different CAPTCHA schemes to see if more features are needed.

Table 6.1: Maps for Control points

Maps	μ_x	μ_y
1	$\begin{bmatrix} a_x & b_x + (\frac{c_x - b_x}{3}) & c_x - (\frac{c_x - b_x}{3}) & d_x \\ a_x & b_x & c_x & d_x \\ a_x & b_x & c_x & d_x \\ a_x & b_x & c_x & d_x \end{bmatrix}$	$\begin{bmatrix} b_y - 3 & b_y - 3 & b_y - 3 & b_y - 3 \\ b_y & b_y & b_y & b_y \\ c_y & c_y & c_y & c_y \\ d_y & d_y & d_y & d_y \end{bmatrix}$
2	$\begin{bmatrix} a_x & b_x + (\frac{c_x - b_x}{3}) & c_x - (\frac{c_x - b_x}{3}) & d_x \\ a_x & b_x & c_x & d_x \\ a_x & b_x & c_x & d_x \\ a_x & b_x & c_x & d_x \end{bmatrix}$	$\begin{bmatrix} b_y - 3 & b_y - 3 & b_y - 3 & b_y - 3 \\ b_y & b_y & b_y & b_y \\ c_y & c_y & c_y & c_y \\ c_y + 3 & c_y + 3 & c_y + 3 & c_y + 3 \end{bmatrix}$
3	$\begin{bmatrix} b_x & b_x + 10 & c_x & d_x \\ b_x & b_x + 10 & c_x & d_x \\ b_x & b_x + 10 & c_x & d_x \\ b_x & b_x + 10 & c_x & d_x \end{bmatrix}$	$\begin{bmatrix} a_y & a_y & a_y & a_y \\ b_y & b_y & b_y & b_y \\ c_y & c_y & c_y & c_y \\ d_y & d_y & d_y & d_y \end{bmatrix}$
4	$\begin{bmatrix} b_x & b_x + 10 & c_x & d_x \\ b_x & b_x + 10 & c_x & d_x \\ b_x & b_x + 10 & c_x & d_x \\ b_x & b_x + 10 & c_x & d_x \end{bmatrix}$	$\begin{bmatrix} a_y & a_y & a_y & a_y \\ b_y & b_y & b_y & b_y \\ c_y & c_y & c_y & c_y - 3 \\ d_y - 3 & d_y & d_y & d_y - 3 \end{bmatrix}$
5	$\begin{bmatrix} b_x & b_x + 10 & c_x & d_x \\ b_x & b_x + 10 & c_x & d_x \\ b_x & b_x + 10 & c_x - 5 & d_x \\ b_x & b_x + 10 & c_x - 5 & d_x \end{bmatrix}$	$\begin{bmatrix} a_y & a_y & a_y & a_y \\ b_y & b_y & b_y & b_y \\ c_y & c_y & c_y & c_y \\ d_y & d_y & d_y & d_y \end{bmatrix}$
6	$\begin{bmatrix} a_x + 10 & b_x + 8 & c_x & c_x \\ a_x + 10 & b_x + 8 & c_x & c_x \\ a_x & b_x & c_x - 5 & c_x \\ a_x & b_x & c_x - 5 & c_x \end{bmatrix}$	$\begin{bmatrix} a_y & a_y & a_y + 7 & a_y + 10 \\ b_y & b_y & b_y + 9 & b_y + 5 \\ c_y & c_y & c_y & c_y \\ d_y & d_y & d_y & d_y \end{bmatrix}$
7	$\begin{bmatrix} b_x & b_x + (\frac{c_x - b_x}{3}) & c_x - (\frac{c_x - b_x}{3}) & c_x \\ a_x + 10 & b_x + 8 & c_x & c_x \\ a_x & b_x & c_x - 5 & c_x \\ a_x & b_x & c_x - 5 & c_x \end{bmatrix}$	$\begin{bmatrix} b_y - 3 & b_y - 3 & b_y - 3 & b_y - 3 \\ b_y & b_y & b_y & b_y \\ c_y & c_y & c_y & c_y \\ d_y & d_y & d_y & d_y \end{bmatrix}$
8	$\begin{bmatrix} a_x & b_x & c_x & d_x \\ a_x + 10 & b_x + 8 & c_x & c_x \\ a_x & b_x & c_x - 5 & c_x \\ b_x & b_x + (\frac{c_x - b_x}{3}) & c_x - (\frac{c_x - b_x}{3}) & c_x \end{bmatrix}$	$\begin{bmatrix} a_y & a_y & a_y & a_y \\ b_y & b_y & b_y & b_y \\ c_y & c_y & c_y & c_y \\ c_y + 3 & c_y + 3 & c_y + 3 & c_y + 3 \end{bmatrix}$
9	$\begin{bmatrix} b_x & b_x + (\frac{c_x - b_x}{3}) & c_x - (\frac{c_x - b_x}{3}) & c_x \\ a_x & b_x & c_x & d_x \\ a_x & b_x & c_x & d_x \\ b_x & b_x + (\frac{c_x - b_x}{3}) & c_x - (\frac{c_x - b_x}{3}) & c_x \end{bmatrix}$	$\begin{bmatrix} b_y - 3 & b_y - 3 & b_y - 3 & b_y - 3 \\ b_y & b_y & b_y & b_y \\ c_y & c_y & c_y & c_y \\ c_y + 3 & c_y + 3 & c_y + 3 & c_y + 3 \end{bmatrix}$
10	$\begin{bmatrix} a_x & b_x & c_x & d_x \\ a_x & b_x & c_x & d_x \\ a_x & b_x & c_x & d_x \\ a_x & b_x & c_x & c_x \end{bmatrix}$	$\begin{bmatrix} b_y - 3 & b_y - 3 & b_y - 3 & b_y - 3 \\ b_y & b_y & b_y & b_y \\ c_y & c_y & c_y & c_y \\ c_y + 3 & c_y + 3 & c_y + 3 & c_y + 3 \end{bmatrix}$

Bibliography

- [1] L. Von Ahn et al. “Telling Humans and Computers Apart (Automatically) or How Lazy Cryptographers do AI,” *Comm. of the ACM*, 47(2):57-60, 2004.
- [2] G. Mori and J. Malik , “Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA,” *Proceedings of Comp. Vision and Pattern Rec. (CVPR) Conf.*, IEEE Computer Society, vol.1, pages:I-134 - I-141, June 18-20, 2003.
- [3] G. Moy, N. Jones, C. Harkless and R. Potter, “Distortion Estimation Techniques in Solving Visual CAPTCHAs”, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR’04)*, Vol 2, June 2004, pp. 23-28.
- [4] J. Yan and A. S. El Ahmad, “Breaking Visual CAPTCHAs with Nave Pattern Recognition Algorithms”, *Proceedings of the 23rd Annual Computer Security Applications Conference (ACSAC’07)*. FL, USA, Dec 2007. IEEE computer society. pp 279291.
- [5] K. Chellapilla ,K. Larson, P. Simard and M. Czerwinski , “Computers beat Humans at Single Character Recognition in Reading based Human Interaction Proofs (HIPs),” *Second Conference on Email and Anti-Spam (CEAS2005)*, July 21-22, Stanford University.

- [6] J. Yan and A. S. El Ahmad, "A Low Cost Attack on Microsoft CAPTCHA", ACM Conf. on Computer and Communications Security (CCS), 2008, October 27-31, Alexandria, VA, USA.
- [7] S. Li, S. A. H. Shah, M. A. U. Khan, S. A. Khayam, A.R.Sadeghi, and R. Schmitz, "Breaking e-Banking CAPTCHAs," In Proceedings of 26th Annual Computer Security Applications Conference (Austin, Texas, USA, December 5-10, 2010). ACSAC2010. ACM, 171-180.
- [8] M. Blum et al: The CAPTCHA Project, "Completely Automatic Public Turing Test to tell Computers and Humans Apart," Department of Computer Science, Carnegie-Mellon University, November 2000, <http://www.captcha.net>.
- [9] M. Chew and H. S. Baird, "BaffleText: a Human Interactive Proof," Proceedings of the 10th SPIE/IST Document Recognition and Retrieval Conference (DRR2003), Santa Clara, CA, 2003, 305-316.
- [10] Microsoft Hotmail, <https://accountservices.passport.net/reg.srf?id=2&sl=1&lc=1033>.
- [11] L. von Ahn, B. Maurer, C. McMillen, D. Abraham and M. Blum, "reCAPTCHA: Human-Based Character Recognition via Web Security Measures," Science, September 12, 2008, pp 1465-1468.
- [12] Google Gmail, <https://www.google.com/accounts/DisplayUnlockCaptcha>.
- [13] The CAPTCHA Project, <http://www.captcha.net/captchas/bongo/>.
- [14] Pix CAPTCHA, <http://gs264.sp.cs.cmu.edu/cgi-bin/esp-pix>.

- [15] J. Elson, J. Douceur and J. Howell, "ASIRRA: A CAPTCHA That Exploits Interest-Aligned Manual Image Categorization," Proceedings of the 15th ACM conference on Computer and communications Security ,October 29-November 2, 2007, Alexandria, VA, U.S.A.
- [16] Y. Rui, "ARTiFACIAL: Automated Reverse Turing Test using FACIAL features," Multimedia Systems, 9(6):493-502, 2004.
- [17] R. Gossweiler, M. Kamvar and S. Baluja, "Whats Up CAPTCHA? A CAPTCHA Based on Image Orientation," In Proc. WWW 2009, ACM Press (2009), 841850.
- [18] R. Datta , J. Li and J. Z. Wang, "IMAGINATION: A Robust Image-Based CAPTCHA Generation System," Proceedings of the 13th annual ACM international conference on Multimedia, November 06-11, 2005, Hilton, Singapore.
- [19] B. B. Zhu, J. Yan, Q. Li, C. Yang, J. Liu, N. Xu, M. Yi and K. Cai, "Attacks and Design of Image Recognition CAPTCHAs," ACM Conference on Computer and Communication Security, Oct 4-8, 2010, Illinois, USA.
- [20] Confident CAPTCHA, <http://www.confidenttechnologies.com/products/confident-captcha>.
- [21] Yuniti CAPTCHA, <http://www.yuniti.com/register.php>.
- [22] K. A. Kluever and R. Zanibbi, "Balancing Usability and Security in a Video CAPTCHA ," Proceedings of the 5th symposium on Usable privacy and security, July 15-17, 2009, Mountain View, California
- [23] NUCAPTCHA, <http://www.nucaptcha.com/>.

- [24] C. Javier, H. Castro and A. Ribagorda, “Pitfalls in CAPTCHA Design and Implementation: The Math CAPTCHA, A Case Study,” Elsevier Computers Security, vol. 29, pp. 141-157, 2010.
- [25] K. Chellapilla, and P. Simard, “Using Machine Learning to Break Visual Human Interaction Proofs (HIPs),” Advances in Neural Information Processing Systems 17, Neural Information Processing Systems (NIPS2004), MIT Press.
- [26] P. Golle, “Machine Learning Attacks Against the ASIRRA CAPTCHA”, Proceedings of the 15th ACM conference on Computer and communications security, October 27-31, 2008, Alexandria, Virginia, USA.
- [27] J. Tam, J. Simsa, S. Hyde and L. Von Ahn, “Breaking Audio CAPTCHAs” , Advances in Neural Information Processing Systems (NIPS), 2009.
- [28] E. Bursztein and S. Bethard, “Decaptcha: Breaking 75% of eBay Audio CAPTCHAs”, In Proceedings of 3rd USENIX Workshop on Offensive Technologies (WOOT2009). USENIX, 2009.
- [29] A. Hindle, M. W. Godfrey, and R. C. Holt, “Reverse engineering CAPTCHAs”, In Proceedings of 15th Working Conference on Reverse Engineering (WCRE’2009).
- [30] J. Yan and A. S. El Ahmad, “Usability of CAPTCHAs or Usability Issues in CAPTCHA Design”, Proceedings of the 4th Symposium on Usable privacy and Security, July 23-25, 2008, Pittsburgh, Pennsylvania.
- [31] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G. M. Voelker, and S. Savage, “Re: CAPTCHAs Understanding CAPTCHA-Solving Services in an Economic Context”, Proceedings of 19th USENIX Security Symposium on

USENIX Security Symposium , August 11-13,2010, Washington,DC.

- [32] Bezier Surface, <http://www.gamedev.net/reference/articles/article1808.asp>.
- [33] M. Nazir,Y. Javed, M. Murtaza Khan, S. Ali Khayam and S. Li, “Poster: Captchcker Automating Usability-Security Evaluation of Textual CAPTCHAs”, Symposium on Usable Privacy and Security (SOUPS), July 2011, Pittsburgh, Pennsylvania.