The Robustness of CAPTCHAs: A Security Engineering Perspective

Jeff Yan and Ahmad Salah El Ahmad

TECHNICAL REPORT SERIES

No. CS-TR-1180 November 2009

TECHNICAL REPORT SERIES

No. CS-TR-1180 November, 2009

The Robustness of CAPTCHAs: A Security Engineering Perspective

J. Yan, A. S. El Ahmad

Abstract

CAPTCHA (or Human Interaction Proof) is now almost a standard security technique for defending against undesirable or malicious bot programs on the Internet. However, the robustness of CAPTCHAs has so far been studied mainly just in communities such as computer vision, and document analysis and recognition. This paper motivates a security engineering perspective of the robustness of CAPTCHAs. Specifically, we show that a number of CAPTCHAs that appeared to be secure, including schemes widely deployed by Microsoft, Yahoo and Google and some other less well-known ones, could be broken with a high success rate with simple but novel attacks. In contrast to earlier work that relied on sophisticated computer vision algorithms, our attacks exploited critical design errors that we discovered in each scheme. The main lesson is that security engineering expertise and experience, in particular adversarial thinking skills, can make a unique and significant contribution to the improvement of the robustness of CAPTCHAs.

Bibliographical details

YAN, J., EL AHMAD, A.S.

The Robustness of CAPTCHAs: A Security Engineering Perspective
[By] J. Yan, A.S. El Ahmad,

Newcastle upon Tyne: University of Newcastle upon Tyne: Computing Science, 2009.

(University of Newcastle upon Tyne, Computing Science, Technical Report Series, No. CS-TR-1180)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE
Computing Science. Technical Report Series. CS-TR-1180

Abstract

CAPTCHA (or Human Interaction Proof) is now almost a standard security technique for defending against undesirable or malicious bot programs on the Internet. However, the robustness of CAPTCHAs has so far been studied mainly just in communities such as computer vision, and document analysis and recognition. This paper motivates a security engineering perspective of the robustness of CAPTCHAs. Specifically, we show that a number of CAPTCHAs that appeared to be secure, including schemes widely deployed by Microsoft, Yahoo and Google and some other less well-known ones, could be broken with a high success rate with simple but novel attacks. In contrast to earlier work that relied on sophisticated computer vision algorithms, our attacks exploited critical design errors that we discovered in each scheme. The main lesson is that security engineering expertise and experience, in particular adversarial thinking skills, can make a unique and significant contribution to the improvement of the robustness of CAPTCHAs.

About the authors

Jeff Yan did his PhD with Ross Anderson in Cambridge, and his main research field is information security. He is interested in most aspects of security, both theoretical and practical. His recent work include systems security (e.g. spam detection, phishing defence, and system and security issues in network games), applied crypto (e.g. cryptanalysis of some traitor tracing schemes) and human aspects of security (e.g. the so-called "usable security"). Research organisations Jeff has been affiliated with: DSO National Labs (Singapore), Hewlett-Packard Labs (Bristol), Chinese University of Hong Kong and Microsoft Research, Asia.

Ahmad is a PhD student, supervised by Dr. Jeff Yan. Ahmad is interested in the many aspects of Security and Usability but in particular the design of Secure and Usable CAPTCHA systems. Ahmad gained his MSc in Computing Science with a distinction from Newcastle University in 2006, and has been award the Philip Merlin prize for the best MSc dissertation for 2006. Ahmad is partially supported by a prestigious Overseas Research Students Award (ORS).

Suggested keywords

CAPTCHA,
ROBUSTNESS,
SEGMENTATION,
INTERNET SECURITY,
SECURITY ENGINEERING

The Robustness of CAPTCHAs: A Security Engineering Perspective

Jeff Yan, Ahmad Salah El Ahmad
School of Computing Science, Newcastle University
Newcastle upon Tyne, NE1 7RU, UK

May 1, 2009

Abstract: *CAPTCHA (or Human Interaction Proof) is now almost a standard security technique for defending against undesirable or malicious bot programs on the Internet. However, the robustness of CAPTCHAs has so far been studied mainly just in communities such as computer vision, and document analysis and recognition. This paper motivates a security engineering perspective of the robustness of CAPTCHAs. Specifically, we show that a number of CAPTCHAs that appeared to be secure, including schemes widely deployed by Microsoft, Yahoo and Google and some other less well-known ones, could be broken with a high success rate with simple but novel attacks. In contrast to earlier work that relied on sophisticated computer vision algorithms, our attacks exploited critical design errors that we discovered in each scheme. The main lesson is that security engineering expertise and experience, in particular adversarial thinking skills, can make a unique and significant contribution to the improvement of the robustness of CAPTCHAs.*

Keywords: CAPTCHA, robustness, segmentation, Internet security, security engineering

1. Introduction

A CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart) is a program that generates and grades tests that are human solvable, but intended to be beyond the capabilities of current computer programs [8]. This technology often makes use of a hard, open AI problem, and is now a common security mechanism for defending against undesirable or malicious Internet bot programs. It has found widespread application on numerous commercial web sites. For example, Google, Microsoft and Yahoo have all deployed their own CAPTCHAs for years to defend against email spam (by making it harder for spammers to harvest free email accounts).

Moni Noar was in 1996 the first person who proposed to use automated Turing tests to verify that a human, rather than a bot, is in the loop [6]. Alta Vista patented a similar idea in 1998 (United States Patent 6195698). However, the term of CAPTCHA was coined in 2000 by a team led by Manuel Blum and Luis von Ahn at Carnegie Mellon University, and the popularity of such technology was largely due to this team's efforts. To date, the most widely used CAPTCHAs are the so-called *text-based schemes*, in which users are asked to recognize a distorted text, which is intended to be beyond the capabilities of the state of the art of pattern recognition programs.

The security of CAPTCHAs concerns a number of aspects. First, the role of a CAPTCHA is effectively the same as the following simple challenge-response protocol.

Service → Client: a CAPTCHA challenge
Client → Service: response

Therefore, protocol-level attacks can be an issue. For example, a spammer could shift the load of solving CATPCHA challenges to porn site visitors, achieving an oracle attack. A spammer could also outsource the task of solving CAPTCHA challenges to people in low-paying countries. On the other hand, systems aspects of security also matter for CAPTCHAs. For example, some early CAPTCHAs could be bypassed simply by re-using the session ID of a known challenge image [12]. However, all these are beyond the scope of this paper, which instead focuses on another aspect of CAPTCHA security, i.e. the *robustness* of CAPTCHAs, which is the strength of their resistance to computer programs that attackers write to automatically solve CAPTCHA tests.

In this paper, we motivate a security engineering approach to the robustness of CAPTCHAs, a subject that has so far been studied mainly just in the computer vision and document analysis and recognition communities, as will be reviewed below. Our discussion will focus on text CAPTCHAs, but some lessons we have learned are also applicable to other types of CAPTCHAs.

Specifically, we will examine a number of recent, representative text CAPTCHAs, including the schemes widely deployed by Microsoft, Yahoo and Google, as well as others that are less known. We show that although these schemes, as deployed in the period of 2006 - 2008, appeared to be secure, they could be broken - in the sense of writing computer programs that automatically solve CAPTCHA tests - with a high success rate using simple but novel attack strategies. In contrast to early work that relied on sophisticated algorithms, our attacks exploited fatal design errors that we discovered in each scheme.

2. Related Work

Specialised computer vision algorithms had some success for breaking some early text CAPTCHAs. For example, Mori and Malik [4] designed sophisticated object recognition algorithms to break the EZ-Gimpy (92% success) and the Gimpy (33% success) schemes, two early CAPTCHAs created by the CMU team. Moy et al [5] developed distortion estimation techniques to break EZ-Gimpy with a success rate of 99% and 4-letter Gimpy-r with a success rate of 78%.

Chellapilla and Simard [1] attacked a number of CAPTCHAs taken from the web using machine-learning algorithms (largely neural networks), achieving a success rate of from 4.89% to 66.2%.

Chellapilla and colleagues [2] argued that if the positions of characters are known in challenge images generated by a CAPTCHA, then breaking this scheme is just a pure recognition problem, which is a trivial task with standard machine learning techniques such as neural networks. However, when the location of characters in a CAPTCHA challenge is not known a-priori, the state of the art (including machine learning) methods do not work well in locating the characters, let alone recognising them. In general, identifying character locations in the right order, or *segmentation*, is still an open problem, and it is computationally expensive, and often combinatorially hard [1].

Therefore, it is suggested that the robustness of text-based schemes should rely on the difficulty of finding where each character is (segmentation), rather than which character it is (recognition) [7, 1, 2]. That is, CAPTCHAs should be *segmentation-resistant*. In other words, if a CAPTCHA can be successfully segmented, then this scheme is effectively broken.

A common method to estimate the strength of a CAPTCHA is as follows. Denote by s the average percentage of challenges that can be entirely segmented correctly, and by r an individual character recognition rate that can be achieved. Then, the overall (segmentation and then recognition) success rate for breaking a scheme can be estimated by $s{*}r^n$, where n is the average text length used in a scheme.

Usability and robustness are two fundamental issues with CAPTCHAs, and they often interact with each other. In [11], we examined usability issues that should be considered and addressed in the design of CAPTCHAs, and discussed subtle implications that some of those issues can have on robustness.

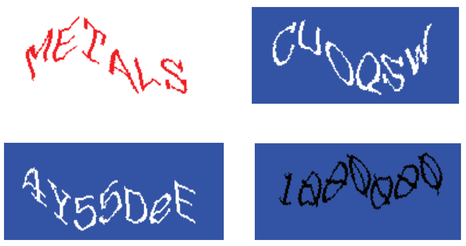
A commonly accepted goal for CAPTCHA design is that automated attacks should not achieve a success rate of higher than 0.01% for passing a CAPTCHA, but that the human success rate should be at least 90% [2].

Some other related studies were surveyed in our recent work [9].

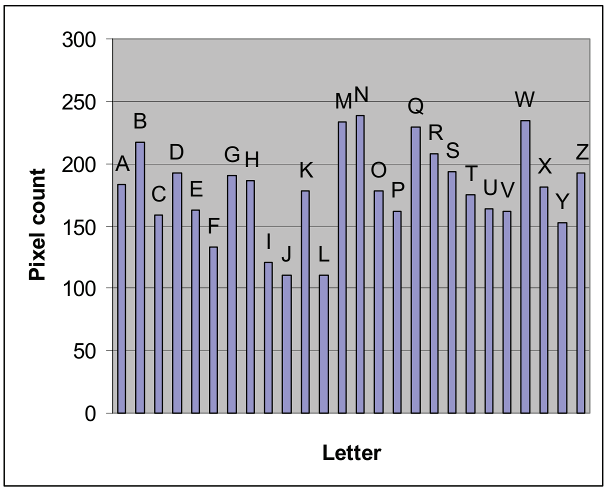
3. *Captchaservice.org* schemes: counting the number of pixels as an attack

Captchaservice.org was a publicly available web service for the sole purpose of generating CAPTCHA challenges. The design of this service and various CAPTCHA schemes it supported were discussed in a recent paper by their designer [3]. We examined four of the CAPTCHA schemes provided by this service. A challenge example is shown in Fig 1(a) for each of the schemes.

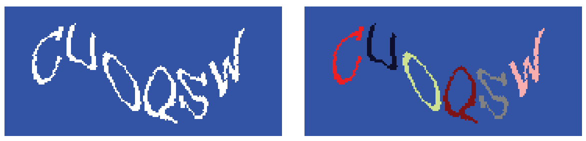
As deployed from 2006 to 2007, all the four schemes based their robustness on a technique of *random-shearing distortion*, which was applied to a challenge image both vertically and horizontally. Specifically, the distortion works as follows: "… the pixels in each column of the image are translated up or down by an amount that varies randomly yet smoothly from one column to the next. Then the same kind of translation is applied to each row of pixels (with a smaller amount of translation on average)." [3]. The main difference between these schemes was the alphabet used, as well as the length of text allowed in each challenge. For example, both the *word_image* and *random_letters_image* schemes used capital letters only and the text length was six. The *number_puzzle_text_image* scheme used only numbers, which could be up to 7 digits. The *user_string_image* scheme was designed to accept any user-supplied string of at most 15 characters that consisted of digital, and capital and lower-case letters.



(a)



(b)



(c)

Fig 1. Four *captchaservice.org* schemes.

(a) An example challenge for each of the schemes (clockwise: *word_image*, *random_letters_image*, *user_string_image*, *number_puzzle_text_image*). These schemes used the same distortion technique, but could differ in alphabet sets and text lengths allowed. Two colours were used in each scheme, with the challenge text being foreground.

(b) Letters A-Z and their pixel counts. ('J' and 'L' had the same pixel count; so were 'K' and 'O', and 'P' and 'V'.)

(c) Color filling segmentation. On the left is an original image, and on the right the segmented image.

We showed in [9] that the *random_shearing* distortion provided all the four schemes a resistance level ranging from reasonable to excellent in terms of being decoded by one of the best commercial OCR products in the market. However, for all the schemes, we could recognise for most of the time all characters embedded in a challenge, by exploiting critical design flaws we identified in each scheme.

A critical vulnerability we identified in all these schemes is that although a character was distorted into a different shape each time, it (almost always) consisted of a constant number of foreground pixels. That is, each character most of the time had a constant pixel count in all challenges generated. Furthermore, most of the characters had a distinct pixel count. For example, Figure 1(b) plots the pixel count of letters A-Z, i.e., the alphabet set used by both the *word_image* and *random_letters_image* schemes.

The second critical vulnerability we identified is that we could use a simple method to identify each character in a challenge image in the right order, that is, to properly segment the image into individual characters. The basic idea is: few characters connected with each other in the target schemes, and therefore we could separate the characters by detecting every connected component in a challenge image.

Our segmentation method works as follows. Only two colours were used in each challenge, with the challenge text being the foreground colour. Therefore, first, we detect a foreground pixel, and then trace all its foreground neighbours until all pixels in this connected component are traversed. Next, the algorithm locates a foreground pixel outside of the area of the detected component(s), and starts another traversal process to identify a next component. This process continues until all connected component in the challenge are located.

This algorithm is effectively like using a distinct colour to flood each character, so we call it a method of "**colour filling segmentation**". Figure 1(c) shows the result of applying the CFS method to a challenge, where the number of colours used to fill the image is the number of characters in the image.

Based on the above two observations, our attack is as follows.

1. Build a character–pixel count lookup table for the alphabet set used in a scheme.
2. Remove small noise dots, if any, in a challenge image - they were easily distinguishable as they had a pixel count much smaller than any legitimate character did.
3. Divide the challenge into multiple segments with the CFS method.
4. Count the number of foreground pixels in each segment;
5. Look up the pixel count table to identify each candidate character. If a pixel count cannot be located in the table, it is very likely that the corresponding segment was just a component of a broken character. We combine this segment with its left and right neighbour segments respectively, and the combination that returns a

meaningful result in the look-up table will be treated as a single character. When both combinations are plausible, we randomly choose one of them.

6. For the characters with identical pixel counts such as ‘J’ and ‘L’, ‘K’ and ‘O’, and ‘P’ and ‘V’, we tell them apart by analyzing their geometric layouts with simple algorithms. (The *word_image* scheme used an English word in each challenge. Therefore, spelling checking could often tell apart the characters with identical pixel counts).

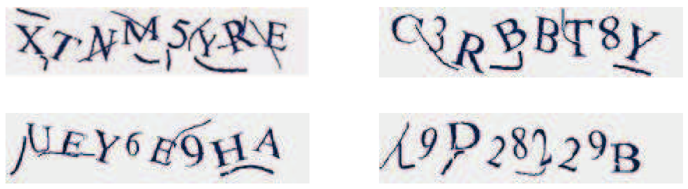
This simple attack has achieved almost 100% success for quickly breaking each of the target schemes. For example, a success rate of 98% was achieved for breaking the *random_letters_image* scheme, and it took only ~16 ms per challenge on an ordinary desktop computer with Pentium 2.8 GHz CPU and 512 MB memory.

4 Microsoft CAPTCHA: Arcs and characters were distinguishable

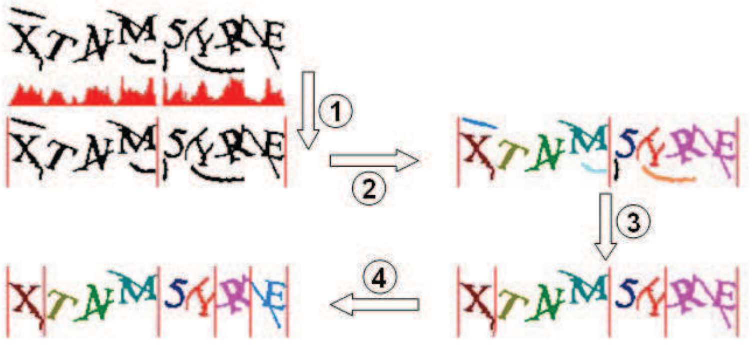
Microsoft first deployed their CAPTCHA in Hotmail’s user registration system in 2002 [7]. Ever since, this scheme has undergone extensive improvement in terms of both robustness and usability [2]. It has been deployed in many of Microsoft’s online services including Hotmail, MSN and Windows Live for years.

Designed to be segmentation resistant, this scheme was a collaborative effort of an interdisciplinary team of diverse expertise in Microsoft including document processing and understanding, machine learning, HCI and security specialists. In fact, the widely accepted “segmentation resistance” principle and the commonly accepted robustness criteria were established by this team.

Fig 2(a) shows some example challenges generated by this Microsoft CAPTCHA.



(a)



(b)

Fig 2. Microsoft CAPTCHA. (a) Four example challenges.

(b) A successful segmentation attack. Step ①: vertical segmentation, which divides a pre-processed image into chunks with the help of a histogram. Step ②: color filling segmentation, which identifies separate objects in each chunk. Step ③: arc removal, where relative position checking plays a major role in removing arcs. After those arcs are removed, the histogram of the image is updated and the image is segmented more chunks. Step④: Identifying and then segmenting remaining connected objects. In the final result, eight valid characters are successfully identified in the right order with each displayed in a different color (and most arcs are deleted).

The main anti-segmentation measure introduced was the addition of random arcs of different thicknesses, such as:

- Non-intersecting thick arcs: these are the same color as texts in a challenge, and their thickness can be the same as the thick portions of characters. They do not directly intersect with any characters to avoid decreasing the usability of the scheme.
- Intersecting thin arcs: these are the same color as texts in a challenge; their thickness is typically not as large as the above type of arcs, but can be the same as the thin portions of characters. They intersect with thick arcs, characters or both.

The rationale behind this design was: these arcs are themselves good candidates for false characters, and therefore the mix of random arcs and characters would confuse state of the art segmentation methods, providing strong segmentation resistance [2].

A key issue for a segmentation attack on this CAPTCHA is to tell apart arcs and valid characters, which we achieved with a simple attack that exploited critical vulnerabilities in the scheme as follows.

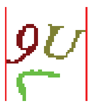
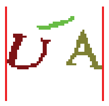
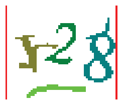
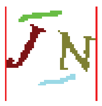
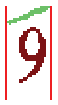
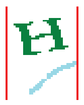
First, after some simple pre-processing including binarization, which converts a color challenge image to a black-white one, a standard vertical segmentation method was applied to segment the challenge vertically into several *chunks*, each of which might contain one or more characters. The process of vertical segmentation starts by mapping the image to a histogram that represents the number of foreground pixels per column in the image. Then, vertical segmentation lines separate the image into chunks by cutting through columns that have no foreground pixels at all. Step ① in Fig 2(b) illustrates this process, where a challenge is divided into two chunks.

Then, shown as Step ② in Fig 2(b), CFS was applied to identify all the connected components in each chunk, which we call objects and can be an arc, character, connected arcs, or connected characters.

We observed that the relative positions of objects in a chunk could tell arcs and real characters apart with a high success rate. For example, typically characters were closer to the baseline (i.e. the horizontal central of a chunk) whereas arcs were closer to the top or bottom image borders. In addition, characters are horizontally juxtaposed, but never vertically. Based on these observations, we identified typical relative position patterns that could pinpoint which object was an arc in a chunk. An incomplete list of the patterns we identified is illustrated with real examples in Table 1.

This method of examining the relative position of objects, as shown as Step ③ in Fig 2(b), identified and removed most arcs in the challenge.

Table 1. Typical relative position patterns

Relative position patterns			Decision
Layout	Description	Example	
O1 O2 O3	*Three objects in a chunk*: two objects more or less align along the baseline, the 3rd object under either of them		O3 is arc and can be removed
O3 O1 O2	*Three objects in a chunk*: two objects more or less align along the baseline, the 3rd object on top of either of them		O3 is arc and can be removed
O1 O2 O3 O4	*Four objects in a chunk*: Three objects more or less align along the baseline, the 4th object under any of them		O4 is arc and can be removed
O1 O2 O3 O4	*Four objects in a chunk*: Two objects more or less align along the baseline, the 3rd and 4th objects under and on top any of them respectively		O1 and O4 are arcs and can be removed
O1 O2	*Two objects in a chunk*: vertically juxtaposed	 or 	The object that is less aligned with the baseline is removed as an arc.

Other vulnerabilities in this CAPTCHA made it possible for us to guess with a high success rate which object contained connected characters and how many such characters there were, and to segment the connected characters properly. The details can be found in [10], but step ④ in Fig 2(b) gives an example of the final segmentation results.

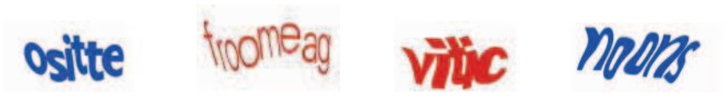
Overall, our segmentation attack achieved a success rate of higher than 90% on this Microsoft CAPTCHA (as deployed in the summer of 2007)[1], and we estimated that this scheme could be broken with an overall (segmentation and then recognition) success rate of over 60% (≈ .9 * .95^8; the individual character recognition rate was about 95% and the text length in this scheme was always 8 [10]). We also found that this attack could be effectively extended to achieve a success rate of about 77% for segmenting a Yahoo CAPTCHA (that had been deployed until March, 2008), leading to an overall success rate of about 60% for breaking this scheme (see [10] for the details).

[1] The work was done in the summer of 2007. We notified Microsoft the weakness of their CAPTCHA in Sept, 2007. Responding to their request, we held this attack confidential until April 10, 2008. To the best of our knowledge this is the first effective segmentation attack on their scheme.

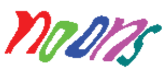
5 Google CAPTCHA: vulnerable to color filling segmentation

Google have also been deploying a CAPTCHA to protect their online services. Fig 3(a) shows some example challenges generated by this scheme. The segmentation resistance mechanism used in this CAPTCHA is the so-called "crowding characters together" method, i.e. letting characters touch or overlap with each other.

However, we correctly segmented 12 out of 100 random samples that we collected between December 2007 and February 2008, using the color filling segmentation alone. Fig 3(b) shows an example that was vulnerable to such an attack. Since the average text length was 6.25, this could lead to an overall success rate of 8.7% (≈ .12 * .95^6.25) for breaking this scheme.



(a)



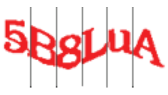
(b)

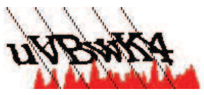
Fig 3. Google CAPTCHA.
(a) Example challenges. (b) A challenge that was vulnerable to the CFS attack.

6 Yahoo CAPTCHA: the number of characters and the text length was correlated

Yahoo started to adopt CAPTCHA technology in 2000, being one of the earliest major organisations to do so. Since then, Yahoo have upgraded their CAPTCHAs a number of times. In March 2008, Yahoo rolled out a new version, in which additional effort was put into making it segmentation resistant. As shown in Fig 4(a), challenge texts in this version were compacted, and characters usually connected - they either touch with each other, or were connected by intersecting random lines. However, just one week after its deployment, we discovered some critical flaws in this CAPTCHA that could be exploited for a successful attack.

(a)





(b) (c)

Fig 4. Yahoo's March 2008 scheme (a) original challenges;
(b) an example of regular segmentation; (c) an example of angular segmentation

The key vulnerability is the following. In this Yahoo CAPTCHA, the length of text embedded in a challenge varied often, and was apparently not predictable. This is a good design feature, since it is harder or even impossible for an automated attack to segment a challenge if the number of characters in the challenge is unknown. However, we discovered that the number of characters in a challenge could be estimated with a high success rate (74% for a sample set, and 68.5% for the test set we used) by measuring the width of the text in the challenge.

We observed that two main types of challenges were generated by this CAPTCHA, and they could be differentiated by a simple program. As shown in Fig 6, one type employed a transformation that shifts character pixels by an angle while maintaining the shapes of the characters. In spirit, this transformation is similar to change characters from a regular shape to the italic form, but to an opposite direction. We call this type *angular* challenge. The other type did not undergo such a transformation, and we call them *regular* challenges.

We designed the following two simple segmentation algorithms to deal with each type of challenges, respectively.

Segmentation of regular challenges. After pre-processing steps such as binarization, color filling segmentation and arc removal, the number of characters in a challenge, denoted by n, is directly estimated using the width of text. If there is only a single connected component, i.e. an object in the challenge, the object will be evenly and vertically cut into n chunks, each being a segment. If there are two or more objects, the relative size of these objects will be used to estimate the number of characters in each object, denoted by n_i. For example, if a challenge is estimated to contain 5 characters and there are two objects in the challenge, then our algorithm will determine that the object with a larger width contains 3 characters, and the other 2 characters. Next, object i is evenly and vertically divided into n_i chunks, each being a segment.

Fig 4(b) shows an example, where it was correctly estimated that there were six characters in the challenge, and our algorithm simply divided the challenge text vertically into six even segments, each turning out to exactly contain a single character.

Segmentation of angular challenges. After pre-processing, we first project a challenge at an angle to the vertical of 33.5 degrees - we observed that the angle used for the angular transformation was almost always the same - to create a histogram that represents the number of foreground pixels per projecting line in the image. Then, we use the span (i.e. length) of the histogram to estimate n, the number of characters in the challenge. (Here, we did not use the width of the challenge text measured from its left-most foreground pixel to the right-most one, since it would be less accurate.) Next, we divide the histogram into n even chunks, which give us $n+1$ boundary points in the X-axis. Starting with each of the points, we draw a line at an angle of 56.5 degrees to the horizontal line to cut the challenge image into n segments, each being supposed to contain a single character. Fig 4(c) gives an example showing that the angular segmentation method correctly estimated the number of characters in the sample and successfully segmented them.

Using the above two segmentation algorithms with associated rules to identify which algorithm to use, we achieved a segmentation success rate of around 33.4% on this Yahoo scheme. As a result, we estimate that this scheme can be broken with an overall success rate of 25.9% (≈ .334*.95^5; the average text length in this scheme was 5).

7 Concluding Remarks

We have demonstrated that a number of recent CAPTCHAs (including some high-profile ones) that were deployed from 2006 to 2008 could be broken with a success rate *much higher* than the widely accepted design goal for their strength. As a consequence of our work, *captchaservice.org* ceased to offer their service, and Microsoft, Yahoo and Google all changed their CAPTCHA design.

Unlike early work on the robustness of CAPTCHAs, our attacks did not rely on sophisticated, specialised algorithms. Instead, we rely on our training in security engineering to identify what could go wrong with the schemes, and then design simple but novel methods to exploit design flaws that we discovered in each scheme.

Compared to the early CAPTCHAs deployed from 2000 to 2004, the schemes we examined in this paper were better designed. The collective understanding of the design of CAPTCHAs has also increased with the passage of time. Overall, the robustness of CAPTCHAs deployed in the field does not seem to be fundamentally improved, however.

It is in particular alarming that some attacks we developed for simplistic CAPTCHAs were widely applicable to CAPTCHAs that were more carefully designed by major companies. For example, the CFS method we used to segment multiple *captchaservice.org* schemes contributed much to our attack on CAPTCHAs designed by Microsoft, Yahoo and Google. The pixel count attack we discovered while studying the *captchaservice.org* schemes not only achieved a surprising success of recognising individual letters in many schemes, but also aided our attacks on Microsoft's and Yahoo's CAPTCHAs [10] by determining whether a particular component of a challenge image was a valid character or a random noise.

Plausible reasons for these failures include the following. First, the robustness of CAPTCHA was mainly studied as problems of computer vision, document recognition and machine learning, due to their apparent relevance. As demonstrated in this paper, security engineering expertise and experience, in particular adversarial thinking skills, can make a unique and significant contribution to the understanding and improvement of CAPTCHA robustness. Unfortunately, sufficiently adequate such expertise was not in place when Microsoft, Yahoo, Google and others were designing their schemes.

Second, robustness is just one side of the coin of CAPTCHA design, and usability is the other side – by definition, a CAPTCHA unusable for humans have no reason to exist.

Both sides often have subtle implications on each other, and it is not easy to strike the right balance. Therefore, designing CAPTCHAs that exhibit both good robustness and usability is much harder that it might appear to be. However, the current collective understanding of this topic is still limited.

One lesson is as follows. Home brew CAPTCHAs seem to be a bad idea, just like home brew cryptography and security systems. The devil is in the details. It takes a lot of experience and skills to get the design of a CAPTCHA right, and even experienced designers make mistakes. Therefore, it is best to use a CAPTCHA that was carefully designed by highly experienced people and that was publicly and independently vetted before deployment.

To summarize, CAPTCHA design is an interdisciplinary topic where expertise from multiple domains including computer vision, HCI, document recognition and processing can all matter. Our experience suggests that CAPTCHA will go through the same process of evolutionary development as cryptography, digital watermarking and the like, with an iterative process in which successful attacks lead to the development of more robust systems. In this process, security engineers will play a major role.

References

1. 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 (NIPS), MIT Press, 2004.
2. K Chellapilla, K Larson, P Simard and M Czerwinski, "Building Segmentation Based Human-friendly Human Interaction Proofs", 2nd Int'l Workshop on Human Interaction Proofs, Springer-Verlag LNCS 3517, 2005.
3. T Converse, "CAPTCHA generation as a web service", 2nd Int'l Workshop on Human Interactive Proofs, Springer-Verlag. LNCS 3517, 2005. pp. 82-96
4. G Mori and J Malik. "Recognising Objects in Adversarial Clutter: Breaking a Visual CAPTCHA", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03), Vol 1, June 2003, pp.134-141.
5. 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
6. Moni Naor. "Verification of a human in the loop, or Identification via the Turing Test". 1996, online manuscript, available at http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.ps.
7. P Simard, R Szeliski, J Benaloh, J Couvreur and I Calinov, "Using Character Recognition and Segmentation to Tell Computers from Humans", Int'l Conference on Document Analysis and Recogntion (ICDAR), 2003.
8. L von Ahn, M Blum and J Langford. "Telling Humans and Computer Apart Automatically", CACM, V47, No2, 2004.

9. J Yan and A S El Ahmad. “Breaking Visual CAPTCHAs with Naïve Pattern Recognition Algorithms”, in Proc. of the 23rd Annual Computer Security Applications Conference (ACSAC’07). FL, USA, Dec 2007. IEEE computer society. pp 279-291.
10. J Yan and A S El Ahmad. “A Low-cost Attack on a Microsoft CAPTCHA”, 15th ACM Conference on Computer and Communications Security (CCS’08). Virginia, USA, Oct 27-31, 2008. ACM Press. pp 543-554.
11. J Yan and AS El Ahmad. “Usability of CAPTCHAs - Or ‘usability issues in CAPTCHA design’”. The 4th Symposium on Usable Privacy and Security (SOUPS), Carnegie Mellon University, July 23-25, 2008. pp 44-52.
12. H Yeend. Breaking CAPTCHAs without using OCR. 2005. http://www.puremango.co.uk/cm_breaking_captcha_115.php.