Usability and Security of Text-based CAPTCHAs

Cody Sutherland
University of Minnesota, Morris
suthe112@umn.edu

ABSTRACT

Several free internet resources, such as email, blogs, polls, and many others, are targeted for exploitation by automated processes. In order to differentiate between these automated processes and human users, it has become standard security practice to implement CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart). This paper discusses various properties of text-based CAPTCHAs and their impact on security and usability. It also outlines an automated attack on a high-profile CAPTCHA and examines one of the most successful CAPTCHA-related projects to date.

Categories and Subject Descriptors

H.1.2 [**Information Systems**]: Models and Principles—*User/Machine Systems*; K.6.5 [**Computing Milieux**]: Management of Computing and Information Systems—*Security and Protection*

General Terms

Security, Human Factors, Verification

Keywords

CAPTCHA, usability, segmentation, interent security

1. INTRODUCTION

In the most general sense, CAPTCHAs are problems that a human can easily complete, yet current computer programs cannot. Thus they are used to determine whether a given request is coming from a human user or an automated process. Throughout their history and even today, the most widely used CAPTCHAs are text based [8]. However, it is important to note that a CAPTCHA may use audio, video, images, or a combination of various media. CAPTCHAs were first developed by faculty and students of Carnegie Mellon University in 2000 and were originally used to protect Yahoo's free internet resources [1, 3]. Since then, they have become an internet security standard to protect a wide variety of internet resources against automated attacks.

The major driving force behind the creation and use of CAPTCHAs is purely economic in nature. A large portion of the economy of the internet is based on advertising revenue. In most cases, users of free internet resources are indirectly paying for these resources through viewing ad content [5]. An automated process bypasses this system in the sense that a resource is being used, but a human is not viewing any ad content. In addition, automated processes can exploit these resources on a large scale. In many cases automated processes designed to solve CAPTCHAs are used to abuse a free internet resource by creating spam or unauthorized advertisements. It is important to understand the basic economics involved in order to understand why CAPTCHAs are a necessary security tool.

Given that CAPTCHAs have become a standard internet security technology, many usability issues have arisen. It has been shown in studies that blind or visually impaired individuals and those who suffer from certain learning disabilities can be unable to solve CAPTCHAs, which prevents them from accessing resources they should be able to use [8]. Usability issues are not limited to those with disabilities. It has been shown that text-based CAPTCHAs that implement heavily distorted characters have significantly lower solve rates by humans [8]. Many of the same properties that make CAPTCHAs more resistant to automated attacks also negatively impact their usability [2]. Since a good CAPTCHA is both user friendly and secure, this creates the need for a delicate balance between usability and robustness in text-based CAPTCHA design.

This paper will analyze several properties of text-based CAPTCHAs in terms of their effect on usability and security with respect to resistance to automated attacks. It will also outline an attack on the Microsoft CAPTCHA based on a series of algorithms created to exploit design weaknesses. Finally this paper will discuss the impact of CAPTCHAs on productivity and the reCAPTCHA project, which addresses this issue.

UMM CSci Senior Seminar Conference, May 2012 Morris, MN.

2. BACKGROUND INFORMATION

This section introduces terminology related to CAPTCHA design and automated CAPTCHA attacks.

2.1 CAPTCHA Text Properties

CAPTCHA texts have several properties that are defined within the field that are important to understanding their design. The majority of these properties are used to disrupt **Optical Character Recognition (OCR)** software, which is software designed to convert images of characters into text. For the purposes of this paper, we define the following terms. **Font** refers to the typeset and size of the text. The **character set** is the collection of characters used in a particular CAPTCHA scheme as well as their case sensitivity. **String text** is comprised of the length and makeup of the string encoded in the CAPTCHA image. **Distortion** is the use of attractor fields which alter the image by changing the relative location of pixels. This creates a deviation from a standard character pattern. **Tilting** is the rotation of characters to different angles throughout a CAPTCHA image. **Waving** refers to positioning tilted characters at various vertical locations creating a wave pattern with the textual foreground [2].

2.2 CAPTCHA Layout Properties

CAPTCHA images also have several layout properties that contribute to their design. Unlike text properties which disrupt OCR software, layout properties are generally used to combat character location by automated processes. A **complex background** is the use of a detailed background in an attempt to hide the foreground text. The idea is to impede an automated solver without disrupting a human's ability to locate the characters. **Stray lines**, sometimes refered to as arcs, are lines which are not part of the textual foreground. This strategy aims to artificially connect distinct characters or create pseudo-characters which may be incorrectly classified as a character by an automated process. **Collapsing** is the removal of space between characters, making it difficult for automated processes to distinguish one character from the next [2].

2.3 Automated CAPTCHA Attacks

Certain CAPTCHA attacks are general and intended to break a wide variety of CAPTCHAs with moderate to high success rates. Others are specifically tailored to a given CAPTCHA scheme and are created to exploit design weaknesses such as the Microsoft CAPTCHA attack covered in Section 4 of this paper. Regardless of the type of attack, automated processes designed to solve CAPTCHAs generally have three phases: The first is **pre-processing** which attempts to make the CAPTCHA image easier to analyze by removing color, removing complex backgrounds, and implementing noise reduction to clarify the image. The next phase is **segmentation** which involves locating the individual characters in a CAPTCHA image while preserving their order. The final phase is **classification** or recognition. Classification takes the segmented characters and converts them to the plain text used to solve the CAPTCHA [2]. This phase is primarily accomplished using standard OCR software [4, 7].

3. ANALYSIS OF CAPTCHA PROPERTIES

The degree to which various CAPTCHA properties are present and how they interact in a CAPTCHA scheme largely impact the scheme's overall usability and security. This section discusses several CAPTCHA properties and their known effects on both usability and security.

3.1 Character Set

A larger character set leads to a more secure CAPTCHA scheme. A larger number of possibilities decreases the odds that any given guess from an automated process is correct. However, this same concept negatively impacts usability. The increased security gained from using a large character set is negligible when compared to the decreased accuracy by human solvers. Studies have shown human solve rates as high as 98% in CAPTCHA images which use only numerical values (0-9) as the character set. This drops to 82% when using an alphanumeric and case sensitive character set (a-z,A-Z,0-9) [2]. It can be assumed that this human accuracy would be further decreased by including non-alphanumeric symbols in the character set. A large character set also leads to more characters, such as lowercase i and j, looking similar after distortion and blurring are applied [8]. This is known to create several usability issues in various widely used CAPTCHA schemes [2].

3.2 String Text

A short string length combined with a small character set can lead to a situation where random guesses can solve a CAPTCHA with a success rate higher than 0.01%, the industry standard for determining whether or not a CAPTCHA is secure. For this reason, having a sufficiently long string length is important. String length has opposite implications in terms of usability depending on whether or not the string text contains a dictionary word or a sequence of random characters. In the case of a dictionary word, increasing the string length directly correlates to an increase in user accuracy. However, there is negative correlation between string length and user accuracy when applied to a CAPTCHA scheme using a string of randomly generated characters. This has been attributed to the fact that human solvers are able to use surrounding characters to infer indistinguishable characters when working with a dictionary word. This obviously is not possible when working with a string of random characters. While using a dictionary word clearly increases usability, it has a negative impact on security which is similar to using a smaller character set as there are fewer possible solutions compared to strings composed of random characters. [8]

Another interesting aspect of the string text is the difference between a fixed-length and a variable-length scheme. A fixed-length string text has negative security implications [7]. If the number of characters in a given CAPTCHA image is known, the segmentation process of an automated attack can make assumptions allowing it to locate individual characters with a higher success rate. This is covered in more detail in the Microsoft CAPTCHA attack outlined in Section 4.

3.3 Distortion

Distortion is often given credit for being the most effective CAPTCHA property to disrupt classification by the OCR component of an automated attack. However, it can also heavily decrease the accuracy for a human solver if implemented excessively. Given that distortion alone is not able to completely resist classification, the usability issues caused by distortion are often more significant than the increased security [2].

3.4 Font

A CAPTCHA scheme that makes use of multiple fonts is more resistant to automated attacks in that OCR software will have a decreased success rate [2]. This is because varying fonts makes character size unpredictable. The use of multiple fonts has not been shown to negatively impact usability.

3.5 Complex Background

The goal of a complex background is to increase security by hiding the foreground text within the background making the CAPTCHA image more difficult for an automated process to segment (see Figure 1). In recent years complex backgrounds have been popular despite the fact it has been shown that they generally add very little to the overall security of a CAPTCHA scheme [2]. A common tactic is to use foreground and background colors that differ very little in their values on the RGB scale (a pixel represented in terms of its red, green, and blue values), yet are easily distinguishable by human vision. However, studies have shown this can be easily countered by converting color from the RGB scale to cylindrical coordinate color models that more accurately reflect human vision [2]. It has been determined that using very similar colors or an overly-complicated color scheme can create usability issues without necessarily increasing the overall security [8].

3.6 Stray Lines

There are three main types of stray lines which have varying impacts on segmentation. The first is small lines that do not cross characters. These have no proven security benefits as they can generally be removed in the preprocessing phase of an automated attack [7]. The second type is small lines that connect distinct characters (see Figure 2). These provide more of a security increase than the first type, but are not entirely segmentation resistant. A histogram attack, as discussed in Section 4, can yield high segmentation success rates against small connecting lines. The final type of stray lines is large lines that have the same thickness as characters in the CAPTCHA image. These have been proven to be an effective anti-segmentation measure in that it is very difficult for automated processes to differentiate them from characters [2]. Large lines must be implemented appropriately, as in matching the foreground color and staying within the CAPTCHA foreground, or they will be more easily detected by a segmentation algorithm [2]. Using the correct type of stray lines will significantly increase a CAPTCHA scheme's security through segmentation resistance without negatively impacting its usability.

3.7 Collapsing

Collapsing is often given credit for being the CAPTCHA property that most directly correlates to segmentation resistance [2]. However, predictable collapsing, which uses a fixed string length and character width, can still result in unacceptable success rates for automated segmentation processes. Predictable collapsing is a factor when the string length and approximate character width are known values. A segmentation algorithm can make high-probability guesses on where to segment, which leads to moderate success rates [2]. This is why it is much more secure to use collapsing with variable-length string text and multiple fonts. While collapsing generally does not create usability issues, studies have proven that extreme collapsing (more than a 5-pixel character shift) can drastically reduce the accuracy of human solvers [2].

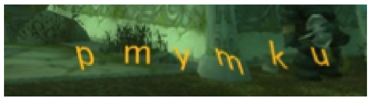

Figure 1: The Blizzard CAPTCHA makes use of a complex background in an effort to hide the text from automated processes while keeping the text recognizable to a human solver.

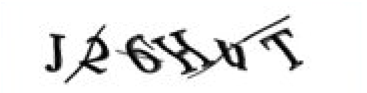

Figure 2: The Yahoo CAPTCHA makes use of thin connecting lines as the main resistance to automated segmentation attacks.

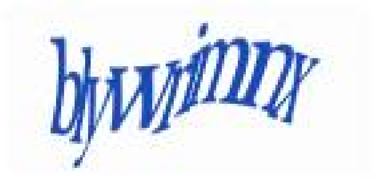

Figure 3: Google relies on collapsing as its main segmentation defense property in its CAPTCHA.

4. MICROSOFT CAPTCHA ATTACK

Various versions of the text-based Microsoft CAPTCHA have been in use since 2002 protecting Microsoft online services from automated processes designed to exploit the free resources. Although not everyone agrees [2], it is a commonly held belief that character recognition is trivial and therefore the majority of CAPTCHAs focus much more on segmentation resistance than techniques to disrupt character recognition. The Microsoft CAPTCHA was designed around segmentation resistance with the goal that an automated process should not be able to solve the CAPTCHA with a success rate higher than 0.01% (see Figure 4). This is on the same level as field-wide CAPTCHA standards. In 2008, Jeff Yan and Ahmad Salah El Ahmad [7] of Newcastle University, UK were able to segment the Microsoft CAPTCHA using automated processes with a success rate greater than 90%. Their attack focuses on the preprocessing and segmentation phases and leaves the recognition phase to common OCR (Optical Character Recognition) software.

While the Microsoft CAPTCHA was resistant to generic

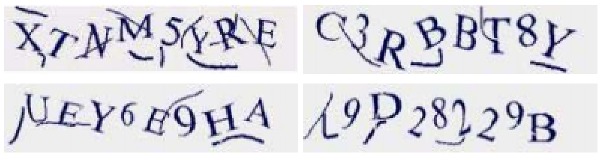

Figure 4: Four examples of the Microsoft CAPTCHA targeted by the segmentation attack [7].

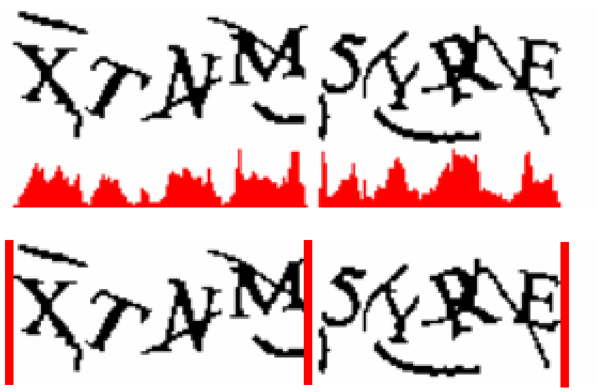

Figure 5: An example of histogram vertical segmentation with each bar representing the pixel count of the column directly above it in the top image. The long vertical lines in the bottom image represent segmentation breaks [7].

attacks, Yan and El Ahmad proved it was vulnerable to an attack based on algorithms designed to exploit several of its design weaknesses. The first step in their approach was to identify the properties of the Microsoft CAPTCHA and then decide which ones could be exploited. By analyzing 100 CAPTCHA images provided by the Windows Live account sign-up page, Yan and Salah El Ahmad concluded that the Microsoft CAPTCHA consistently exhibited properties making it vulnerable to an automated attack. Based on these properties, Yan and Salah El Ahmad tailored a six-step attack to remove the anti-segmentation arcs and locate each character in the CAPTCHA images.

The first step is preprocessing which starts by converting the colored image to only black and white. This is done by analyzing pixel intensity and converting those higher than a determined value to white and converting the rest to black. The exact threshold value was determined through empirical data based on the 100 images originally analyzed. This process occasionally separates characters in the image by converting a small number of pixels of the foreground (text) to the background color. To correct this, all pixels of the background color which have pixels of the foreground both to the left and right or above and below are converted to the foreground color. This process results in a black and white image with all characters intact.

The next step involves segmenting the characters vertically. This is done by creating a histogram representing the number of foreground pixels in each column (see Figure 5). In many cases, valleys in the histogram represent spaces between the characters. However in the algorithm, the image is only segmented into chunks split on columns with no pixels of the foreground color. This guarantees a correct split.

The third step separates each chunk from the previous step into objects represented with a unique color. An object is defined as a connected series of foreground pixels. They are detected by selecting a pixel of the foreground and converting it to a new color. Then every foreground pixel adjacent to a pixel of this color is converted to the given color until the whole object is detected. This process is repeated for non-connected foreground pixels using new colors until all foreground pixels have been converted. This ensures all objects have been located.

Next is the removal of arcs, which are small connecting lines as discussed in Section 3.6. Properties of arcs are exploited to distinguish them from valid characters. Using pixel count, shape, and location, it is possible to detect arcs for removal. Pixel count and location are straightforward. Arcs generally have significantly fewer pixels than characters and only arcs are found near the edge of the image. Shape is much more interesting in that characters may contain circles (0, 4, 6, A, B, D, etc.) but arcs never will. Circles are detected by isolating an object in a rectangular box. Starting at one corner, all adjacent background pixels are converted to the new color as well. This continues with all pixels adjacent to a pixel of the new color being converted to the new color. When all pixels that satisfy the requirement have been converted, if any of the original background color remains, there is a circle present. This means the object definitely contains a character. Figure 6 shows a visual representation of this process. All objects not containing circles are analyzed in terms of their other properties and are removed if the probability of being an arc is significant.

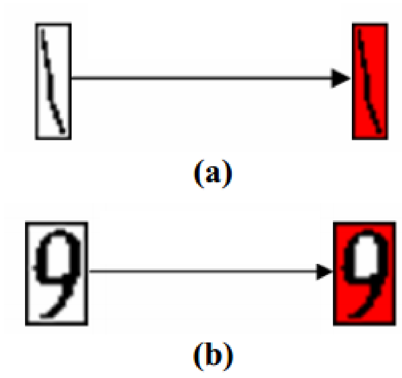

Figure 6: A visual representation of the circle detection algorithm when applied to a stray line (a) and a character containing a circle (b). Any remaining pixels of the original background color after the algorithm has been completed indicates a circle is present. [7].

The final two steps involve locating and segmenting connected characters. This relies heavily on three assumptions that hold true for all Microsoft CAPTCHA images: there are exactly eight characters in the string text, characters are never positioned above and below each other, and the same font is used for all CAPTCHA images. Given the number of chunks separated by the second step and the known amount of characters, the algorithm uses the chunk width to locate connected characters within the given chunks. The chunks are then segmented by dividing them into sections of equal width, each containing one character. This is possible because a constant font allows for assumptions to be made on approximate character width.

At this point the image is entirely segmented and OCR software is used to solve the CAPTCHA. The success rate for the segmentation phase of this attack was reported to be 92%. The success rate assumes the first attempt is correct for this attack as its focus is on correct segmentation in one attempt. Nearly all CAPTCHAs allow multiple attempts per request as human solvers naturally will make occasional mistakes [2]. When factoring in standard OCR software, it is estimated that the success rate of the automated attack as a whole is approximately 60%. It is important to note that the data was gathered by testing this attack on 500 randomly generated CAPTCHA images provided by the same source used to obtain the images used for the original analysis. The attack averaged 84.2 milliseconds to segment each image which is entirely reasonable for a large scale attack. This proves that the Microsoft CAPTCHA was vulnerable to an automated attack based on exploiting its design weaknesses.

5. EFFICIENCY AND RECAPTCHA

While solving a CAPTCHA may take the average human user about 10 seconds, when put into perspective with how many CAPTCHA images are solved each day worldwide, this adds up to a significant amount of time lost in order to protect resources from automated attacks. It is estimated that over 200 million CAPTCHAs are solved by human users each day [3], which adds up to more than 500,000 hours of lost productivity on a daily basis. Although this seems to be a large price to pay, the alternative could lead to free internet resources becoming virtually unusable due to the sheer amount of spam and resources lost to unintended automated consumers.

To combat this problem of lost work, Luis von Ahn, a pioneer of CAPTCHA technology at Carnegie Mellon University, developed the reCAPTCHA project [6]. The goal of reCAPTCHA is to utilize the results of solved CAPTCHAs in a meaningful way so that the effort in solving CAPTCHAs is not entirely wasted. Von Ahn's idea is centered around using the data generated by users solving reCAPTCHA images to assist OCR software in digitizing books and archives such as those from the New York Times (see Figure 7). Google acquired reCAPTCHA in 2009 and has been using it ever since in the way von Ahn had intended [6].

What makes reCAPTCHA unique is that it contains two separate CAPTCHA problems. The first is very similar to traditional CAPTCHAs. It is a known sequence of characters altered to resist an automated attack. The second is a scanned word directly from text that OCR software has failed to recognize. A given word is sent to reCAPTCHA

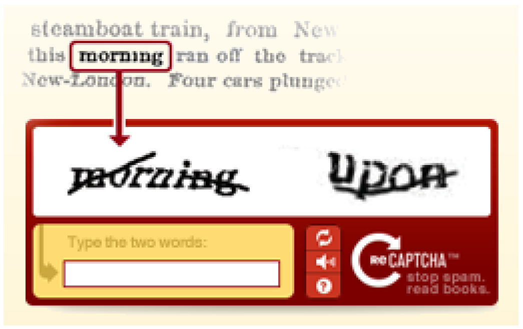
steamboat train, from New
this morning ran off the track
New-London. Four cars plunged
morning
upon
Type the two words:
reCAPTCHA
stop spam. read books.

Figure 7: An example of a reCAPTCHA image demonstrating how it implements an unknown word from scanned text (left) and a known control word (right) [3].

when two separate OCR programs yield different results or if the result is not a dictionary word. The first CAPTCHA problem is treated as the control, in that if solved correctly by the user, the user response to the second CAPTCHA problem is assumed to be correct as well [3]. Once a given word has been solved sufficiently many times to create a reasonable confidence interval, reCAPTCHA returns the text value to the digitization process.

In addition to being a solution to the lost productivity problem, reCAPTCHA is one of the most user-friendly [8] and secure CAPTCHA schemes currently being used [2]. Large-scale studies have shown that reCAPTCHA has a solve rate as high as 97% which clearly puts reCAPTCHA near the top of the field in terms of usability [8]. With respect to security, reCAPTCHA is similar to many of the most popular text-based CAPTCHAs in that they share many of the same anti-segmentation and anti-classification features. Various generations of reCAPTCHA have demonstrated distortion, blurring, tilting and waving to resist classification by OCR software and stray lines, in combination with collapsing, for anti-segmentation purposes [8, 6]. What sets reCAPTCHA's robustness apart is that it uses **Adaptive Security**, a term created by Google. All reCAPTCHA images are generated by Google's reCAPTCHA image generation servers and in the event that an automated process is able to solve a reCAPTCHA image (usually evidenced by a spike in spam activity), image alterations can be made in very little time with nearly instantaneous deployment [3]. This guarantees that even if reCAPTCHA is compromised, it can recover quickly with no downtime, which makes it one of the most secure CAPTCHA schemes in use.

While time spent solving CAPTCHAs is still lost at the level of an individual, the reCAPTCHA project has succeeded in harnessing that data in order to complete a worthwhile task. It is reported that over 100 million reCAPTCHA images are solved each day which all contribute to to the digitization of various forms of print media [6]. It is also important to note that although reCAPTCHA requires users to solve two separate CAPTCHA images, it takes no longer on average than solving other widely-used CAPTCHAs [3]. This ensures that reCAPTCHA doesn't unnecessarily contribute to the productivity problem it was created to address.

6. CONCLUSION

CAPTCHA design is very complex because of the delicate balance of security and usability necessary to create a successful CAPTCHA. Creating a CAPTCHA that is so secure that no human can solve it, or so user friendly that it is a trivial task for CAPTCHA breaking software, is very easy to accomplish. A successful CAPTCHA, by its definition, is able to tell humans and computers apart. Creating a CAPTCHA scheme that is unsolvable by the latest CAPTCHA solving software and is still user friendly requires an understanding of important CAPTCHA design properties.

The goal is to add security features whenever possible as long as they do not significantly or unnecessarily decrease the accuracy of human solvers. In terms of increasing security by resisting segmentation, collapsing, and the use of thick lines tend to be the best options. Both of these properties make the location of each character much more difficult to determine for an automated process if implemented correctly and do not drastically reduce usability. A complex background may seem to add value, but in nearly all cases it can be removed in the preprocessing phase of an attack or it greatly reduces a CAPTCHA scheme's usability to the point where it is no longer serving its purpose. Many of the most effective properties in resisting classification can also make a CAPTCHA unusable. An unnecessarily large character set or heavily distorted characters will certainly increase security, but at an extreme cost to usability. The best CAPTCHA schemes use smaller character sets without using characters that look very similar to each other after collapsing or distortion is applied. Distortion only should be used lightly as it is effective in disrupting OCR software, but excessive distortion can lead to usability issues. A solid CAPTCHA design resists both segmentation and classification, but it seems to be the case that it is much easier to resist segmentation than classification while maintaining a high level of usability. As CAPTCHA solving software becomes more powerful, CAPTCHA security will have to increase as well. It is an ongoing battle between those who wish to exploit internet resources and those who are protecting them.

Although at times CAPTCHAs can seem like a nuisance, they are an extremely important tool in internet security. The alternative is intolerable amounts of spam and resources overly consumed by automated processes. Projects like reCAPTCHA are attempting to make use of CAPTCHAs in meaningful ways so the effort put into solving CAPTCHAs is not entirely lost. CAPTCHAs are a fundamental tool for protecting internet resources and will remain extremely important to internet security for the foreseeable future.

7. ACKNOWLEDGMENTS

Thanks to Kristin Lamberty, Elena Machkasova, and Rob Jansen for their helpful feedback and advice throughout the process of writing this paper.

8. REFERENCES

[1] L. V. Ahn, M. Blum, N. J. Hopper, and J. Langford. Captcha: using hard ai problems for security. In *Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques*, EUROCRYPT'03, pages 294–311, Berlin, Heidelberg, 2003. Springer-Verlag.

[2] E. Bursztein, M. Martin, and J. Mitchell. Text-based captcha strengths and weaknesses. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 125–138, New York, NY, USA, 2011. ACM.

[3] Google-ReCAPTCHA. Telling humans and computers apart automatically. `http://www.google.com/recaptcha/captcha`, Mar. 2012.

[4] 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 the 26th Annual Computer Security Applications Conference*, ACSAC '10, pages 171–180, New York, NY, USA, 2010. ACM.

[5] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G. M. Voelker, and S. Savage. Re: Captchas: understanding captcha-solving services in an economic context. In *Proceedings of the 19th USENIX conference on Security*, USENIX Security'10, pages 28–28, Berkeley, CA, USA, 2010. USENIX Association.

[6] Wikipedia. Recaptcha — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Recaptcha&oldid=176827364`, 2007. [Online; accessed 8-March-2012].

[7] J. Yan and A. S. El Ahmad. A low-cost attack on a microsoft captcha. In *Proceedings of the 15th ACM conference on Computer and communications security*, CCS '08, pages 543–554, New York, NY, USA, 2008. ACM.

[8] J. Yan and A. S. El Ahmad. Usability of captchas or usability issues in captcha design. In *Proceedings of the 4th symposium on Usable privacy and security*, SOUPS '08, pages 44–52, New York, NY, USA, 2008. ACM.