Attacking Image Recognition CAPTCHAs
A Naive but Effective Approach

Christoph Fritsch, Michael Netter, Andreas Reisser, and Günther Pernul

Department of Information Systems,
University of Regensburg, 93053 Regensburg, Germany
{christoph.fritsch,michael.netter,andreas.reisser,guenther.pernul}
@wiwi.uni-regensburg.de
http://www-ifs.uni-regensburg.de

Abstract. The landscape of the World Wide Web today consists of a vast amount of services. While most of them are offered for free, the service providers prohibit their malicious usage by automated scripts. To enforce this policy, CAPTCHAs have emerged as a reliable method to setup a Turing test to distinguish between human and computers. Image recognition CAPTCHAs as one type of CAPTCHAs promise high human success rates. In this paper however, we develop an successful approach to attack this type of CAPTCHA. To evaluate our attack we implemented a publicly available tool, which delivers promising results for the HumanAuth CAPTCHA and others. Based upon our findings we propose several techniques for improving future versions of image recognition CAPTCHAs.

Keywords: CAPTCHA, Image Recognition CAPTCHA, HumanAuth, Experimentations, Security Analysis.

1 Introduction

Today's highly networked world is based on a vast amount of electronic services provided and requested via the world wide web. A large number of these services, i.e. e-mail and social networks, is available free of charge, solely requiring the user to register with the service provider. Yet the last decade has shown an increasing interest in abusing Internet services for malicious economical reasons. E-mail accounts, frequently available free of charge after initial registration, are misused for sending SPAM and phising mails to a plethora of plagued Internet users. There are still many more impermissible but still feasible and reasonably economic ways to abuse services trough the Internet.

To prevent corruptive usage of these services by automated scripts and thereby mitigate the threads illustrated above, in the majority of cases it is sufficient to remotely distinguish between humans and machines. Different approaches for such so-called Turing tests, generally known as CAPTCHAs[1], HIPs[2] or POSHs[3]

[1] Completely Automated Public Turing test to tell Computers and Humans Apart.
[2] Human Interactive Proof.
[3] Puzzle Only Solvable by Humans.

S. Katsikas, J. Lopez, and M. Soriano (Eds.): TrustBus 2010, LNCS 6264, pp. 13–25, 2010.

have been proposed and are currently under active development and deployed to various systems. All of them are built upon problems that can easily be solved by humans but are very hard for machines to solve. Likewise, problems that emerged to be very hard challenges in the field of artificial intelligence (AI) are often implemented in one form or another into CAPTCHAs.

In this paper we present an approach, that does not try to solve the hard AI problem behind CAPTCHAs but forges a different solution, also known as side-channel attack. We focus on elementary image processing and color value distribution calculations to precompute characteristic attributes for all images of the chosen image recognition CAPTCHA implementation. Based on these characteristic attributes, our approach is able to recognize randomly distorted CAPTCHA images with impressive precision. We evaluated the applicability of our approach against several implementations (HumanAuth, Microsoft ASIRRA, UMIST FACES) to prove its reliability.

The main contributions of this paper are the following: We develop an attack on image recognition CAPTCHAs, called PixelMap. We evaluate our approach, attacking several image recognition CAPTCHAs with very promising results. Based upon our findings we propose several techniques to harden future image recognition CAPTCHAs from being vulnerable to this kind of attacks. Furthermore we implemented a prototype tool to show the practical applicability of the presented approach.

The remainder of this work is structured as follows: We start in Chapter 2 with an overview of related work and other approaches for automatically solving CAPTCHAs. Subsequently, in Chapter 3 we briefly explain the general CAPTCHA approach and different kinds of CAPTCHAs before we comprehensively introduce and evaluate our approach for attacking image recognition CAPTCHAs in general and the HumanAuth scheme in particular in Chapter 4. In Chapter 5 we infer ideas for both, improving our attack and tweaking image modification procedures to impede future attacks. Finally, Chapter 6 summarizes our findings.

2 Related Work

The first thoughts related to the field of 'Automated Turing Tests' were written down by Naor in 1997 [9]. Since the introduction of the term CAPTCHA by von Ahn et al. [12] a wide variety of CAPTCHAs showing different characteristics have been developed. Banday and Sha differ between three classes: Text-based (OCR), image-based and audio-based CAPTCHAs [1]. As CAPTCHAs are used for security purposes, different attack-schemes have emerged. Most attacks are based on OCR, meaning that they try to solve the underlying AI problem [8], [7], [14].

Furthermore, side-channel attacks exist trying to circumvent the AI problem. For example, Hernandez-Castro et al. propose a side-channel attack scheme on Microsoft's image-recognition based CAPTCHA ASIRRA [6] and the HumanAuth CAPTCHA (see Chapter 3.2), using different statistical test on the ASIRRA database and the HumanAuth image library. In [5] another attack on ASIRRA

has been published, using support vector machines together with color and text processing for classification.

Yan and El Ahmad published an attack, using both, side-channel techniques as well as tackling the AI-problem, on an OCR-resistant text-based CAPTCHA [16]. The authors combine different techniques: First, they automatically separate foreground text from background to perform a text-segmentation. By counting the number of foreground pixels in each segment, they are able to identify characters according to an unique pixel count in most of the time, using a dictionary approach additionally in order to reveal unidentified characters.

3 CAPTCHA – A Modern Kind of Turing-Test?

In the following we shortly lay out background and implementation of different kinds of CAPTCHAs before we briefly pick up image recognition CAPTCHAs and HumanAuth as the implementation we attacked in particular.

3.1 Background and Alternatives

A CAPTCHA is defined as a challenge-response test, generated by a computer, that only a human can solve. Hence CAPTCHAs are suitable to prevent corruptive usage of web services by automated scripts. Since Ahn et al. [12] proposed the concept of CAPTCHAs, industry and researchers have developed a variety of different CAPTCHAs. Usually CAPTCHAs are based on the three principles developed by Chew and Tygar [3]: (1) Easy for humans to solve. (2) Hard for computers to solve. (3) Easy to generate and verify.

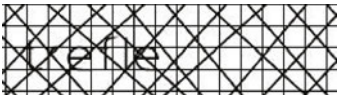
Hard artificial intelligence (AI) problems are often used to construct CAPTCHAs fulfilling the first and the second principle. Those problems are difficult to solve without special knowledge, i.e. the problem context, which is usually available to humans but not to computers. Several types of CAPTCHAs exist relying on various human sensory abilities, such as seeing and hearing.

The third principle is hard to fulfill. Similar to cryptographic functions and for security reasons, the algorithm to create a CAPTCHA is usually publicly available. Therefore a CAPTCHA generation algorithm must be able to quickly create a large number of unpredictable tests. Furthermore, the verification algorithm must know the solution a priori, since the verification is done by a computer that itself cannot solve the generated CAPTCHA.

To solve the verification problem two types of CAPTCHAs have emerged [4]: Algorithm- and database-based CAPTCHAs. The first uses an algorithm that is initialized with a secret random number to derive a challenge, allowing the verification to prove or disprove the test response. Thus the security and effectiveness for this type of CAPTCHAs depends on the secrecy of the random number. The second type employs a large database of preclassified challenges and an algorithm to randomly select single challenges. Using the classification, the verification algorithm can evaluate the test response.

The underlying principle of the most widespread type of CAPTCHAs is optical character recognition (OCR). The correct identification of characters in an image

(a) Google Captcha



(b) Unreadable CAPTCHA

Fig. 1. OCR-based Captchas

is a hard AI problem, especially if the characters are distorted (see Figure 1(a)). Using the definition introduced above, OCR Captchas are usually algorithm-based. The advantage of this type of Captcha is the ability to easily generate a large number of text strings, that are distorted and displayed as a Captcha. However this approach has a major shortcoming. Text with minimal distortion is easily readable by current OCR algorithms. Introducing more noise makes the Captcha very hard to solve even for humans (see Figure 1(b)) [15]. Thus there is a small gap between human and non-human success rates [4] and human computer interaction (HCI) research is heavily involved.

To overcome the shortcomings of OCR-based Captchas, several approaches have been proposed using different types of AI problems, such as the recognition of spoken letters (so-called audio Captchas) and image classification.

3.2 Image Recognition Captchas

Chew and Tygar [3] were among the first to use images (usually photographs) to create a new type of Captchas, called image Captcha. While several variations of image Captchas exist, the kind we focus on – image recognition Captchas – requires to understand what is depicted on an image which constitutes a hard AI problem [2]. Image recognition Captchas are usually database-based, thus requiring a database of preclassified images. Microsoft's Asirra is a well-known example [4]. Asirra presents a list of cat and dog images, asking the user to identify the cat images.

Studies indicate that the biggest advantage of image recognition Captchas is the improved human success rate compared to OCR-based Captchas [3], [4]. The main shortcoming is the dependency on a database of images that are preclassified by humans. Furthermore the database needs to be large and updated frequently with new images. Otherwise an adversary can create hashes of all images (similar to the rainbow table approach in cryptography [10]) and lookup a questioned image hash in his database.

3.3 The HumanAuth Captcha

HumanAuth[4] is an image recognition Captcha implementation, written in PHP and released under the GPL version 2 license. It presents to the user nine images chosen from an enclosed image database, three of them being *nature* and the

[4] `http://sourceforge.net/projects/humanauth/`

other six images being *nonnature* images. To solve the Captcha, the user has to select the three *nature* images. HumanAuth constitutes a hard AI problem as the user has to understand what the images represent.

In order to protect from side-channel attacks, HumanAuth places a randomly positioned watermark on every image, rendering a precalculation of image hashes for each image in the database useless.

4 PixelMap – An Approach to Attack Image Captchas

Within this section we present PixelMap as our approach to attack image recognition Captchas, especially the previously mentioned HumanAuth implementation. The basic idea behind PixelMap is that even distorted images are at least very similar for the better part of the image area.

4.1 General Approach

Based on a Captcha's freely available database of categorized source images (the learn images), we first pre-calculate a characteristic and distinguishing attribute, our so-called PixelMap (see Section 4.2) for each image (step 1 in Figure 2). To automatically solve a Captcha, we calculate the PixelMap for each of the 'test images' (step 2 in Figure 2) and compare it to the pre-calculated PixelMaps of the original image database (step 3 in Figure 2). This comparison detects the 'learn image' which is most similar to the current test image. We know the learn image's category from the learning phase and assume the same classification for the test image. For the HumanAuth example this means that we finally pick a test image as *nature* image if the most similar learn image is classified as *nature* and vice versa.

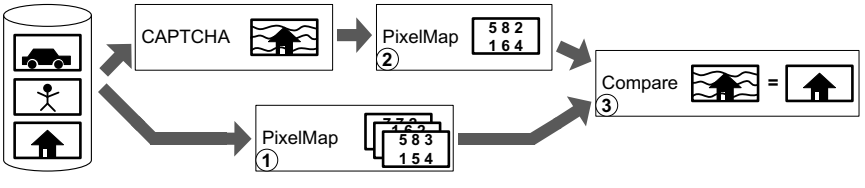


Fig. 2. PixelMap – Approach Outline

Please note that the image comparison has to be conducted in a fuzzy manner as the test image is distorted in some way or another, for example by an overlaid watermark. These image distortions however, do regularly not affect the whole image but only a minor part of the image area. Our analysis has shown that throughout the remaining image area, corresponding learn and test are at least very similar if not completely identical.

4.2 Fuzzy Image Recognition Algorithm

Algorithms 1, 2 and 3 show the foundations of our fuzzy image comparison approach. As mentioned above, we calculate a PixelMap based on which we compare two images fuzzily.

Algorithm 1. Image Classification Algorithm

```
function computePixelMap(img)
    pixelMap ← ARRAY[img.WIDTH * img.HEIGHT]
    k ← 0
    for all pxl ∈ img do
        pixelMap[k] ← pxl.ALPHA + pxl.RED + pxl.GREEN + pxl.BLUE
        k ← k + 1
    end for
    return pixelMap
end function
```

Each single pixel of an image consists of red, green, blue (RGB) and alpha values. The mixture of the RGB-values establishes the color of the pixel while the alpha value represents its transparency. All four values vary within the range from 0 to 255. Our PixelMap is simply calculated by summing up the RGB and alpha values for each single pixel of an image, resulting in a sum between 0 and 1020. For example a purely red pixel thus gets the value 510 as its red value is 255, its alpha value is as well 255 because pixel opacity is 100%, and the remaining green and blue values are 0. Please note that consequently the resulting PixelMaps for i.e. purely red and purely green images do not differ but are exactly identical. Algorithm 1 shows the calculation of the PixelMap in pseudo code. This PixelMap is pre-calculated and cached together with the classification for each undistored image of the underlying image database.

Algorithm 2. Image Identification Algorithm

```
 1: function uncoverImage(testimg, img_class_db)
 2:     max_sim_img
 3:     max_sim ← 0
 4:     for all img ∈ img_class_db do
 5:         sim ←CompareImages(testimg, img)
 6:         if sim > max_sim then
 7:             max_sim_img ← img
 8:             max_sim ← sim
 9:         end if
10:     end for
11:     return max_sim_img
12: end function
```

To uncover the classification of a test image, we check it against the image database. Algorithm 2 shows the basic procedure. Based on its previously calculated PixelMap we compare the possibly distorted test image fuzzily against each single undistorted learn image in the image database. As a result we receive the most similar learn image from which we know its classification.

Algorithm 3. Fuzzy Image Comparison Algorithm

```
1: function compareImages(img1, img2)
2:     no_identical_pxls ← 0
3:     for all pxl_id1 ∈ img1 and pxl_id2 ∈ img2 do
4:         sim ← Compare(pxl_id1, pxl_id2)
5:         if sim < THRESHOLD then
6:             no_identical_pxls ← no_identical_pxls + 1
7:         end if
8:     end for
9:     return (no_identical_pxls/all_pxl)
10: end function
```

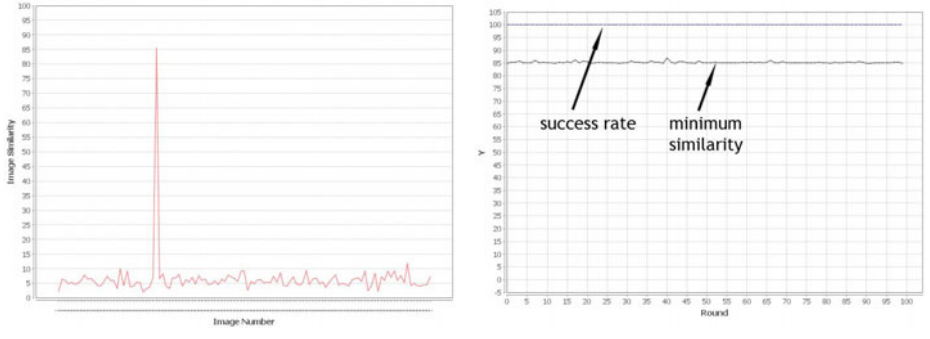
Finally, the pseudo code of our unpretentious image comparison algorithm is shown in Algorithm 3. It simply accepts two images as inputs which are fuzzily compared pixel by pixel based on the values of their PixelMaps. The fuzziness is implemented by a threshold which defines by which amount the pixel values of a single pixel in both images may differ before this pixel is rated as not matching. For our current approach a threshold of 20, which is about 2 percent of the maximum pixel sum, shaped up as suitable from several tests. The fuzzy image comparison algorithm simply returns the percentage of pixels that are similar between both images within the defined threshold.

As a result of these three fundamental building blocks of our image comparison approach, we receive for each distorted test image the unmodified image that is most similar to it together with its classification and a numeric similarity value ranging between 0-100 percent. Figure 3(a) shows the result in a graphical form. The abscissa marks all images, i.e. nature and non-nature images, from the image database while the ordinate marks the similarity of the test image to each of the learn images. Figure 3(a) is calculated using the original HumanAuth image database and a test image generated by the original HumanAuth implementation therefore containing a watermark. As can be seen from the graph, the similarity between the test image and the learn image most similar to it is about 85 percent. In other words, almost 85 percent of all pixels of the learn and the test image do not vary more then the previously defined threshold. Of particular interest is the significant gap between the similarity to the single most similar image (the peak in Figure 3(a)) and the much smaller similarity to all all other images. This result suggests that test images have to be distorted significantly before our algorithm does no longer accurately match correct test and base images (see Figure 6).

4.3 Evaluation

As we have shown in Figure 3(a), our image identification approach quite clearly identifies single distorted images based on the HumanAuth Captcha implementation. To evaluate our approach in more detail, we checked it more accurately against HumanAuth and applied it to several other Captcha image databases.

To effectively evaluate our approach, we implemented a HumanAuth simulator within our prototype (see Section 4.4). The simulator randomly choses 3 nature and 6 non-nature images from the image database and embeds the HumanAuth default watermark using the default opacity. In a second step we fed these distorted images into our image identification algorithm from which we receive the 9 most similar undistorted images. Finally, we checked whether the correct undistorted images have been uncovered. If and only if (a) the 3 nature images have been uncovered correctly as nature images and (b) no further actually non-nature image has wrongly been classified as nature image, the Captcha is solved correctly. Figure 3(b) shows the results for one hundred simulator rounds (marked on the abscissa). The lower dotted black line shows the minimum similarity of correctly identified nature images to the image it has been derived from. The upper dashed blue line shows the success rate for solving the HumanAuth Captcha, i.e. the percentage of correctly classified distorted nature images. As can easily be seen, our approach exhibits a 100 percent accuracy for the default HumanAuth settings. These results remained stable for several test runs with randomly selected images and watermark positions.



(a) Similarity between Distorted Test Image and Original Base Images

(b) Image Identification Results from our Simulator

Fig. 3. Evaluation of the Image Identification Algorithm

Please note that for solving the HumanAuth Captcha it is not essential to correctly identify the base image of each distorted image. It rather suffices to correctly classify each image correctly as (non-)nature even if the correct base image is not uncovered. In case our algorithm uncovered an image as a nature one but not as the correct base image, Figure 3(b) would have shown an spacious dashed red line below the blue one.

Additionally, we performed a broader evaluation against two more image databases, Microsoft's ASIRRA[5] and the UMIST FACES database[6]. We applied our attack on a publicly available excerpt of the ASIRRA database, which contains 30000 images. The large amount of images leads to a higher similarity among the images. However since the images are not modified, this does not affect our fuzzy image comparison algorithm. The UMIST FACES database consists of black and white images, showing peoples' faces photographed in front of a white background. This setting implies that large parts are similar for all images. However, identical pixels, i.e. the pixels showing a white background are implicitly ignored by our algorithm, leaving the rest of the image for comparison. Hence the results of our attack on this database are identical to attack on the HumanAuth database.

4.4 Prototypical Implementation

We have implemented a prototype to attack image recognition CAPTCHAs. The prototype is available for download and testing on our Web site[7]. We integrated the HumanAuth images as a sample image database to allow a broad audience to easily evaluate the tool. The prototype comes with a variety of functions that are shortly explained hereafter.

Figure 4 depicts the HumanAuth solver which allows solving CAPTCHAs created by an arbitrary HumanAuth instance. Our prototype fetches the images from a HumanAuth instance we set up at our Web site[8] (see Figure 4(a)) and can easily solve it using our PixelMap approach (see Figure 4(b)).

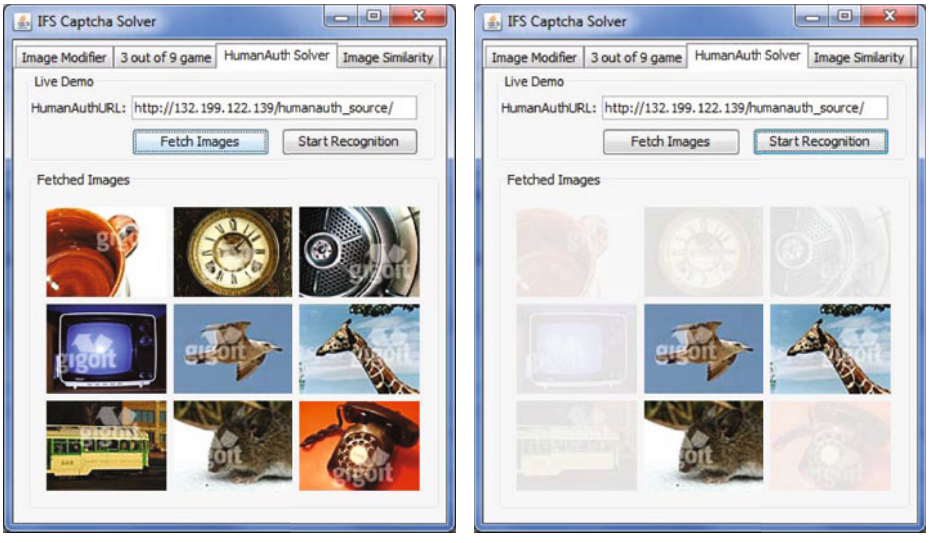
Furthermore, to evaluate the effectiveness of our approach we implemented a generic CAPTCHA simulator. This allows for rapdily testing our attack against all database-based CAPTCHA with an arbitrary number of iterations. The simulator also implements a variety of image modification functions to test the success rate of our approach against hardened CAPTCHA instances.

For analytical reasons the prototype also contains image comparison functionalities to examine our PixelMap approach in a controlled environment. Analogous to the simulator it also implements the image modification functions. To enable an in-depth analysis of the results, most functions allow for a graph visualization (see for instance Figure 3(a) and 3(b)).

5 Ideas for Improvement and Future Work

Based on the results of our current PixelMap approach depicted in Chapter 4 we identified several ideas for both improving our attack and improving image choice and distortion to generate improved CAPTCHAs. These ideas are described shortly in the following.

[5] http://research.microsoft.com/en-us/projects/Asirra/corpus.aspx
[6] http://www.shef.ac.uk/eee/research/vie/research/face.html
[7] http://www-ifsresearch.wiwi.uni-regensburg.de/paper/captcha/solver/
[8] http://www-ifsresearch.wiwi.uni-regensburg.de/paper/captcha/humanauth/



(a) Fetched CAPTCHA Challenges (b) Correctly Identified Nature Images

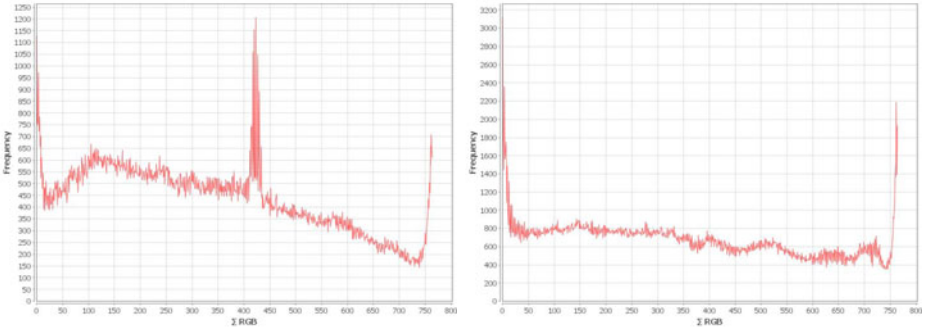
Fig. 4. IFS Captcha Solver

5.1 Improving the Attack

Although delivering promising results compared to other approaches [6], we identified several starting points for improving our approach. As stated in Section 3.2, a major shortcoming of many image CAPTCHAs is the dependency on a preclassified image database. In the majority of cases the image database is packaged with the CAPTCHA and thus available to an attacker. Other types of CAPTCHAs, such as ASIRRA [4] are based on a secret image database which is constantly being extended by new images. To prove the general applicability of our approach, we sketch two strategies to attack this second kind of CAPTCHAs.

Using the secrecy of the image database as the main pillar for the security of a CAPTCHA protects only poorly from exploitation by an attacker. Unlike classical cryptosystems where the key remains always secret, a CAPTCHA reveals a small portion of its database each time a new test is created and new images are displayed. An attacker is able to store these images and classify them e.g. using a crowd-sourcing approach such as the ESP Game [13]. This process can be repeated until the whole database is downloaded and classified, making the CAPTCHA fully exploitable with our approach. Thus the security solely depends on the increased effort to reconstruct the secret database, rendering the security of this type of CAPTCHA an economical question.

Another strategy to attack secret database CAPTCHAs without the need to download the whole image database might be used to correctly classify previously unclassified images. Based on our PixelMap approach, we examined the color distribution of the HumanAuth images and found an interesting property.



(a) Nature Images (b) Nonnature Images

Fig. 5. RGB Value Frequency for HumanAuth Nature and Nonnature Images

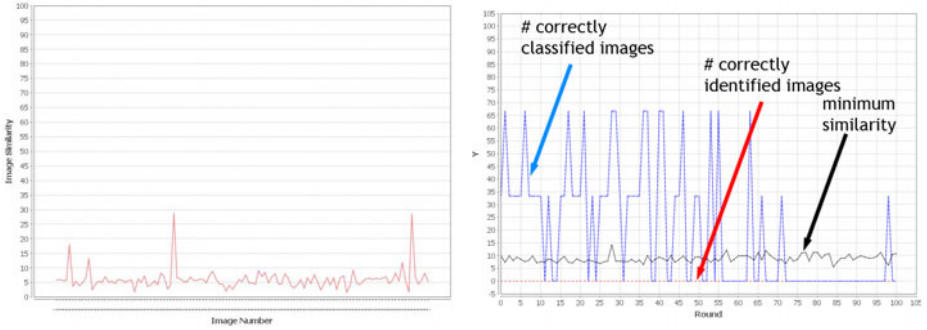
Figure 5 depicts the cumulated and aggregated RGB value frequency for nature and non-nature images. The abscissa contains the cumulated, i.e. ALPHA + RED + GREEN + BLUE values and the ordinate holds frequency of those values among all images in the database. While both graphs in Figure 5 show a similar aggregated RGB value frequency the peak between 410 and 430 on the abscissa is evident. Nature images contain an above-average frequency of a certain small range of RGB values, which is reducible to their high portion of green and/or blue pixels. Using this characteristic property a fuzzy image comparison algorithm might be employed to classify previously unknown images. This approach is slightly similar to a method for classifying images in two semantic classes, namely photographs and graphics developed by Oliviera et al. [11].

5.2 Improving Image Choices

Section 3.2 identified a public image database as one of the major shortcomings of many image recognition CAPTCHAs like HumanAuth making them vulnerable to our attack. Furthermore in Section 5.1 we demonstrated the general applicability of our approach even for secret image database CAPTCHAs. In this section, we outline several improvements for image CAPTCHAs preventing fuzzy classification attacks like our PixelMap approach.

It must be acknowledged that an attacker can always obtain the image database and preclassify all images. Therefore the security of image CAPTCHAs must not depend on the image database at all. To effectively improve the security of image recognition CAPTCHAs, preclassification of all images and all potential variations and modifications thereof has to be uneconomically.

We propose to randomly modify each image using several image modification algorithms simultaneously. A randomly placed watermark prevents hashtable attacks, since the hash of an image is different for each position of the watermark. However we have showen in Section 4.2 that a watermark alone does not protect from fuzzy image classification. To improve that, we propose a combination of



(a) Image Recognition Rate After Slight Zoom

(b) Combined Variation of Several Parameters

Fig. 6. Image Modification Improvements Applied

several image modification algorithms: Watermark size, watermark alpha, image zoom, image alpha, image color and image flip. Figure 6(a) depicts the similarity of a slightly zoomed test image to all other images. Please note the decrease in similarity from 85 percent for original HumanAuth images (see Figure 3(a)) to 30 percent after applying our image modification techniques. As a result the previously evident gap between similarities of the test image and the single most similar image is nonexistent.

The combined application of several image modification algorithms creates a slight variation of the original image which is still easily recognizable by humans. To evaluate the effectiveness of our approach, we implemented a modified version of the HumanAuth CAPTCHA that applies the proposed image modification algorithms. Figure 6(b) depicts the associated success rate for our modified HumanAuth implementation. Compared to the original HumanAuth, where the success rate of our attack is 100 percent, the modified version of HumanAuth lowers the success rate depending on the applied image modifications to 0-3 percent. Further iterations show that these results remain stable.

6 Conclusions

In this paper we present PixelMap, a side-channel attack on image recognition CAPTCHAs. Based on a fuzzy image comparison, our approach clearly identifies the shortcomings of several currently existing image recongition CAPTCHAs. To evaluate our approach we applied our attack on several image recognition CAPTCHAs, especially on HumanAuth with very promising results. We implemented a prototype tool that is available for download to demonstrate the practical applicability of our attack. Building upon our findings we develop several image modification techniques to protect future versions of image recognition CAPTCHAs, preventing image preclassification while not having an impact on human recognition success rates.

References

1. Banday, M.T., Shah, N.A.: Image flip captcha. ISC International Journal of Information Security (ISeCure) 1(2), 105–123 (2009)
2. Barnard, K., Duygulu, P., Forsyth, D.A., de Freitas, N., Blei, D.M., Jordan, M.I.: Matching words and pictures. Journal of Machine Learning Research 3, 1107–1135 (2003)
3. Chew, M., Tygar, J.D.: Image recognition captchas. In: Zhang, K., Zheng, Y. (eds.) ISC 2004. LNCS, vol. 3225, pp. 268–279. Springer, Heidelberg (2004)
4. Elson, J., Douceur, J.R., Howell, J., Saul, J.: Asirra: a captcha that exploits interest-aligned manual image categorization. In: Proc. of the 14th ACM Conference on Computer and Communications, CCS '07 (2007)
5. Golle, P.: Machine learning attacks against the asirra captcha. In: Proc. of the 15th ACM Conference on Computer and Communications Security, CCS '08 (2008)
6. Hernandez-Castro, C.J., Ribagorda, A., Saez, Y.: Side-channel attack on labeling captchas. Computing Research Repository (08/2009)
7. Mori, G., Malik, J.: Recognizing objects in adversarial clutter: Breaking a visual captcha. In: Proc. of the 16th IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR '03 (2003)
8. Moy, G., Jones, N., Harkless, C., Potter, R.: Distortion estimation techniques in solving visual captchas. In: Proc. of the 17th IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR '04 (2004)
9. Naor, M.: Verification of a human in the loop or identification via the turing test, available electronically, `http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.ps`
10. Oechslin, P.: Making a faster cryptanalytic time-memory trade-off. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 617–630. Springer, Heidelberg (2003)
11. Oliveira, C.J.S., de Albuquerque Araújo, A.: Classifying images collected on the world wide web. In: Proc. of the 15th Brazilian Symposium on Computer Graphics and Image Processing, SIBGRAPI 2002 (2002)
12. von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: Captcha: Using hard AI problems for security. In: Proc. of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2003) (2003)
13. von Ahn, L., Dabbish, L.: Labeling images with a computer game. In: Proc. of the 22th Conference on Human Factors in Computing Systems, CHI '04 (2004)
14. Yan, J., El Ahmad, A.S.: A low-cost attack on a microsoft captcha. In: Proc. of the 15th ACM Conference on Computer and Communications Security, CCS '08 (2008)
15. Jeff, Y., Ahmad Salah, E.A.: Usability of captchas or usability issues in captcha design. In: Proc. of the 4th Symposium on Usable Privacy and Security, SOUPS '08 (2008)
16. Jeff, Y., Ahmad Salah, E.A.: Captcha security: A case study. IEEE Security & Privacy 7(4), 22–28 (2009)