CAPTCHAs: The Good, the Bad, and the Ugly

Paul Baecher Marc Fischlin Lior Gordon Robert Langenberg
Michael Lützow Dominique Schröder

Darmstadt University of Technology, Germany
www.minicrypt.de

Abstract. A CAPTCHA is a program that generates challenges that are easy to solve for humans but difficult to solve for computers. The most common CAPTCHAs today are text-based ones where a short word is embedded in a cluttered image. In this paper, we survey the state-of-the-art of currently deployed CAPTCHAs, especially of some popular German sites. Surprisingly, despite their importance and the large-scale deployment, most of the CAPTCHAs like the ones of the "Umweltprämie", the Bundesfinanzagentur, and the Sparda-Bank are rather weak. Our results show that these CAPTCHAs are subject to automated attacks solving up to 80% of the puzzles. Furthermore, we suggest design criteria for "good" CAPTCHAs and for the system using them. In light of this we revisit the popular reCAPTCHA system and latest developments about its security. Finally, we discuss some alternative approaches for CAPTCHAs.

1 Introduction

A CAPTCHA is an automated challenge and response program to distinguish humans from computers. The term CAPTCHA was coined by van Ahn et al. [vAMM$^+$08] as an acronym for "Completely Automated Public Turing test to tell Computers and Humans Apart". Roughly, CAPTCHAs are small puzzles which a human being should be able to solve easily, whereas an automated attack is infeasible.

State-of-the-Art. CHAPTCHAs are paramount in today's Internet: almost every user encounters a CAPTCHA in his or her daily usage of the Internet. CAPTCHAs are used to protect websites like googlemail or wordpress, to prevent spam in blogs, and to protect e-mail addresses. Despite their importance there are still a lot of seemingly bad CAPTCHAs out there. We highlight this by revisiting three popular German examples.

In March 2009 the stimulus program "Umweltprämie" offered car owners up to 2,500 EUR for a new car, scrapping their old one. Users could register online and, since the bonus was given on a first-come first-serve basis, the immense amount of accesses repeatedly caused unavailability problems of the hosting site (`www.ump.bafa.de`). About 2 Million car owners eventually received a bonus; we are not aware of the additional number of unsuccessful applications. As part of completing the online application form users had

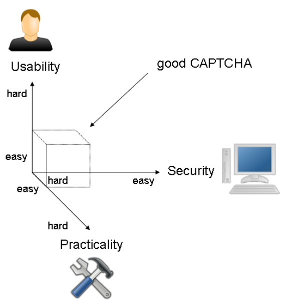


Figure 1: Requested Properties of CAPTCHAs.

to solve a CAPTCHA, seemingly to prohibit automated attempts to secure the bonus. The website is no longer operational since the program has ended.

In a similar vein, the Bundesfinanzagentur, specifically the Bundeswertpapierverwaltung (German Federal Securities Administration), supports the sale and custody of Governmental bonds, treasury financing papers and federal notes. For private investors it is free of charge, and currently used by about 450,000 private investors, with the agency's goal to increase this number to 1 Million in 2013 (according to their homepage). Around June 2009 the agency has augmented the log-in process for online banking by a CAPTCHA to "protect unauthorized persons to log into the Internet banking automatically and anonymously" (translated from German).

The Sparda-Bank, although divided into regional groups like Sparda-Bank Hessen or Sparda-Bank Berlin, uses a central log-in system for the online banking. Every user has to solve a CAPTCHA to log in, preventing unauthorized access since "reading the code by computer programs is very hard" (translated from German).

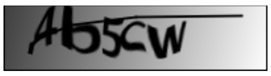
Here we show that none of the above CAPTCHAs achieves its intended goal; they are easily solvable by automated attacks, with success rates of $70\% - 87\%$. It is also worth noting that the governmental sites do not offer barrier-free alternatives for users with visual impairment (note that, while citizens with impaired vision may not drive a car, they may still own one and would have been eligible to sign up for the Umweltprämie).

Usability vs. Security vs. Practicality. Developing a CAPTCHA is always a trade-off between different goals. We identify three orthogonal requirements (see Figure 1):

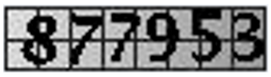
Usability refers to the difficulty of solving the CAPTCHA for a human, as well as the time that a human actually needs to find the solution.

Security on the other hand, gives a rough guide how difficult it is to find a solution for the computer.

Practicality refers to the effort to realize the CAPTCHAs in practice, e.g., if it requires only a standard web browser or can be used easily on a smart phone. Practicality could also refer to the acceptance on the users' side.



(a)



(b)

(c)

Figure 2: Three insecure text-CAPTACHAs. Figure 2a has been used by the German government for the “Umweltprämie”. The second CAPTCHA is part of the Sparda Bank, and the third one is implemented on the BFA website.

Ideally, a CAPTCHA should be highly usable by humans, provide strong security against automated attacks, and is easy to realize. As an example how practicality affects other dimensions consider an image-based CAPTCHA where users should interpret the semantics of the image. If, for ease of implementation, the CAPTCHA uses a public database with many freely available images then automated attacks may take advantage by trying to find the image in the database.

From a practicality point of view text-based CAPTCHAs, where a word or a sequence of digits or characters is embedded into a slightly distorted image, are easy to implement and are thus very popular. The main question regarding such text CAPTCHAs is the trade-off between usability and security. An in-depth discussion focused on usability of CAPTCHA designs can be found in [YA08].

When is a CAPTCHA broken? The question if a CAPTCHA is secure or not depends heavily on the application. For example, protecting spammers from posting advertisement in a blog may not be as dangerous as a bot that tries to hack an email account. As a rule of thumb, the developers of reCAPTCHA [vAMM+08] ask that computers can solve at most 5% of the generated puzzles, or else the CAPTCHA system should be considered broken. Note that this explicitly rules out attacks in which the CAPTCHAs are solved by exploiting cheap human labor, or via so-called “porn-relay” attacks in which CAPTCHAs are forwarded to other sites and then solved by humans trying to access these sites.

2 Bad Text-CAPTCHAs Do not Die Out

In this section, we discuss several in-use text-CAPTCHAs that are completely insecure. Figure 2 depicts the text-CAPTCHAs from the German government, the Sparda Bank, and the BFA. These CAPTCHAs have in common that they are built on a fixed-length sequence of black characters (or, the characters at least have the darkest color). The background consists of a color gradient from white to gray. The CAPTCHA of the “Umweltprämie” has in addition a black line that always ends in the right upper part of the picture and the Sparda Bank CAPTCHA uses a grid in the background.

2.1 Preparing the CAPTCHAs, or How to Segment the Characters

Since all discussed CAPTCHAs follow roughly the same idea, we describe a general method to break them all using only one program. We first remove the background. This is possible because the characters always have the darkest color. Thus, we parse the image and set the darkest gray value as a threshold, such that all lighter shades of gray values below this threshold are set to white.

Since the CAPTCHA of the "Umweltprämie" has a black line through the picture, we have to remove it. The main observation is that line starts in the right upper part of the picture. To remove this line we fist apply a line detection algorithm starting at the first upper black pixel and we then set all detected pixels to white. Likewise, the grid in the background of the Sparda-CAPTCHA can be removed easily because it is static and always appears at a fixed position (some care must be taken in order to remove genuine parts of the grid only).

In the last step, we transform each picture into one where each character has exactly 32x32 pixels. To do so, we start with a vertical line at the left part of the picture and move it towards the right part until we "hit" the first black pixel. This pixel must correspond to the first character. Afterwards, we use an algorithm that finds the shortest path from the bottom to the top (which is ideally a straight line). See Figure 3 for an example of the algorithm. Note that this partition algorithm is applicable to these CAPTCHAs because the characters are not connected. The algorithm then returns all separate characters.

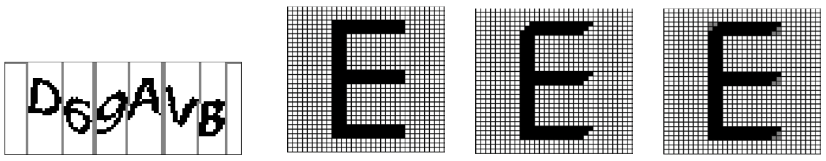
2.2 The K-Means Algorithm, or How to Break the CAPTCHAs

To break the CAPTCHAs of the Finanzagentur and the Umweltprämie with a single algorithm, we apply the *k-Means* algorithm [Mac67]. While it is in principle also possible to employ this algorithm also against the Sparda-Bank CAPTCHA, a simple OCR job with Tesseract after the preprocessing stage is sufficient in that case. For the other two CAPTCHAs the k-means algorithm yields stronger results with only a little more effort.

The k-Means algorithm has been developed for digital image processing and it is a so called clustering algorithm. Clustering means that this algorithm compares the given image to a database that contains several partitions and it returns the partition to which the image fits best. We illustrate the main idea of this algorithm for the following example.

The first part of the algorithm is a learning phase where it gets as input a set of images. The algorithm then computes on this basis its partitions. Figure 4 depicts the learning phase of the k-Means algorithm. The images 4a and 4b are two training images that the algorithm get as input. The k-Means algorithm then computes the difference between both pictures and it treats this difference as a tolerance. This is shown in Figure 4c. The black parts of the letter "E" are fixed wheres the gray parts may be contained (nor not) in the challenge picture. This way the algorithm also handles italic letters of different font types.

After having finished the learning phase, the k-Means algorithm is applicable to the challenge CAPTCHA. Whenever it gets as input a CAPTCHA, then it simply compares char-



(a) (b) (c) (d)

Figure 3: Partition of the picture and learning phase of the k-Means algorithm.

acter per character with its partitions (via the common Euclidean norm) and outputs the partition that fits best.

Results. The following results for Finanzagentur and Umweltprämie are based on a semi-automatic learning phase. Semi-automatic means that whenever an a certain letter matches to a wrong cluster, then we create a new cluster with this image. The advantage is that we get more partitions such that different fonts match to better to certain partitions. The results are based on a test set of 100 CAPTCHAs each (which are distinct from the CAPTCHAs in the learning phase).

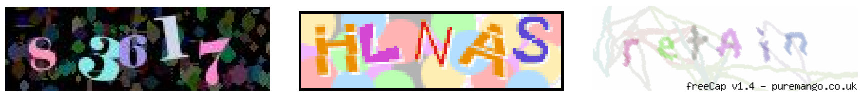
For the Sparda Bank CAPTCHA 330 challenges are preprocessed to remove the background grid and then simply fed to the Tesseract OCR software.

Finanzagentur	Sparda Bank	Umweltprämie
70 %	87 %	68 %

Note that the sample size (especially with 100 CAPTCHAs) appears to be rather small, and that a "lucky" set of test samples may not allow to extrapolate to the average success probability. However, since the success rate is quite high we can use the Chernoff bound to conclude that these CAPTCHAs should be considered broken, in the sense that one can solve significantly more than 5% of the puzzles by an automated attack (after an initial training phase which is inherent to many OCR or k-Means approaches). To be more precise, the Chernoff bound says that, given the statistics above that one can solve at least 70% of the random test sample of 100 CAPTCHAs, the probability that an individual CAPTCHA can be solved with more than 50% success rate, is more than $99,5\%$.

3 Best Practice for Designing Text CAPTCHAs

Given the history of weak CAPTCHA systems and their continuing use on popular sites, we aim to provide a set of recommendations on how to build secure CAPTCHAs and to avoid common pitfalls in this section. In the first subsection, we discuss high-level design issues that apply to the actual challenges. The second subsection discusses implementation questions which are easily overlooked but equally important to the security of the system.



`clipfish.de` `n24.de` freeCap 1.4

Figure 4: Careless of use of color that makes segmentation and extraction of the characters trivial.

3.1 Design Considerations

When designing a text-based CAPTCHA, it may be tempting to use dictionary words as challenges. While this provides superior usability since the user might be familiar with the word and thus be able to read it under extreme circumstances, it comes with several disadvantages. The most obvious aspect here is that the attacker can easily verify if a given attempt to pass the test successfully offline. Consequently, she is able to submit only those solutions which are likely to be considered valid by the challenger. This increases the difficulty to detect automated clients even further, since hopeless responses can be detected and discarded offline. Another drawback of dictionary words is the comparatively limited challenge space. This sets the stage for a class of attacks where a database containing every possible challenge can be precomputed and the best match for a given challenge selected. Finally, the usability advantage applies only if the testee is familiar with the language or even with the character set in question.

The usability penalty when using random character strings can be easily offset by allowing a small error in the given response. An edit distance metric that allows one bad character for example is suitable in this case.

Careless usage of different colors is arguably the most effective way to diminish the security of any CAPTCHA. The fundamental problem is that any change in color, slight or drastic, constitutes immediately a hint on how to segment the image. For example, rendering a string of overlapping characters where each character uses a distinct color is a trivial segmentation task. One may assume that this is well-understood, but instances of this design have been found in the wild as Figure 5 demonstrates.

Adding background clutter is another popular method that is supposed to complicate character recognition. There are, however, numerous instances where this kind of noise is entirely irrelevant for an adversary, including the examples in Section 2. This is somewhat related to the color discussion above: In the extreme case, characters are typeset in one color and some seemingly complex background uses any other color. A simple binarization operation is sufficient to separate background from characters in this case. Well-designed background clutter on the other hand is highly random and uses character-like shapes that share the same color and the same line width with the actual characters. Unfortunately, this is exactly what degrades usability and is one possible explanation why many strong CAPTCHAs use a plain backgrounds.

In general, one can differentiate between character segmentation and recognition. The former is the task of correctly decomposing a string of overlapping characters into individual characters while the latter deals with the recognition of a given (segmented) character. It is

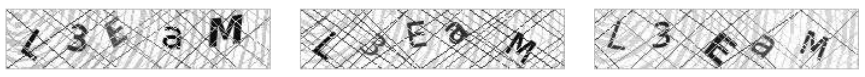


Figure 5: One challenge, three versions; allowing for independent analysis of each letter. Used on `digg.com`.

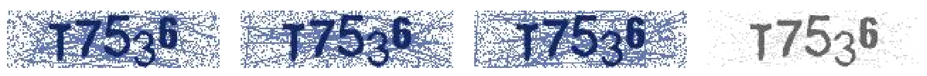


Figure 6: CAPTCHA used on `quoka.de`; three versions of the same challenge (a–c) with different backgrounds. A pixel-wise multiplication eliminates background noise almost entirely (d).

important to understand that segmentation is relatively hard for machines but recognition is easy given a correct segmentation. This has been studied extensively in [CLSC05]. A CAPTCHA designer should therefore rely on both the difficulty of both segmentation and recognition rather than recognition alone.

A very elegant way to generate new challenges automatically is to take advantage of the human user. This concept has been used by reCAPTCHA [vAMM+08] and [GKB09]. The basic idea is to present one challenge whose legitimate response is unknown together with $k-1$ known challenges at once. If the user is able to answer the $k-1$ challenges correctly, it is likely that she also provided the correct response to the unknown challenge. If a sufficient number of users agree on the response, this pair can be stored as known. It is, however, important that known and unknown challenges are indistinguishable to the user. Otherwise, the system can be exploited easily by a large number of users that always provide the same bogus response to all unknown challenges. Eventually, the database is polluted and accepts the bogus solutions when newly "learned" challenges are generated. Note that this kind of attack is still possible for small values of k even if the type of the challenges is indistinguishable.

3.2 Realization Advice

As an attacker only needs to target the weakest component of any CAPTCHA system, it is important to pay close attention to the surrounding implementation that is not strictly part of the generated challenge.

Prevent replay attacks. If allowed, this is perhaps the most effective type of attack. The system needs to keep track either of challenges that are yet to be answered or challenges that have been answered already (both correct or incorrect). Otherwise, a human attacker may solve one single instance and reuse the answer multiple times in a subsequent automated attack. This also implies that some kind of stateful server is necessary – an unattractive requirement in high-load environments, but vital nevertheless.

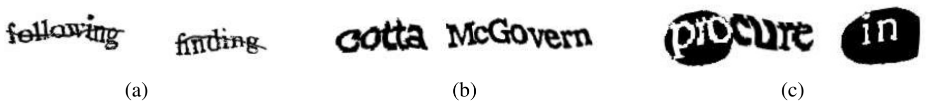


Figure 7: Evolution of reCAPTCHA.

One version per challenge. One specific challenge should be presented in one *version* only, i.e., the according image should be created in a deterministic fashion. Generating multiple versions of the same challenge by applying different transformations, as in Figure 6, gives a great advantage to the attacker. By requesting the same challenge multiple times, unrecognizable characters in one version can still be recognized in another version of the same challenge. In fact, since the same characters are used, the problem is reduced to correctly recognizing one single character independently from a set of samples instead of an entire string at once. This increases the success probability for the attacker exponentially in the number of requested versions.

A particular weak variant of this flaw arises when only background clutter varies over different versions. Figure 7 is one such example, where a pixel-wise multiplication operation essentially removes the background noise completely (Figure 7d). This CAPTCHA is used on "Quoka.de" as of March 2010, a popular online marketplace that claims[1] to have had over 78 million page impressions in July 2009. Note that this CAPTCHA can be broken by even less sophisticated means; it only serves as an example here.

Impose timeouts. A challenge should be valid only for a limited amount of time in order to restrict the adversary's attack frame. Note that the timeout should not be chosen too short either, since the user may be required to fill out some form first[2]. It is, however, unreasonable to accept challenges that have been given out several hours or days before submission.

Adapt to attackers. If distinct "users" can be identified, it makes sense to increase the difficulty of the test after several unsuccessful attempts have been made by the same user to solve it. This can be done by stronger distortions and/or more background clutter. For many public services it is impossible to identify distinct users, though. Using the testee's IP address is a popular choice in practice but generates many false positives within institutions that share one public address. Hence this option should be used with great care.

4 reCAPTCHA

reCAPTCHA (`recaptcha.net`) is perhaps the most successful and robust CAPTCHA systems of the last few years [vAMM+08]. The crucial property that separates it from regular text-based CAPTCHAs is the method of generating challenges—they are a by-product of digitizing huge amounts of text. Whenever OCR software fails to recognize one word properly, it is considered hard to digitize and is added to the reCAPTCHA system, after undergoing even further distortions. This immediately ensures that the word in question is not trivially readable by machines.

Since the correct responses to these challenges are not known, the system learns from its users as described in Section 3.2 by presenting one known and one unknown challenge. The system in its entirety also constitutes a remarkable win-win situation: We either get a secure CAPTCHA implementation or, if an attack exists, a significant advance in the accuracy of OCR systems. This has lead to enormous success: reCAPTCHA is used on many of the most popular sites on the Internet, including Facebook, Twitter and Craigslist. Over 200 million challenges are solved each day by its users according to reCAPTCHA.

Conceptually reCAPTCHA is also unique in the sense that it is a centralized service rather than downloadable software. One important implication is that there is no definite version of reCAPTCHA; it is a highly dynamic system that changes both slightly and drastically (cf. Figure 8) over time. Occasional claims that reCAPTCHA has been broken are therefore to be taken with a grain of salt, since this can possibly only refer to one point in time. It is consequently often difficult to verify such claims without a large test set of challenges from the relevant period of time. Even then, prompt changes by reCAPTCHA may have rendered the attack irrelevant in the meantime. Nevertheless, specific versions of reCATPCHA are known to be prone to attacks from which we can learn. We discuss these in the next section.

Attacks on reCAPTCHA. In its original version, as depicted in Figure 8a, reCAPTCHA added a horizontal line to each challenge word as well as per-word distortion. In [Wil09] the author describes how challenges of this version can be broken by first applying a set of morphological image processing operations, specifically erosion (remove pixels on the border of objects) and dilation (add pixels on the border). Standard off-the-shelf OCR software, namely Ocropus, then processes the resulting images. The exact parameters for the morphological operations are learned from a training set of 200 labeled challenges. This approach is claimed to yield a 5% success rate for a test set of 200 challenges.

Likewise, a more recent version (Figure 8b) of reCAPTCHA can be solved automatically in 5% of the time by simply passing it to the Tesseract OCR software, without any preprocessing, according to the aforementioned author. Considering that the previous version was completely resistant against this trivial attack, there is some indication that reCAPTCHA in fact weakened their system between these two evolution steps.

The latest version of reCAPTCHA, as of April 2010, inverts the colors within a randomly

[1] `http://www.quoka-gmbh.de/resource/pdfs/Quoka_Mediaunterlagen_Online.pdf`

[2] Ideally, the CAPTCHA is presented separately before or after the actual form completion.

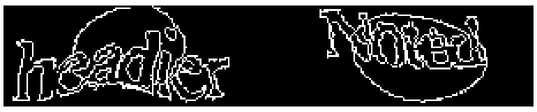


Figure 8: One of reCAPTCHA's current challenges, edge detection and binarization operations applied.

chosen, elliptic area that overlaps with the challenge words; Figure 8c depicts an instance of this method. This change seems to be reCAPTCHAs response to the publication of Wilkins [Wil09] in December 2009, since it has been introduced shortly after. This modification is remarkable in two aspects.

First, the statistics in [Wil09] appear to be quite fragile. The (basic) attack reported in [Wil09] guarantees about 5% on a sample space of only 200 challenges. This sample space is rather small compared to the low success probability. Assuming independent distributions with a success probability of 5% each, and approximating the (averaged) sum of successes for the 200 challenges by a normal distribution, we consider a confidence interval of 80%, i.e., the actual mean value really falls into the interval with at least 80% (leaving 20% for possibly lower or higher values). Then this confidence interval still ranges from 3% to 7%. One may quite reliably observe 5% success even though the actual success rate is much lower. As an example, for an actual success rate of 4% the 80% confidence interval would be approximately $[2.3\%, 5.7\%]$. The problem does not occur for our examples in the previous section as the success rate there is sufficiently high for the number of samples.

We also note that the optimistic success rate of up to 17.5% mentioned in [Wil09] is only valid under the assumption that, of the 25% of the additional cases where only one of the two words is recognized, half of these solutions are for the unknown word. It is unclear if this assumption is justified. In the worst case, the overall success rate could still be as low as 5% (or even lower, as discusses above).

In summary, it must be observed, though, that the actual recognition rate could in fact be higher than the estimates. It is only safe to say that the statistics, as given, do not provide a reliable lower bound.

The second issue about reCAPTCHA's evolution is that it is rather unclear to us if the aforementioned modification significantly improves the security. Although we are not aware of any attacks at this stage, there are a few properties that make this type of distortion possibly vulnerable:

1. There is little variance in the general shape of the object. It is always well-rounded and can be approximated roughly with an ellipse.

2. The edges of the distortion object are very "clean", it is therefore easy to detect these with standard algorithms. As shown in Figure 9, these edges have low curvature compared to the curves of characters and they often extend to areas where they stand out (top left and bottom right here). This makes for an easy start point for a line trace algorithm that also takes the shape of the object, an ellipse, into account.

3. The shape of the object forms a relatively large blob of black pixels in the verbatim challenge, making it easy to locate. For example, a vertical pixel projection reveals two maxima that correspond directly to the objects.

Once this distortion or large parts of it can be removed, the same attacks that apply to the previous version should work. With these considerations in mind, we expect that it is only a matter of time until reCAPTCHA is forced to adjust their system again.

5 Other CAPTCHA Types

So far we dealt exclusively with text-based CAPTCHAs. While this type of CAPTCHA is certainly the most common one, other approaches have been proposed and implemented, most predominantly image-based systems. There are many different reasons why these are interesting alternatives. We first discuss a few of these reasons and then give some concrete examples of implementations.

First, OCR systems constantly improve and have reached a level that demands very strong distortions. These, in turn, affect the usability of course and make it harder for humans to solve the challenge. Another aspect is the increasing number of small mobile devices that are capable of using standard web services. Often these devices do not have a hardware-based keyboard and the user is forced to use an alternative—possibly cumbersome—input method to solve a text-based CAPTCHA. Additionally, many web-based applications rely strongly on a pointing device for navigation where the user's hand spends most of the time on this device. It can be annoying if such a "click-based" navigation flow is suddenly interrupted by a text-based CAPTCHA challenge. Finally, it seems that CAPTCHAs which resemble some kind of small game challenge are more likely accepted than writing down a string of characters.

A typical issue of CAPTCHA systems that are not based on text input is a small response space. For example, the large class of binary decisions is of limited use only. Since the attacker can always be successful with probability $\frac{1}{2}$ by guessing, many of such challenges have to be combined in order to achieve a reasonably sized response space: A guessing probability of $0.1\% \approx 2^{-10}$ requires 10 challenges to be solved at once. Likewise—in the case of image-based systems—whenever the response is a position (more precisely a tolerance area) within the challenge image, comparatively large images are needed to reach 0.1%. A circular response area of 16 pixels diameter for instance would require approximately a 450×450 image, which is prohibitively large for mobile displays.

The systems we present here are a selected set of research projects mostly. We find these particularly interesting, despite their availability or readiness for productive use. Some of them are, in fact, already broken but they demonstrate novel approaches to this topic.

Image Rotation. This novel CAPTCHA system, presented in [GKB09], asks the user to rotate randomly oriented images to their natural upright position. While this task is easy to automate for a certain class of images, such as photographs of faces, type or clear horizon lines, it is considered hard for a wide range of images in general. Experimental results indicate that this system works well if the images do have a natural upright orientation. The actual challenge here is to filter the other, infeasible, images for which the authors suggest an automated method. However, unsuitable images may still slip through this preprocessing stage and need to be filtered out in the live system by evaluating the variance of the chosen user rotations.

Asirra. Asirra [EDHS07] is an image-based CAPTCHA developed at Microsoft Research. It relies on the presumed difficulty to automatically tell pictures of dogs and cats apart. The system presents a set of 12 pictures at once and asks the user to select those which depict a cat. Surprisingly, an automated attack [Gol08] exists which is able to distinguish these two types of photos very reliably with a probability of 82.7% and is thus able to solve the Asirra challenge in 12.2% of the time. This result highlights how difficult it is to design a strong CAPTCHA, even if the challenge it may appear hard at first.

6 Conclusions

Designing good CAPTCHAs is a tedious business. Text CAPTCHAs achieve a high level of practicality, but very often fall short of providing a good balance between usability and security. Currently, the best choice seems to be reCAPTCHA: it is easy to incorperate, achieves appropriate security and usability levels, and because of the centralized structure behind reCAPTCHA it allows fast migration in response to emerging threats.

Acknowledgments. We thank the anonymous reviewers for valuable comments. Paul Baecher, Marc Fischlin and Dominique Schröder are supported by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation (DFG). This work was also supported by CASED (`www.cased.de`).

References

[CLSC05] Kumar Chellapilla, Kevin Larson, Patrice Y. Simard, and Mary Czerwinski. Building Segmentation Based Human-Friendly Human Interaction Proofs (HIPs). In *HIP*, volume 3517 of *Lecture Notes in Computer Science*, pages 1–26. Springer, 2005.

[EDHS07] Jeremy Elson, John R. Douceur, Jon Howell, and Jared Saul. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. In *ACM Conference on Computer and Communications Security*, pages 366–374. ACM, 2007.

[GKB09] Rich Gossweiler, Maryam Kamvar, and Shumeet Baluja. What's up CAPTCHA?: a CAPTCHA based on image orientation. In *WWW*, pages 841–850. ACM, 2009.

[Gol08] Philippe Golle. Machine learning attacks against the Asirra CAPTCHA. In *ACM Conference on Computer and Communications Security*, pages 535–542. ACM, 2008.

[Mac67] J. B. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[vAMM$^+$08] Luis von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. reCAPTCHA: Human-Based Character Recognition via Web Security Measures. *Science*, 321(5895):1465–1468, 2008.

[Wil09] Jonathan Wilkins. Strong CAPTCHA Guidelines v1.2. 2009.

[YA08] Jeff Yan and Ahmad Salah El Ahmad. Usability of CAPTCHAs or usability issues in CAPTCHA design. In *SOUPS*, ACM International Conference Proceeding Series, pages 44–52. ACM, 2008.