Journal of Information Security and Applications 63 (2021) 103018

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

BASECASS: A methodology for CAPTCHAs security assurance

Carlos Javier Hernández-Castro[1], David F. Barrero[1], María D. R-Moreno *,[1]

Universidad de Alcalá, Escuela Politécnica Superior, ISG, Alcalá de Henares, Spain

ARTICLE INFO

Keywords:
CAPTCHA
Methodology
Machine Learning
Statistical analysis
Security assurance

ABSTRACT

Today, much of the interaction between clients and providers has moved to the Internet. Some tricksters have also learned to benefit from this new situation. New improved cons, tricks and deceptions can be found on-line. Many of these deceptions are only profitable if they are done at a large scale. In order to achieve these large numbers of interactions, these attacks require automation.

CAPTCHAs/HIPs are a relatively new security mechanism against automated attacks. They try to detect when the other end of the interaction is a human or a computer program (a *bot*). However, CAPTCHA/HIP design is still in its initial conception as the stream of successful attacks highlight it.

This paper focuses on the design of CAPTCHAs and if there is a way in which to assess a basic level of security for new CAPTCHA designs. To do so, we first review main attacks to different types of CAPTCHAs and then, we describe BASECASS, a methodology that can help in avoiding some of these design pitfalls. The application of the methodology is exemplified in three attacks to CAPTCHAs and how following the methodology designers could have avoided them.

1. Introduction

The Internet has spread to every realms of life. New generations spend more time on-line both socializing and working. People are getting used to the advantages of being constantly connected. Using this ample base of both services and people, attackers have found ways to run exploits that provide an infinitesimal reward, but could generate substantial revenue by increasing the number of times they are run. The fundamental way of protection from these attacks has been to try to detect if at the other end of the communication there is a human person or a computer program.

There are many proposals for ways of remotely detecting humans. Most of them fall into the category of asking the *human* to perform a task that is considered hard for computers but not too demanding for humans. These tests are known as *Human Interaction Proofs* (HIP) or *Completely Automated Public Turing test to tell Computers and Humans Apart* (CAPTCHA) and they provide this protection mechanism.

Sometimes the name of HIPs and CAPTCHAS are misleading because CAPTCHAs, as they were defined by Naor [1] and Ahn et al. [2], are just a specific version of this protection mechanisms. For a HIP to be a CAPTCHA, it has to meet certain requirements, including being based on a AI-hard problem, using a public algorithm, etc.

Since the first CAPTCHA used in Altavista in 1997, there have been numerous, very varied CAPTCHA designs proposed, implemented, and cracked. Even though it might look like an easy problem to the inexperienced, CAPTCHA design is not a straightforward problem to solve. Summarizing, we could identify the following difficulties related to the design of CAPTCHAs:

- CAPTCHAs are typically used to protect resources that for the customer are not of a very high value (e.g. adding comments to a story in the news), or to which there are other alternatives for the customer (for services such as web-mail). This competition means the CAPTCHA needs not to be felt as a burden by the user.
- For the same reason as above, a CAPTCHA must not require a big commitment for its completion, even if the experience is very playful and positive for the user. Completing a CAPTCHA is never the reason, but a means to an end.
- CAPTCHAs should present alternatives for impaired users that offer the same level of security. This is not straightforward, as typically a CAPTCHA would use some human ability that is linked to a sense of perception (visual, auditory, etc.) thus not being valid for users with disabilities in that sense.
- The number of attacks per second against a CAPTCHA could be augmented automatically: it is just a matter of resources. Thus, a very small success rate could imply that for practical purposes, a CAPTCHA is broken. This is the case as soon as the Return of Investment (ROI) for the attacker is positive. Thus, in order to

* Corresponding author.
E-mail addresses: david.fernandezb@uah.es (D.F. Barrero), malola.rmoreno@uah.es (M.D. R-Moreno).
[1] All authors equally contributed to the work.

https://doi.org/10.1016/j.jisa.2021.103018

Available online 26 October 2021

protect the most interesting resources, we need *AI-hard* challenges with extremely constant hardness throughout their domain.

- For some attackers, it might be profitable to hire low-wage human workers (what is typically called a *farm*) to solve a particular CAPTCHA challenge and then proceed to do whatever they wish. This would constitute a semi-automated attack. It is good if a CAPTCHA has some way of preventing this from happening. Some CAPTCHA designers consider this requirement, yet the rest do not try to counter it [3,4].

To date, all proposals for CAPTCHAs that have been analysed have been found not secure, typically within a short span of a few months from their proposal time or from when they were put into production. Most of the attacks found against CAPTCHAs could be considered to be side-channel attacks. They do not try to solve the underlying problem on which the CAPTCHA designer has created her system, nor they try to advance the state-of-the-art in Machine Learning (ML). Instead, they find weaknesses in the particular design of the CAPTCHA and ways to use them to gather enough information as to bypass the challenge a sufficient number of times. The frequency with which this type of attack is successful conveys the message that it is quite difficult to translate an *AI-hard* problem into a secure CAPTCHA.

There have been a few proposals for design guidelines for CAPTCHAs. Typically, the result of a security analysis of one or more CAPTCHAs, and thus with limited scope and usability [5–8]. Nowadays, is not unusual that a new CAPTCHA design is put into production without performing a sound security assessment nor conducting external IT Security tests. These CAPTCHAs are implementations just based on an idea though to be hard enough by its designers.

In this work we want to show whether there are some basic tests that we could run as to ascertain a basic level of security for a new CAPTCHA design, and that possibly could be automatic or semi-automatic. Our goal is to increase the security of CAPTCHA designs. For that we propose a methodology called BASECASS that could help in avoiding some of the design pitfalls. Although BASECASS does not provide guidance for possible mitigations of the vulnerabilities found, it could help assessing that a new CAPTCHA proposal meets a basic level of security against primary and statistical side-channel attacks.

The paper is structured as follows. Next section summarizes the main attacks against CAPTCHAs. Then, we present a methodology for ascertain a basic level of security. After, some state-of-the art CAPTCHAS are tested against our methodology. Finally, conclusions and future work are outlined.

2. Attacks against CAPTCHAs

The different attacks that have been successfully able to break CAPTCHAs/HIPs have strongly guided their evolution. Not all attacks have been public, and is not unusual to see that a CAPTCHA design evolves without presenting a reason why. In general, the evolution of CAPTCHAs design has followed a path to avoid known weaknesses, so it is reasonable to assume that the main attacks, those more fundamental to their design, have seen the light.

In the next subsections we revise some of the main attacks for different CAPTCHAs. We can further divide them into different design categories both based on their transport media (text, text images, audio, images, …) and in the particular problem they are based on (OCR, classification, understanding, …).

2.1. Attacks to text based CAPTCHAs

Text-based CAPTCHAs pertain to two main categories: those based on the problem of text recognition from an image (OCR), and those using text as a mean to ask a question. OCR CAPTCHAs will be explained in this subsection. Text semantic CAPTCHAs in Section 2.2.

One of the earliest examples of CAPTCHAs was an algorithm that randomly generated an image of printed text with some distortions so that OCR programs could not read it, requesting the *human* user to input such text [9]. The distortions included random typefaces, rotation and scaling, as well as the optional addition of background noise. Characters were chosen at random, not from a dictionary. It is important to note that by that time or shortly after, there were known algorithms able to recognize patterns even after being rotated and scaled [10].

By those days, two researchers were also using their already developed framework for object detection to successfully break both the EZ-Gimpy CAPTCHA in use at Yahoo! and the *hard* Gimpy CAPTCHA [11]. Their work is notable because of two aspects. It was the first research paper that was peer-reviewed and published that focused on finding weaknesses on a CAPTCHA and breaking it. The second important aspect is that it showed some weaknesses of the *hard* Gimpy CAPTCHA. These weaknesses and their exploit were not clearly applicable to any OCR thus not improving the state-of-the-art. Contrary, these weaknesses were related only to the way in which these particular CAPTCHA obfuscated the characters.

This started the *arms race* between CAPTCHA developers and breakers. An example is the work of Yan [5] that was able to successfully attack Captchaservice.org using pixel-counting of contiguous regions for character detection as well as vertical pixel counting for segmentation attaining a 36% success rate. Adding a dictionary look-up assisted by a total pixel sum matching, as well as a dictionary pruning for characters with similar pixel count, they attained a 94% success rate, increased up to 99% with additional heuristics.

In 2008, Yan and El Ahmad published an attack on the CAPTCHA deployed by Microsoft in services like Hotmail, MSN and Windows Live [12]. They used a similar idea as before [5] but this time they also detect continuity of groups of pixels by flood-filling. This idea helps with those chunks of characters not correctly segmented by using the vertical pixel histogram that contains more than one character. They were able to break the Microsoft CAPTCHA on 92% of the occasions [12].

In 2010, El Ahmad and Yan were able to break the CAPTCHA of a popular file sharing (Megaupload.com) that used substantial overlapping to avoid segmentation. They did so identifying and merging character components [13].

An important break work was the use of ML to attack both character segmentation and recognition simultaneously, scoring hundreds of different possible segmentation decisions [14].

In 2016, Gao et al. [15] published another generic attack for OCR CAPTCHA based on Log-Gabor filters able to break the characters into their different components. When they tried their attack on CAPTCHA deployed by the top 20 most popular websites, they found that their attack successfully broke all of them, with success rates varying from 5% for Yahoo! up to 77.2% for reCAPTCHA.

During the evolution of the research in OCR CAPTCHAs, it was seen that distortions to characters have their limits, especially when computers are better at recognizing single characters, and segmentation can be solved using Neural Networks (NN) and other methods. Thus, some researchers looked into how to still use characters, but using a different representation that could be made harder for machines. This is how the ideas of 3D OCR CAPTCHAs (i.e. Teabag CAPTCHA) and animated character CAPTCHAs (i.e. HelloCAPTCHA) started. However, several attacks have been successfully proposed in both cases [16,17].

2.2. Attacks to language/semantic CAPTCHAs

This type of CAPTCHAs asks users very simple questions as to detect which word is different from a list of words, or solve a very simple arithmetic problem.

We can mainly mention two CAPTCHAs, the TextCAPTCHA, a textual question generator that presents important design flaws that allow to easily reverse-engineer it. In particular, it is straightforward to detect which subtype of challenge it is using, and thus apply an ad-hoc solver to each case and break it [18]. And the Egglue CAPTCHA, that uses a proprietary algorithm, accessible through a web service, that creates two sentences that the user has to fill in with the correct verb. Its mechanisms remained as a black-box, with no information on how Egglue created and marked the challenges. However, it was seen that the algorithm it is using for marking a challenge was not strong. For example, it allowed using general verbs successfully even for sentences for which they did not make sense. Some verbs had a success rate over 90%, which is clearly not related to the distribution of appearances of verbs in English. This also implies that several sentences are considered correct with many different verbs and can be easily broken [18].

2.3. Attacks to image classification CAPTCHAs

Some people started to believe that OCR/text CAPTCHA were fundamentally limited. Thus, they looked for different alternatives that could lead to stronger CAPTCHAs that still had high usability. Many of these researchers focused on the more general problem of Computer Vision (CV). This was a natural election, given that CV was an AI field that had several unsolved problems at that time. Then, we can mainly consider three different types of image-based CAPTCHAs. First, those based on image classification, that is, the ones that classify an image into a single class that describes the main content of the image. Second, the CAPTCHAs that use some image-related CV problem as their base. And finally, those based on face identification, recognition, or extracting information from faces.

In the first type, we can mention the HumanAuth CAPTCHA that requests to distinguish between pictures depicting either a natural object (a tree) or an artificial one (a watch). It was an Open Source project that was shipped with 69 pictures, very lightly obfuscated using a watermark. This obfuscation did not serve much, as each image had assigned a textual description for the visually impaired [19]. It was easily broken using some simple metrics from each image measured using the ENT pseudorandom number sequence test program from the Fourmilab and training an ML classifier on these metrics. This was possible even when the CAPTCHA was using the watermark and the attack did not take advantage of the textual description [20,21].

In the case of the ASIRRA CAPTCHA, the authors published an initial security assessment and provided a training set for anyone willing to try their ML algorithms on it. Golle [22] experimented with similar ML methods to the ones used by the creators of ASIRRA, using different features to train a SVM classifier. His SVM used a radial basis kernel. The most successful features where boolean colour presence (if a colour was or not present in a certain part of the image) and 5×5-pixel texture features, selected at random and filtered to be different enough. Golle was able to break ASIRRA with a success rate of 10.3% (82.7% accuracy for a single image).

Related to the second type of CAPTCHAs, we can mention the IMAGINATION CAPTCHA [23] that asks the user to click in or around the centre of any of the images that form a mosaic. This mosaic of images is created in a way that makes it difficult for known CV techniques to segment it. This proposal was broken by Yan et al. [24] using a clever algorithm to find candidates for image boundaries.

Another example is the proposal of Gossweiler et al. [25] from Google Research. Their CAPTCHA presents rotated images to the user. The user has to rotate them back to their original orientation. This proposal was never implemented at large scale by Google, so a proper analysis is pending.

Finally, in the gender recognition CAPTCHAs proposals we can mention several works [26–29]. However, the only one that was put into production is FunCAPTCHA [28]. FaceDCAPTCHA [30] and FR-CAPTCHA [31] are two CAPTCHAs based on human face recognition. FR-CAPTCHA asks the user to find matching pairs of human faces in an image. FaceDCAPTCHA presents images of both real and fake faces, distorted and partially occluded, and asks the user to select the images containing real faces. Both were broken by Gao et al. [32].

Facebook also studied the use of their face identification data as the base for a CAPTCHA. This CAPTCHA proposal was analysed by Kim et al. [33] finding possible attacks. It was later broken using well known classifiers kNN and SVC (with better results), but choose kNN as results were similar and it is computationally less expensive. Optionally, they performed a *social engineering* attack to reach "sensitive" data. They do so using fake Facebook profiles to befriend friends of the target. With the data collected through *social engineering*, they reach a 100% success rate [34].

2.4. Attacks to game-like CAPTCHAs

In the earlier part of the decade of the 2010s, several proposals appeared that tried to increase the usability of CAPTCHA by making them appear as simple games [28,35]. This technique was termed "gamification". The User Interface (UI) also improved to include techniques like drag & drop, more user friendly, especially when using mobile (tactile) devices. The underlying mechanisms for the CAPTCHA did not change abruptly, but the interaction with the user was improved.

One of the first production CAPTCHA to use gamification techniques is the one created by *Are You a Human*, the *PlayThru* CAPTCHA. It is composed of small drag & drop games. Mohamed et al. [36] were able to easily break this CAPTCHA by using simple heuristics to detect the background. Similarly they detect the foreground objects as well as learning from the objects by remembering them.

2.5. CAPTCHAs based on the understanding of video

Some ideas appeared that were purely based on video. These are different from the proposal based on *emergence* [37] and other proposals based on adding animation to OCR CAPTCHA [38]. We will call them *pure-video* CAPTCHA because they are based on extracting semantic information from the sequence of actions that the video portrays. Some of these proposals are the ones from Kluever et al. [39], Hernández-Castro et al. [40] or the similar "motion and interaction based CAPTCHA" [41]. They have not been implemented, so a proper security analysis is missing.

For any CAPTCHA based on video, there is the concern that the additional information that the video will provide will somehow make it easier to find clues in order to break them. No proper security analysis can be done until there is a public implementation.

2.6. Audio CAPTCHAs

Most CAPTCHA proposals have been based at least partially on the visual capacities of ordinary humans, but there is a number of people who have vision problems. Some CAPTCHAs that have been put into production have had to provide an alternative for visually impaired users. This is the way that most audio CAPTCHA proposals have appeared.

One of the most popular audio CAPTCHA was the Google audio CAPTCHA, presented in 2008. Each challenge consisted on a series of digits being spoken with background noise. Santamarta [42] showed that the Google audio CAPTCHA could be broken using very basic methods. In particular, it was possible to detect the characteristic wave and FFT of each digit spoken, and because they were played with higher volume than the background noise, it was possible to easily distinguish them.

Another successful attack on it was based on well-known ML algorithms, in particular using AdaBoost, SVM, and kNN for both letter and digit recognition [43]. They used a static window size, and train on well-known features for NLP, in particular, twelve MFCC and twelfth-order spectral and cepstral coefficients from PLP and RASTA-PLP. They were able to break Google Audio CAPTCHA with a 67% success rate, Digg with a 71% success rate and reCAPTCHA with a 45% success rate.

This has lead to an increase in their difficulty, adding noise and choosing audio cues that are difficult to understand. Even after improvements, the audio version of reCaptcha by Google was broken again using simple speech recognition [44] and later using the speech recognition API of Google [45]. It remains to be seen if a strong yet usable audio CAPTCHA could be created.

2.7. Attacks to “behavioural” CAPTCHAs

Even though many bot detection mechanisms are marketed as “no-CAPTCHA” alternatives, they are a mix of CAPTCHAs and algorithms to decide whether to display them or not as well as with what level of difficulty. These different proposals typically resort to some conventional CAPTCHA when they determine that there is not enough information. They sometimes call themselves *behavioural analysis*, which is a fancy term to refer to more or less typical mechanisms to automatically create blacklists of potential attackers and/or white-lists of low-risk users. These mechanisms associate a level of potential danger to each *different* client.

NuCaptcha uses a combination of what they call a “behaviour analysis system to monitor all interactions on the platform” to modify the difficulty of the CAPTCHA challenge, relying in an improved OCR/text CAPTCHA that incorporated moving characters. Note that this did not prevent the attacks by Bursztein et al. [46] and Xu et al. [47].

Another CAPTCHA in this category is the “No CAPTCHA reCAPTCHA” by Google broken within a week after its release [48]. It used extreme obfuscation code for their Java Script client and client–server communications. This reverse-engineering allowed to understand the local metrics that Google’s reCAPTCHA was using. Among them were the list of plug-ins installed in the browser, the user–agent string, screen resolution, execution time, time-zone, number of user actions inside the CAPTCHA iframe, the behaviour of some CSS rules and functions that are typically browser-specific, whether the browser renders canvas elements, etc. Other security flaws of “No CAPTCHA reCAPTCHA” were also soon pointed out by Homakov [49].

2.8. Attacks to CAPTCHAs based on empathy

The authors of the Civil Rights CAPTCHA (*CRC*) use the human ability to feel empathy to strengthen a typical OCR/text CAPTCHA. The *CRC* picks up a Civil Rights news from its database (DB) and then uses Securimage to create three images containing words depicting possible emotions related to the text. These images contain words describing feelings (for instance, “upset”, “happy” and “furious”). The user has to write down the correct one based on the emotions originated from the news headline presented to her. Thus, the *CRC* is based on the human ability to show empathy after being presented with a news excerpt, typically containing some news about Human Rights and/or Civil Rights around the world.

The Civil Rights CAPTCHA uses a traditional OCR CAPTCHA, to which there are known attacks, but it is further secured by the detection of empathy. There is currently no ML algorithm that tries to simulate empathy. There are ML approaches to understanding the human languages, but they focus on detecting the feelings and opinions of the writer through the use of adjectives and adverbs. They do not focus on the induced feeling on the reader. However, as Hernández-Castro et al. [50] shown, the combination of two CAPTCHAs is not always more secure than one of them alone, as the way the *CRC* uses *Securimage* lowers its security, and in turn allows us to break the *CRC*. As well, they show that general metrics, along as some other metrics slightly modified for the case, can give enough information about the challenges as to allow various ML algorithms to break the CAPTCHA a significant number of times.

2.9. Puzzle CAPTCHAs

A recent game-like CAPTCHA proposal are puzzle CAPTCHAs. In them, an image is divided into parts, of which at least one is missing. The user has to place the missing parts correctly to solve it. Other variants have the parts shuffled and the user has to reorder them. In any case, the user has to reconstruct the original image.

These proposals are fundamentally different, as there is no need to recognize and interpret the different elements. Also, the puzzle pieces are not differentiable elements by themselves, that is, a puzzle piece is not recognized by our visual system as a ball, a lamp or a door; it is nothing more than a puzzle piece. Thus object detection does not serve a purpose here. These proposals are also different from image classification CAPTCHA, as the only classification relevant here is if the image is correct (as the original) or one of the many incorrect possibilities, with the puzzle pieces wrongly placed.

There are many attacks on image classification CAPTCHA and other image-based CAPTCHA, but none on puzzle CAPTCHA before we did apply BASECASS to them. As explained, these pose a fundamentally different problem, in which we are not interested in interpreting the images, but on restoring it to its original state. Some of these puzzle CAPTCHA are Capy, KeyCAPTCHA, and Garb, all of them already broken by the same authors by using JPEG to measure the image’s continuity [51]. After Capy was broken and the creators were contacted with the attack info, they introduced “Capy Risk-Based Authentication”, an “authentication system which takes into account the profile of each user requesting access to the system to determine the (login) history”.

3. BASECASS: A framework for BAsic SEcurity CAPTCHA ASSessment

The idea of BASECASS is to apply a series of partially-customized steps to analyse a particular design trying to find some possible vulnerabilities. In that sense, it is related to a vulnerability assessment or a penetration test. A vulnerability assessment will typically look only for well-known vulnerabilities in a semi-automated or automated way. In a penetration test, the testers will additionally look for variations in these vulnerability types. The pen-testers will try to find variations of them, using their previous knowledge of the system, the security measures in place, and the typical vulnerability scenarios. Contrary to these tests, BASECASS focuses on fundamental properties of each CAPTCHA proposal, and the distribution of different values within the challenge and the label space. BASECASS cannot be applied to a web application or API, and it is fundamentally designed for CAPTCHA security only.

Our framework proposes an analysis that lies closer to a penetration test. In it, the tester will have to apply her knowledge of previous CAPTCHA side-channel attack techniques, but also propose the use of possibly known useful metrics, and possibly come up with new ones which are variants more suitably tailored to the particular CAPTCHA being analysed. The main difference between our framework and a typical penetration test lies in the particular steps we propose in it. In our case, these steps are tailored specifically for analysing CAPTCHA designs, and are generic, and thus applicable to most designs.

BASECASS can be divided in three main steps:

1. A black-box basic security analysis of the CAPTCHA.
2. An additional analysis based on Statistical Analysis (SA) and/or ML.
3. A parameter-related SA and/or ML analysis.

Depending on the CAPTCHA type, the third step might not be possible, as it will require further insight or access into the CAPTCHA design. If it is possible, it will typically provide more accurate information about the minimum security level of the CAPTCHA. We will use the same analysis tools in the steps two and three. Thus, we call each step

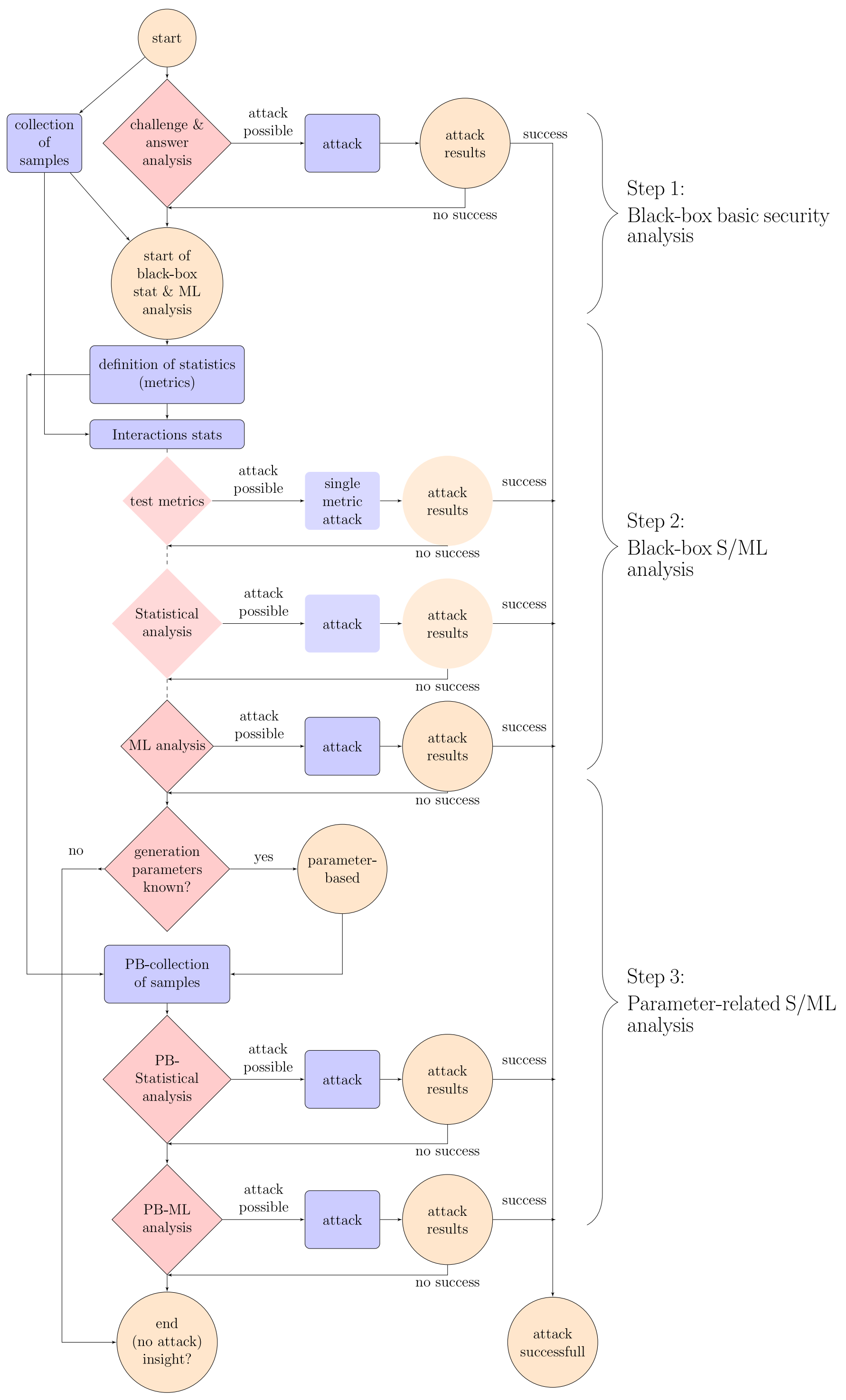


Fig. 1. BASECASS generic flow chart. Continuous lines represent compulsory tasks, while dashed lines represent optional tasks.

an iteration, as the main difference between both is how much internal information on the CAPTCHA design is available and thus able to be analysed.

Fig. 1 shows a simple depiction of the iterations of BASECASS, and also the relation between the definition of the metrics and later use of them in the posterior analysis. This figure serves as a guide and reference to understand the different iterations of BASECASS (black-box analysis, and if possible and necessary, parameter-based analysis). It also shows the steps of BASECASS: the challenge and answer space analysis, the black-box statistical and ML analysis and the parameter-based analysis. Note that as soon as we find weaknesses and test that they are strong enough to enable an attack, we can finish our analysis. This can happen in any of the steps of BASECASS.

Next, we will give a description of the different BASECASS steps: the challenge and answer domain analysis, the statistical/ML analysis, and the parameter-based analysis.

3.1. Step 1. Black-box basic security analysis

BASECASS starts by doing a Black-Box basic, initial security analysis of the CAPTCHA. This is an external analysis, based only on public information. During it, we will not pay attention to possible clues about the challenge design. In a general way, our Black-Box analysis can be divided into the following steps:

- **Phase I: Automatic interaction.** The objective of this phase is to develop a way to interact semi-automatically with the CAPTCHA. We want to do so in order to download challenges from the CAPTCHA, send the possible answers to the CAPTCHA server and receive its answer, so we can grade the answers.
- **Phase II: Analysis of the challenge space.** In this phase, we try to know what types and subtypes of challenges the CAPTCHA presents. For example, a CAPTCHA can present two different types of challenges: OCR and image-based challenges. The subtypes that it presents can be heavily distorted words or sentences (for OCR), and image classification and reconstruction (for the image-based challenges). We are interested into establishing what possible different challenge types are easily distinguishable by a bot. We will relate these subtypes to the base problem that the CAPTCHA is theoretically based on. Is the base domain easy to explore for a bot? If it is possible within a reasonable cost, we will also want to check statistically their distribution to search for deviations from uniform. When possible, we also compare its size to the size of the base problem of the CAPTCHA.
- **Phase III: Analysis of the answer space.** This phase focuses on checking the size and distribution of the possible answers to the challenges. Note that not always it will be possible to explore this space automatically. We might need to solve a number of challenges to study the distribution. This might be within reasonable costs or not depending on each case. Following with the previous example, we would like to know if all words or sentences are possible solutions for the OCR CAPTCHA, and what classes are used in the image-based CAPTCHA. We want to check their distribution, both globally and per challenge type. Are there any deviations from the uniform? If so, are they severe enough as to allow a successful attack?

Fig. 2 represents the part of the phase I that interacts with the CAPTCHA in order to collect the necessary data for the analysis that takes place in phases II and III. The first part detects and downloads the different types of challenges, and estimates their number by calculating the percentage of them that have already been seen using statistical methods such as Mark & Recapture [52]. The second part uses human input to reply to a number of challenges enough to later check their distribution. This is done for each challenge subtype that we want to study.

This black-box basic security analysis (Step 1) would render at least answers to the following questions:

- What types and subtypes of challenges does the CAPTCHA present? What parameters affect when they are served to the user?
- How many different challenges per subtype are there? If infinite, what is their domain?
- Do all seem equally difficult both for a human and a machine?
- How many possible answers are there for each challenge subtype?
- For both the challenge space and answer space, are they uniformly distributed? If not, what are the deviations?
- Is it possible to automatically detect challenge subtypes? If so, and if one of them is easier, is it possible to break the CAPTCHA at this point?
- How is the communication with the server, regarding the grading of answers?

During this analysis, other questions might rise giving further insight into the CAPTCHA: even if the domain and answer sizes are big enough, and their distribution is uniform, it is possible that we might find hints at some weak correlations between characteristics of the challenges and their correct answers. The next step deals with these kind of weaknesses.

3.2. Step 2. Black-box S/ML analysis

The previous step was our “first encounter” with the CAPTCHA. If it resists this basic analysis, we can move forward to the following step, that comprises a semi-automatic analysis of the side-channel statistics referred to the challenges.

In order to proceed, we will typically need to focus on one or a few of the subtypes of challenges served by the CAPTCHA, if there are many. This is so because possibly not all statistics will have sense for the different sub-challenge types. We will nevertheless focus on a subtype or subtypes that comprise a significant amount of the challenges served, as it would be useless to break them otherwise.

The analysis presented in this step would render at least answers to the following questions:

- Is there or are there a metric or metrics that are somehow correlated with the answer of the challenge?
- Is this possible correlation linear (if the SA is successful) or not (only ML is successful)?
- Is it possible to explain this correlation in a human understandable way? (Will depend on which ones are the most successful ML techniques)
- Is it possible to predict the accuracy of our correlation? I.e., it corresponds to some challenge subtype that can be classified by our metrics.
- Is this correlation possibly strong enough to base an attack on it?
- Which metrics contribute more to the correlation?

This step has four clearly defined phases. In the first one, we will prepare the challenges for processing. In the second phase, we select and/or create metrics (statistics) that are potentially useful to characterize the challenges. In the third phase we will use these metrics, together with the previously saved challenges and answers, to analyse the CAPTCHA statistically. This phase is optional. The fourth phase uses again the same metrics to analyse the CAPTCHA using different ML algorithms. A more detailed description of these phases follows.

- **Phase I. De-noising.** In some cases, a CAPTCHA designer might try to protect the information on the challenges by adding to them different types of noise or distortions. Sometimes, these can affect many of the metrics we can use on them. In these cases, we can think about de-noising techniques that might eliminate or minimize the influence of that noise in the metrics. Note that this phase is interrelated with the next phase, so they are complementary and not necessarily sequential.
- **Phase II. Pre-processing and transformations.** In some cases, we can think of a different domain in which the challenge might be easier to analyse. A typical case could be transforming an audio CAPTCHA from the time domain (wave) to the frequency domain using a FFT, or similarly transforming an image to a 2D frequency domain. Even though BASECASS does not emphasize to create anything like features to later analyse the challenges and answers, these kinds of transformations can be useful in some cases. This is something that should be done within the constrains of a low-cost attack.
- **Phase III. Definition of metrics.** For the selection and/or creation of statistics for the selected(s) challenge subtype(s), we will proceed as follows:

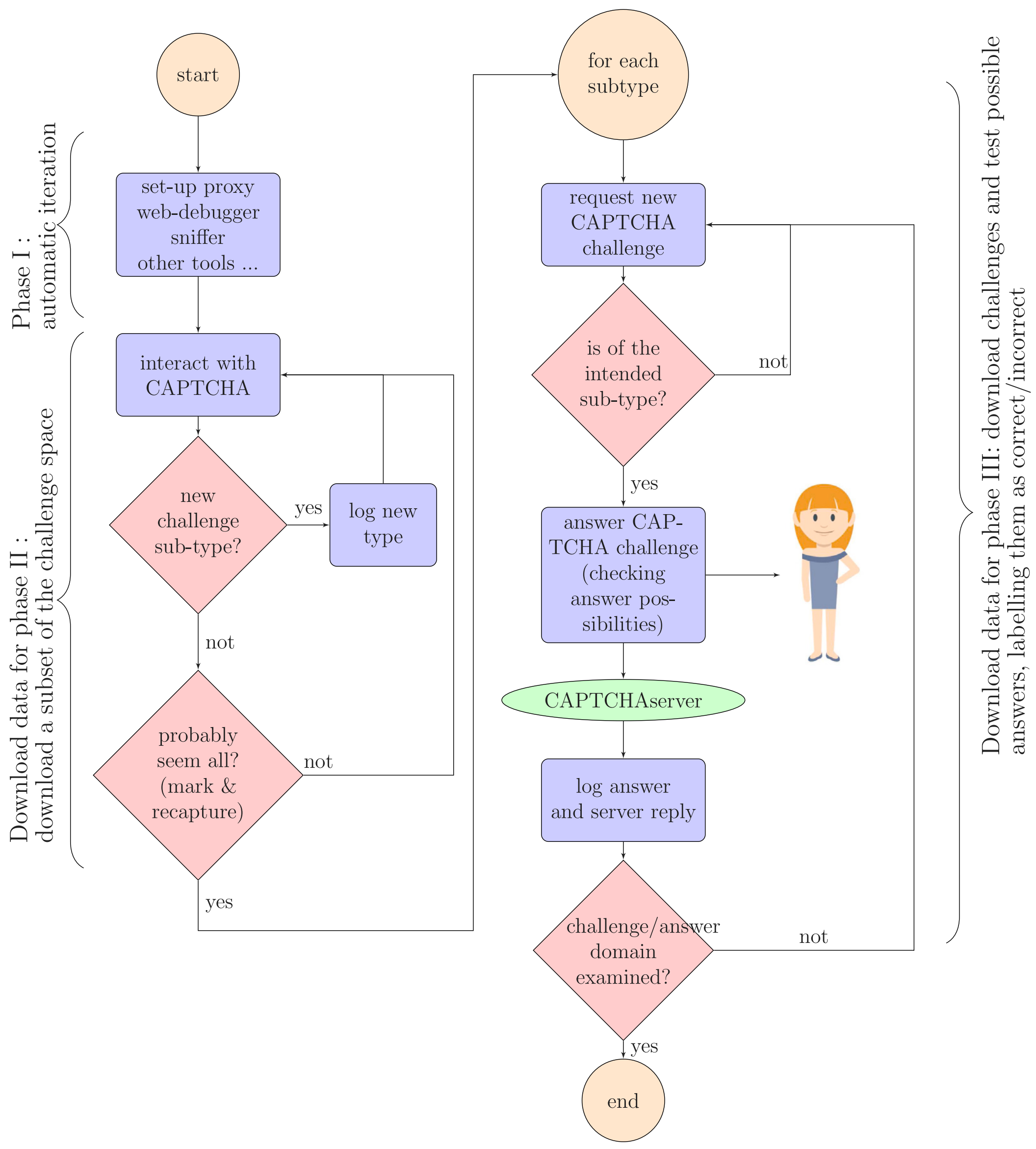

start
set-up proxy web-debugger sniffer other tools ...
Phase I : automatic iteration
interact with CAPTCHA
new challenge sub-type?
yes
log new type
not
probably seem all? (mark & recapture)
not
yes
Download data for phase II : download a subset of the challenge space
for each subtype
request new CAPTCHA challenge
is of the intended sub-type?
not
yes
answer CAPTCHA challenge (checking answer possibilities)
CAPTCHAserver
log answer and server reply
challenge/answer domain examined?
not
yes
end
Download data for phase III: download challenges and test possible answers, labelling them as correct/incorrect

Fig. 2. Generic flow chart for downloading the data needed for the Step 1 of BASECASS. This flow chart encompasses phase I. The data gathered will be analysed in phases II and III.

1. *Selection of basic statistics:* this step is done after we have examined a fairly broad subset of the CAPTCHA domain. Then, we will be able to select statistics that can be applied to the challenges. These will be general, broad sense statistics, that can be applied to the challenges in order to extract some information from then. The statistics will depend on media type, as they will be different for CAPTCHAs based on text, images, audio, or games. As an example of such general statistics, we can mention the randomness metrics returned by the ENT test applied to a binary file. These general metrics that can be applied to a very broad type of challenges, for instance, image challenges, to which we can also apply histogram of colour usage, pixel count, etc. These general metrics will depend on the media type of the CAPTCHA challenges, and on little else.
2. *Selection of tailored statistics:* in this step we select additional statistics that are more related to the CAPTCHA contents. For example, if it is a CAPTCHA based on images, then a statistic showing the quantity of image information can be useful. These statistics should be well-known for the CAPTCHA type or low-cost to obtain, having been previously defined. We are not interested in performing a full-blown CAPTCHA analysis here that will extract extremely significant, high-level information.
3. *In-challenge relational statistics:* (optional) in challenge relational statistics are those that relate different metrics obtained for different answers. If, for example, a challenge has 10^5 possible answers, instead of (or additionally to) giving the value of one of the metrics for those 10^5 possible answers, we can give the (for example) relative order of those values, so that way the statistical or ML algorithm will know if this solution has the lower (or top) value among the possible solutions for that challenge. These statistics are useful to relate the possible solutions of a single challenge among themselves. This might be useful or not depending on the CAPTCHA type. For example, a value of a metric of 155 might be good for an answer to a challenge but bad for another challenge. But knowing that value is the lowest among all possible answers (or highest, depending on what we are looking for) might provide much more information. As explained, a typical

way of doing this would be by ordering some of the previously extracted statistics within the possible answers to a challenge, and then registering this relative order, either absolutely or by percentiles.

- Selecting and/or creating the metrics is a phase that requires some experience, as it is not fully automatic. Yet, in this phase we will use some general guidelines, which broadly speaking can be:
  - Previous literature about well-known side-channel attacks.
  - Randomness metrics that can be applied to the challenge type. Among these, and of special interest, are cryptographic tests of randomness.
  - Low-cost metrics: metrics that are already implemented and easy to use. These are typically extracted from libraries that can manipulate the media formats that contain the challenges (text, images, audio, ⋯).
- This is an important phase, as the efficiency of the following S/ML analysis depends on it, so it is worth investing some time. If we cannot come up with any possible metric, we can just use the well-known ones for a basic security check. If possible, trying new metrics (always based on readily available software or procedures) can lead to interesting results.
- Normally, there will no be a need for feature selection, given than the number of applicable metrics will not be very high. If this is a problem, some Exploratory Data Analysis can be performed to select the most promising metrics. This can also be done using L1 or Lasso regularization, correlation analysis, or using Linear Discriminant Analysis for dimensionality reduction. The label will be whether the answer is correct, for the set of possible answers.
- When we have both the metrics and some correctly and wrongly solved challenges, we can proceed to the Statistical and ML analysis phases, which constitute the first iteration of BASECASS.
- **Phase IV: Statistical analysis (optional) and ML analysis:** we will try to find correlations among challenge data extracted using our metrics and the solutions. To do so, we will apply statistical analysis techniques. If we skip this analysis or if we do not obtain positive results, ML techniques might be suitable. From an attacker point of view, we can skip the statistical analysis and proceed directly to a ML analysis, that renders more powerful tools than the statistical analysis, as some ML algorithms are able to automatically cope with non-linear classification and/or heavily unbalanced data sets.
- From an attacker point of view, a ML analysis has the potential to provide for the most interesting results. For the ML analysis, we use the previously solved challenges, and the metrics data extracted from them to try to find a relation among the challenges and their correct answers. We do so using ML algorithms to look into the data trying to find significant patterns. We will try different families of ML algorithms with default parameters to search for the one that finds stronger relationships among challenges and their answers. In a second step, we can grid-walk its parameters to fine-tune the ML algorithm to obtain the best possible result. During this step, we will use either different test and training sets, or Cross Validation.

It is possible that, after this analysis, we will focus more on a subset of the metrics, and maybe come up with additional metrics that will require a re-run of this iteration. This is ok, as this iteration is fully automatic.

An important aspect of an attack is the cost. Given the lower cost of using Deep Learning, and the availability of models that can be used for transfer learning using a few hundred labelled samples, the use of Deep Learning to see whether the CAPTCHA is resilient or not to basic attacks should be considered here. This applies mostly to image and audio-based CAPTCHAs. Although a DL-based attack cannot be considered a side-channel attack, given its availability, it should be considered part of a basic testing process whenever it applies.

3.3. Step 3. Parameter-related S/ML analysis

This step explores possible weaknesses and correlations between the challenges and their correct answers, but does so taking into account the values of the different parameters that are used when creating a challenge. Note that these values are not always accessible nor easy to deduct from a produced challenge. Thus, this step is not always possible. Next, we will comment when this step is applicable, as well as its utility: what additional information we want to extract.

This step will typically only be possible if either the CAPTCHA designer is collaborating with the analysis, if the CAPTCHA is open-source, or when the value of the main parameters affecting the generation of the CAPTCHA challenges are evident given a particular challenge. If these circumstances are not met, then it is in general impossible, or costly, to learn the value of the challenge creation parameters from a particular challenge. This analysis itself can be more costly than the attack we are looking for.

For example, let us imagine a CAPTCHA shows synthetic images of people from different professions that the user has to categorize by social status or perceived income. When the CAPTCHA wants to create an image to be used in a challenge, it has to decide (typically randomly) the value of some parameters: the profession (among a certain number and type of professions), a particular 3D model (among a number of models of different types), what colours to use, the field of view of the image, what additional elements to use (number of clothing, tools, etc.), lightning conditions, etc. The value of all these parameters affects the challenge created. Their particular value might affect also the difficulty of the challenges created, and that is precisely what we are trying to discover.

The type of questions that this part of our analysis wants to answer are such as if given different values of parameters of the CAPTCHA generation algorithm, some of them especially weak and thus should be avoided, or if there are factors or measures that contribute more to the strength of the CAPTCHA, or what parameters are more sensible towards the CAPTCHA security. In a way, what we want to know is whether the CAPTCHA design seems to be correctly using the base problem to its full strength, or at least, be certain that it is avoiding specially weak cases.

Typical questions asked during this phase could be: is it possible for an attacker to identify identical elements (backgrounds, sprites, etc.)? Is it possible to automatically deduct some of the values of the parameters affecting the generation of a given challenge? How do the different design elements affect the strength of the CAPTCHA? The tools for this second iteration are the same used in the earlier analysis. Now, we will use them with restricted parameter values and study how they perform in these cases.

If we do not have access to the CAPTCHA source code or the collaboration of its designer, in some cases we still can separate the correctly and wrongly solved challenges in sets depending on the different parameter values with which they were generated. If we have access to the challenge creation mechanism, we can generate challenges automatically using different parameter values.

During the exam of these questions, we forget about the user friendly aspect of the CAPTCHA. What we want to know is only how they affect its security. To measure how these different design decisions affect the CAPTCHA security, we will use the same analysis tools as we used in the previous step. If during that analysis we find that certain tools are more promising than others, we will focus our efforts in those, but we will use in any case all of them, as a different parameter set for the CAPTCHA can render it susceptible for a different type of attack.

This analysis would render at least answers to the following questions. If we found a correlation in the previous step or in this one:

- Does this correlation affect to all challenges uniformly, or does it depend on some parameter values?
- How does each parameter and parameter value affect this correlation?

- Is it possible to invalidate this correlation, using some parameter values?
- Is there one or different correlations, depending on the parameter values?

If we have not found any strong correlation.

- Is there any sub-domain of parameters that shows a hint at a correlation, and should be further explored with more examples or values?
- What parameter values seem to give the most uniform distributions in the metrics used?

3.4. BASECASS summary table

The procedures used in each application of BASECASS can be summarized in a table, along with the results found. If, during the distinct phases of the analysis, BASECASS finds vulnerabilities that might be sufficient enough for a side-channel attack, and if such attack is feasible and within the ethics of each particular case, then we can also include the results of such attack. Depending on them, it might not be necessary to continue with the application of BASECASS, if the CAPTCHA is considered broken beyond a reasonable effort of correction.

The findings that result out of the different BASECASS steps can be summarized in a template table. This table is divided in different types of analysis, and at the end of each one we present the main findings. Each section of the table represents one analysis type of BASECASS. Some sections of the table are optional and dependent on the result of the previous sections. Next section shows examples of the BASECASS table applied to different CAPTCHAs attacks.

3.5. Design guidelines

Although it is difficult to offer a general design guideline that will apply to every CAPTCHA design, what we can learn from the application of BASECASS to several CAPTCHAs, and also other published side-channel attacks to CAPTCHAs, is that it is important to have in consideration, while designing the CAPTCHA, how to prevent correlations that can, somehow, be measured just accessing the challenge — and after implementing it, measure them.

4. Cases of study

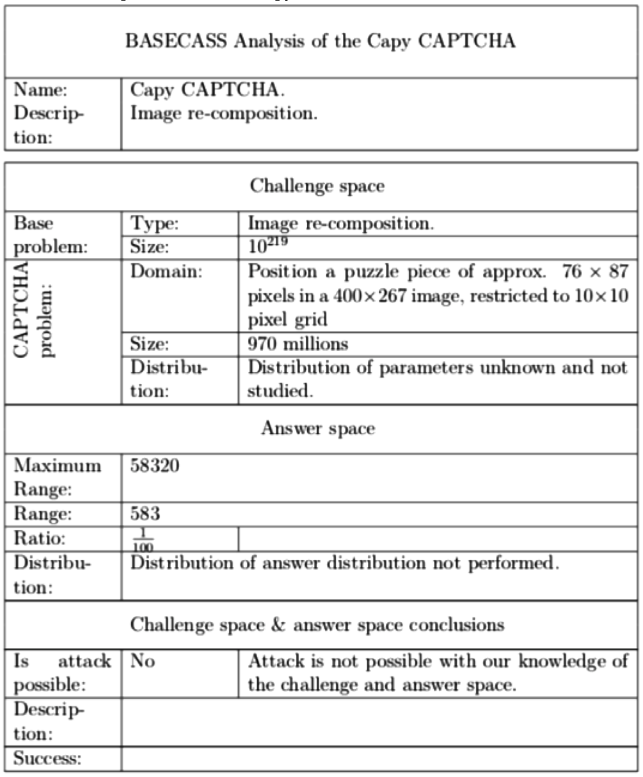
In this section we provide three examples on the application of BASECASS. More examples can be found in the PhD of Hernández-Castro [53].

We want to stress that BASECASS does not claim to be able to check the security of any CAPTCHA type. It aims though to be general enough to be applicable to many different CAPTCHA types. And this is what we tried to prove by selecting the three CAPTCHA types that were regarded as secure, had earned a strong reputation, offered an original proposal and were previously not broken when we analysed them.

The reader may wonder if BASECASS can be applied on reCAPTCHA, one of the leaders in the CAPTCHA market. In 2014, Google decided to add an additional security layer to it. They called it “v2” and determined publicized it as being much more intelligent and robust, avoiding most bots automatically, and only showing the CAPTCHA on very few occasions. However, this type of security, based on *an algorithm* that is private, and in principle only known to Google, is what is called Security by Obscurity. This extra layer of security is not really part of the reCAPTCHA CAPTCHA and should not be considered as such. The P in the acronym CAPTCHA stands for a Public algorithm: thus the key security point of the CAPTCHA should reside on the strength of the algorithm, not on its secrecy. Then, BASECASS does not try to address the extra layers of security one might add to a security mechanism, but the CAPTCHA security mechanism itself.

Besides, reCAPTCHA has been at least broken once that we know before BASECASS was conceived.

Table 1
BASECASS description section for Capy.



BASECASS Analysis of the Capy CAPTCHA			
Name:	Capy CAPTCHA.		
Description:	Image re-composition.		
Challenge space			
Base problem:	Type:	Image re-composition.	
	Size:	10^{219}	
CAPTCHA problem:	Domain:	Position a puzzle piece of approx. 76×87 pixels in a 400×267 image, restricted to 10×10 pixel grid	
	Size:	970 millions	
	Distribution:	Distribution of parameters unknown and not studied.	
Answer space			
Maximum Range:	58320		
Range:	583		
Ratio:	$\frac{1}{100}$		
Distribution:	Distribution of answer distribution not performed.		
Challenge space & answer space conclusions			
Is attack possible:	No	Attack is not possible with our knowledge of the challenge and answer space.	
Description:			
Success:			

4.1. BASECASS analysis of Capy CAPTCHA

Here we will briefly comment some of the aspects of applying BASECASS to the Capy CAPTCHA and present the result summary table of its application. See Section 2.9 for a detailed description of puzzle CAPTCHAs.

In the **first step** of BASECASS, we analyse the interaction of Capy with its server. A whole PNG image is transmitted that contains a sub-image of 400×267 pixels (the challenge image) and, in its right part, a puzzle piece of approximately 76×87 pixels, that is present to the user below the challenge image. This size might vary as the puzzle piece shape could change. The user answer is sent as a string containing the succession of drag & drop coordinates that the user’s pointer crosses in order to put this puzzle piece in place, coded using base 32.

Following Phase II, we analysed the challenge domain and we determined that Capy was using four background images, and that the puzzle piece could have different shapes and be from any of these images. Also, the puzzle piece void inside the background image is filled with a portion from any of the four backgrounds available. As we wanted to know whether the base problem Capy is based on could be good enough for a CAPTCHA, we assumed from now own that Capy authors could easily augment the number of background images to thousands or millions, and proceeded assuming this.

BASECASS encourages us to compare the base problem space with the challenge problem space to have a basic understanding of their relative difficulty. To measure the base problem space size, we assumed that we limit the image size to that used by Capy. In that case, there are $(400 \times 267)^{8^3}$ maximum images. For each one, we can select up to $(76 \times 87)^{8^3}$ −1 fillings for its puzzle piece (the size varies, but it is around 76×87 pixels). Each one, we can position in $\frac{400}{10} \times \frac{267}{10}$ different positions. This is so because Capy restricts the movements to a 10×10 grid, to make it easier for the human users to find the correct position. This makes a total of $(400 \times 267)^{8^3} \times (76 \times 87)^{8^3} - 1 \times 40 \times 26 \approx 10^{219}$ images.

Table 2
BASECASS metrics definition section for Capy.

Metrics	
Denoising:	No de-noising technique used.
Pre-processing:	No pre-processing technique used
Generic	Histogram of colours used Number of pixels detected as borders Results of the ENT test: entropy; χ^2 test; arithmetic mean; interpretation as a sequence of 24-bit X and Y coordinates for estimating π using a Monte-Carlo algorithm; serial correlation coefficient
Order	Order in size after compression (JPEG). Order in number of pixels detected as borders.
Specific/ Tailored	Size after compression using the JPEG lossy compression algorithm.
Test of metrics	
JPEG size order : metric is able to guess correct answer on a large number of cases.	
Is attack possible:	Yes
Description:	JPEG size order for a single image.
Success:	While testing this metric off-line, it seems to perform well enough for a successful attack, possibly with over 20% success ratio.

Table 3
BASECASS results and conclusion sections for Capy.

Attack & Results If previous phase leads to an attack	
Possible?:	Yes
Description:	Given an image, we position its puzzle piece in the 40×26 possible positions. The resulting image is compressed using JPEG. We choose as correct the position that renders the image that, once compressed, requires a smaller file size.
Success rate:	65% on an attack for 1000 challenges.
Observations:	
Conclusion	
Weaknesses:	• Small set of possible background images (4). • Artificially reduced set of possible answers (10^2 smaller than the possible 400×267).
Broken?:	Yes, with a 65% success rate.
Workarounds:	• Increased number of background images through larger database and image alterations. • Broader solution space (larger size of images, more possible puzzle positions, increased number of puzzle pieces). • Challenge pre-filtering to limit the usefulness of our metric.

To measure the size of the challenge problem, we perform a similar calculation, but now with the real number of images, four. The number of possible puzzle fillings is then $\approx 4 \times (400-76) \times (267-87) - 1 = 233279$.

Table 4
BASECASS description section for CRC CAPTCHA.

BASECASS analysis for the CRC - Empathy		
Name:	CRC - Empathy	
Description:	A short news excerpt typically related to Human Rights.	
Challenge space		
Base problem:	Type:	Empathy
	Size:	A human emotional reaction or subjective stand on a subject. Depending on the classification, there might be 8 basic emotions (not including weaker and stronger variants, and also complex emotions based on these) (Plutchik, 1991), or up to 42 different emotions[12]
CAPTCHA problem:	Domain:	Unknown, but seems to categorize the news excerpts in two categories, *positive* vs. *negative*.
	Size:	The news excerpts are from a set of 21 elements.
	Distribution:	Parameters do not vary through all the challenges: the news excerpts are a fixed.
Answer space		
Maximum Range:	Apparently, there seems to be a coarse discrimination only among positive and negative reactions.	
Range:	2	
Ratio:	$\frac{42}{2=21:1}$	
Distribution:	They are imbalanced, as 66% of the news excerpts are negative.	

The number of positions to place the puzzle piece is 40×26. Thus, the total size: 970 millions.

BASECASS encourages us to consider calculating the distribution of challenges. In this case, the only parameters we could consider are the background, the puzzle piece shape, its position, and the filling used for the puzzle piece void on the background image. Although the parameters could be reconstructed once all the backgrounds are known, the cost of this analysis is out of scope for a low cost attack, so it is not produced in this case.

BASECASS also compares the possible answer space with the real answer space used in the CAPTCHA. The answer space is easier to calculate if we restrict ourselves to an image of the size of Capy, the maximum possible should be $(400-76) \times (267-87) = 58320$. As Capy restricts movements to a 10×10 grid, this is instead 100 times smaller, that is, $(400-76) \times (267-87) \approx 583$. That means that a random brute-force attack has a success rate of 0.17%, slightly high, but possible for a CAPTCHA according to some authors.

BASECASS recommends to determine if the distribution of answers is uniform or is instead skewed. As the answer space is not small, for this test to be significant we should collect a very large number of examples and their solutions, at least in the order of 25 000. This test is again too costly, and in this particular case was not performed.

Now, we have some basic data about Capy, and we proceed with the **second step** of BASECASS and define the requirements and metrics for the S/ML analysis. Some of the possible metrics are:

- General purpose metrics:
  - Histogram of colours used: as Capy fills the space where the puzzle piece should go with a sub-image, we could try to see if it adds colours to the image and if the colour histogram could be modified.
  - Number of pixels detected as borders: we will use different border detection algorithms and count afterwards what percentage of the pixels are detected as borders.

Table 5
BASECASS metrics definition section for CRC.

Challenge space & answer space conclusions		
Is attack possible:	Yes	A brute-force attack would be possible, given the small answer domain and not uniform distribution of answer appearances.
Descrip- tion:	We could just reply picking randomly any of the possible negative answers, even better, any negative answer that appears.	
Success:	If we would be able to read the 133 possible answers, and always pick the negative one, we would pass this part of the challenge 71% of the time.	
Metrics		
Denoising:	No denoising technique is needed (content is text).	
Pre- processing:	Some pre-processing can be done using the text categories on WordNet.	
Generic	Appearance, using TF-IDF.	
Order		
Specific/ Tailored	Three possible transforms using WordNet: no transform, synonyms, hypernyms.	
Data preparation		
Training set	Size:	643 training news excerpts from the Civil Rights Association, used for training and testing using 10-fold CV.
	Balance:	167 positive, 290 negative, 165 neutral.
	Notes:	English stop-words removed. TF-IDF with a cut-off value of two.

Table 6
BASECASS ML analysis and conclusion sections for CRC.

ML analysis	
Selection:	f_1[13] and classification accuracy.
Best algo- rithms:	SVM Linear[14] using synonyms.
Accuracy:	90%
κ-statistic :	0, 85
S/ML attack & Results (If previous phase leads to an attack)	
Possible?:	Off line classification accuracy is 90%, so an attack seems plausible.
Descrip- tion:	Classification of the news excerpts in either positive or negative, using previously trained classifier.
Success rate:	Combined for both OCR and Empathy: 20%
Observa- tions:	
Conclusion	
Weak- nesses:	• Small set of challenges (21). • Small set of possible answer values (positive or negative). • Appearance of answer values is not uniform ($\mathcal{X}^2_{20}$ with a p-value = 0.336).
Broken?:	Yes. With a 20% success rate using simple metrics.
Work- arounds:	• Finer emotion classification. • More uniform distribution of emotions.

- Results of the ENT test: a number of general metrics, including the entropy, serial correlation, lossless compression rate, Monte-Carlo estimation of π, etc.

• Ad-hoc metrics:

- Size after compression: an original image will typically be more regular than the same image with a puzzle piece filled with some other image. We could use compression algorithms, like JPEG, to verify if the size of the resulting compressed image has been affected.

• Comparative metrics:

- Order in size after compression: if the size after compression is a measure of goodness of the solution, an ML algorithm could know which is the smallest/largest one (or n) of the set of possible solutions to a challenge.
- Order in number of pixels detected as borders: in a similar fashion, this would possibly serve an ML algorithm to improve the accuracy while classifying among a set of possible solutions.

BASECASS includes a step to test the performance of the different metrics. In particular, the metric that analysed the resulting file size after JPEG lossy compression seemed well able to break the Capy CAPTCHA. According to BASECASS, we performed an attack based on this result and we did not need to use a ML classifier in order to completely break the Capy CAPTCHA. Tables 1–3 summarize the results obtained.

4.2. BASECASS analysis of the Civil Rights CAPTCHA

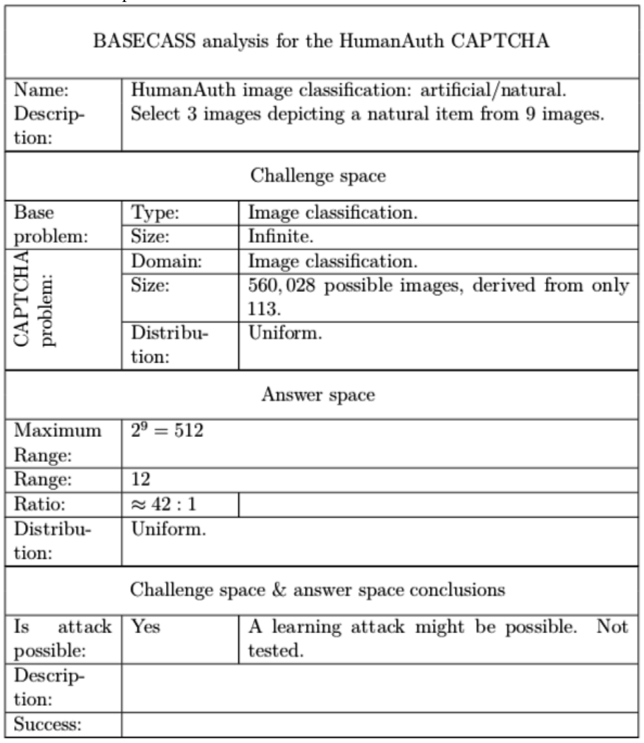
In this section we describe the BASECASS analysis to the CRC CAPTCHA (go to Section 2.8 for a detailed description of the CAPTCHA).

The **first step** of BASECASS requires to analyse and create a way to interact automatically with it. In the case of the CRC, we need to download enough data for this analysis. After downloading 1000 challenges, we only found 21 news items. This number is insufficient and could be easily identified by a bot allowing us to do a brute-force attack in which a program would learn the possible correct answers just by trial and error. BASECASS also recommends that we analyse these 21 news: both how many times they are actually presented to the user, and in answer space (positive and negative news). We find them to be it strongly biased towards negative news. Their appearances distribution remains similarly biased.

At this point we find that this part of the challenge is solvable by a brute-force attack, if the answers to each news excerpt are coarsely divided into positive and negative. As we do not know whether this is the case, we proceed to do some analysis of how the CAPTCHA server validates the answers. Apparently, the answer has to actually come from the set of three answers presented to the user. In any case, this part of the challenge is to be considered broken, that is, it does not add security to the CAPTCHA, because if the emotional answers could be read and classified into positive or negative, it would be straightforward to solve the challenges.

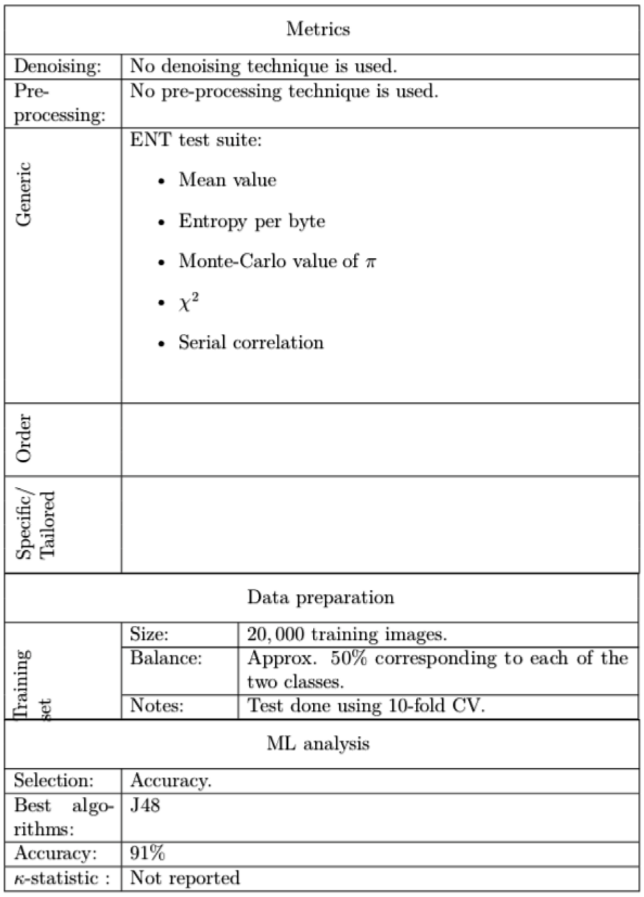
Phase III of step I recommends to analyse the answer space, both theoretical and the real one used in the CAPTCHA. Note that, if we restrict the answers to one word, the potential answer space is not very large: according to some word lists, there are around 167 1-word

Table 7
BASECASS description section for HumanAuth CAPTCHA.



BASECASS analysis for the HumanAuth CAPTCHA		
Name: Description:	HumanAuth image classification: artificial/natural. Select 3 images depicting a natural item from 9 images.	
Challenge space		
Base problem:	Type:	Image classification.
	Size:	Infinite.
CAPTCHA problem:	Domain:	Image classification.
	Size:	$560,028$ possible images, derived from only 113.
	Distribution:	Uniform.
Answer space		
Maximum Range:	$2^9 = 512$	
Range:	12	
Ratio:	$\approx 42:1$	
Distribution:	Uniform.	
Challenge space & answer space conclusions		
Is attack possible:	Yes	A learning attack might be possible. Not tested.
Description:		
Success:		

Table 8
BASECASS metrics definition section for HumanAuth.



Metrics		
Denoising:	No denoising technique is used.	
Pre-processing:	No pre-processing technique is used.	
Generic	ENT test suite: • Mean value • Entropy per byte • Monte-Carlo value of π • χ^2 • Serial correlation	
Order		
Specific/ Tailored		
Data preparation		
Training set	Size:	20,000 training images.
	Balance:	Approx. 50% corresponding to each of the two classes.
	Notes:	Test done using 10-fold CV.
ML analysis		
Selection:	Accuracy.	
Best algorithms:	J48	
Accuracy:	91%	
κ-statistic :	Not reported	

emotions, so adding a few of the modifiers "very", "a bit", "totally" as the CRC does, we can get to 668 words and two words combinations.

After some initial interactions, we start seeing repetitions on the set of possible answers. This is expected, given that the amount of possible emotions that can be described with one or two words is limited. We download a set of 1989 challenges and manually classify the possible answers, which are 133. Most of them appear with more than one repetition, so we consider this to be the total set of possible answers (or a good approximation to it) for our further analysis. The distribution of their appearance is not uniform, with a Pearson's χ^2_{132} value of 482.12. This allows for a potential brute-force attack in which we will repeat the most frequent five answer (to avoid possible detection by repetition), that can pass the CAPTCHA with a 1.2% success rate.

Then, we can now proceed to the **second step** of BASECASS, the S/ML analysis of the CRC. The answers of the CRC are protected using Securimage that offers many configuration parameters. In this case, Securimage is used with a static configuration, that includes two or three lines over the text.

In order to proceed with the S/ML analysis, we want to define what metrics to use. Initially, we choose quite simple metrics: the total pixel count, as some characters use more pixels than others, could give us an idea of the size (in pixels) of the characters used; we measure in relatively to the maximum. To be more precise, we also use the pixel count per column, and per groups of three and five columns.

When we decided to use these metrics, we realized that the lines introduced by Securimage might influence their result. A way in which we could minimize their impact is if we consider instead the differential in pixels, because a line that has approximately the same width and an horizontal component (that is, is not purely vertical) would use approximately the same number of pixels per column during its length. Of course, this still would alter the results of our metrics when the lines start and end, and also when they occlude parts of a character. But still, this might be a good way to, in general, decrease the influence of the lines over our metrics. Thus, we decide to add these differential metrics.

In order to read the text of each of the three images in each challenge, we define these metrics to extract from every image, and proceed to train a set of classifiers on them. We obtained the best classification results with Linear Regression and Linear Support Vector Machines (LibLINEAR) [54], attaining 59.3% accuracy. This means that in a challenge composed of 3 possible answers, we have 28.8% of correctly reading the three possible answers and 35% of reading two of them.

The metrics to use for the classification of the news bits are taken from basic NLP techniques. In particular, after some data cleaning removing country names, stop-words, etc., we transform the words to their WordNet synset representations and to TF-IDF normalized vectors with a cut-off of two. In order to train our classifiers, we use 622 manually downloaded and classified news bits from the Civil Rights Defenders. We test different classifiers and different syset representations. Finally, we choose SVM Lineal, translating the texts to chains of WordNet hypernyms, which obtained 1.00 precision during our tests.

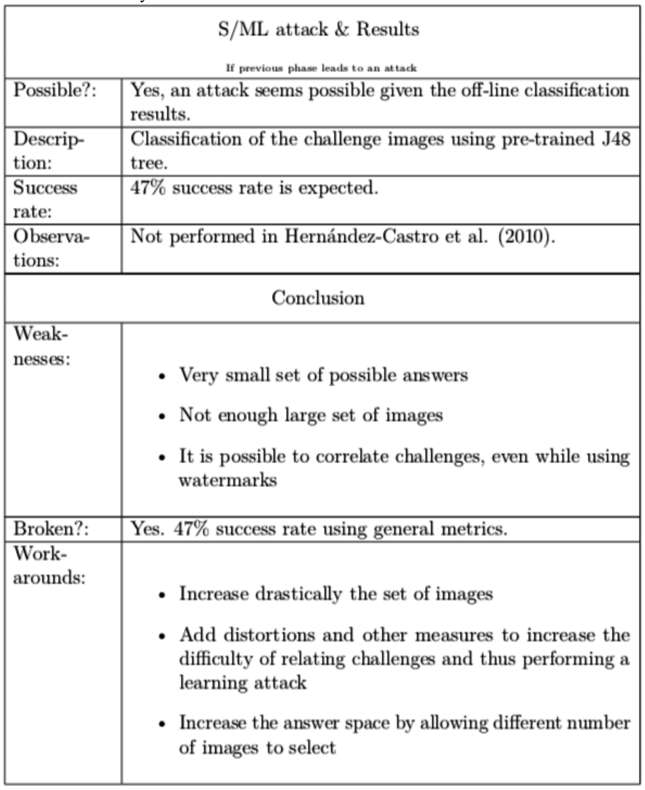
Following BASECASS, we put together a program that automatically downloads and answers CRC challenges, testing if its answer is classified as correct or not by the CRC CAPTCHA server. After 1000 challenges, we obtained a success rate of 16.5% challenges correctly solved. Using an slightly improved version that memorizes previous results, we soon obtained a success rate of 20.7%.

Tables 4–6 summarizes the application of BASECASS for the CRC CAPTCHA to its OCR/text sub-challenge.

4.3. BASECASS partial analysis of the HumanAuth CAPTCHA

In this section we will present the application of BASECASS to another CAPTCHA proposal that has already been analysed from a

Table 9
BASECASS ML analysis and conclusion sections for HumanAuth.



S/ML attack & Results	
	If previous phase leads to an attack
Possible?:	Yes, an attack seems possible given the off-line classification results.
Description:	Classification of the challenge images using pre-trained J48 tree.
Success rate:	47% success rate is expected.
Observations:	Not performed in Hernández-Castro et al. (2010).
Conclusion	
Weaknesses:	• Very small set of possible answers • Not enough large set of images • It is possible to correlate challenges, even while using watermarks
Broken?:	Yes. 47% success rate using general metrics.
Workarounds:	• Increase drastically the set of images • Add distortions and other measures to increase the difficulty of relating challenges and thus performing a learning attack • Increase the answer space by allowing different number of images to select

security standpoint. The application of BASECASS will be partial, based on the publicly available data of its security analysis by Hernández-Castro [20]. At the end of our partial application, we will check if BASECASS is able to find whichever weaknesses have been reported.

The HumanAuth CAPTCHA is an Open Source CAPTCHA that asks users to distinguish between images with natural and non-natural contents. The HumanAuth application comes with an image repository consisting of 45 nature images and 68 non-nature ones in JPEG format.

The **first step** of BASECASS strongly recommends to create a way to interact automatically with the CAPTCHA being studied. We could analyse the HumanAuth CAPTCHA as either a text-based CAPTCHA or an image CAPTCHA. Hernández-Castro et al. [20] decided to do the second, so we will follow this route.

BASECASS recommends to estimate the size of the base problem and the size of the real problem being posed by the CAPTCHA and then compare them, in a way to estimate its strength compared to the base problem. The size of the images is 100 × 75 pixels, using 3 RGB channels with 8-bits per channel. The size of the set of all possible images is equal to $100 \times 75 \times 2^{8\times3} = 125829120000$ possible images, even though this includes all images that differentiate from another in just a pixel and a bit that is, many would look the same to the human eye. The size of the challenge problem is much smaller though, as it includes 45 nature images and 68 non-nature images, that are protected with the addition of a watermark. The watermark does not change, it just changes the position in which it within the image. The original watermark has a size of 16 × 16 pixels.

Then, there are $(100-16)\times(75-16) = 4956$ positions for it. Thus, the size of the challenge problem is equal to $(68+45)\times 4956 = 560\,028$ total possible images different at pixel level, although their differences are typically less than $100\times\frac{(100-16)\times(75-16)}{100\times75} = 66\%$ different from many others, and as little as 16 pixels different (or less) than the closest one.

Phase II of the first step also recommends to estimate the answer space of the CAPTCHA and its distribution, in a way to estimate its strength against brute-force attacks. The answer space of the HumanAuth CAPTCHA is reduced: we need to pick a number of elements from a set of 9. Thus, theoretically the number of answers could be

$$\sum_{i=1}^{9}\binom{9}{i} = 2^9 = 512$$

Yet HumanAuth presents always just 3 images to select, thus the answer space is the smaller $\binom{9}{3} = 9!/3!\times 7! = 8 * 9/3 * 2 = 12$ only different answers.

According to the source code, their distributions should be uniform. Given the small answer space, and the fact that many challenges could be identified as having a similar image, as they are quite similar at pixel level, it might be possible to perform a learning attack against HumanAuth. This is not the attack that Hernández-Castro et al. [20] performed, as they want to know whether the idea behind HumanAuth was sound, even if their image database was bigger.

After completing the first step of BASECASS, we proceed to the **second step**, BASECASS S/ML analysis. To do so, it is necessary to choose some metrics that we will use to extract information from the challenges. Hernández-Castro et al. decided to use the ENT test for this. This test provides several numerical values for each image: the numerical value of the entropy, as measured by ENT in bits per byte; the χ^2 value for the corresponding degrees of freedom (width x height in pixels); the mean value of each byte; the value of π obtained using a Monte-Carlo algorithm that is supplied with the image data instead of a random stream; and the correlation of one byte against the next one.

Hernández-Castro et al. [20] apparently used the whole set of HumanAuth as training images, checking them using CV. They obtained a 78% accuracy using Random Forests. This indicates that an attack might be possible. In this situation, BASECASS encourages us to test our findings performing an attack. In order to test an attack, they create a set of 20,000 images using the provided watermark. They do so using the public source code available. The accuracy of the same classifier drops to 72%, but attain 91% using J48. Although they do not implement an attack, it is expected that with such accuracy, an attack would be successful on $0.91^8 = 47\%$ of occasions.

Tables 7–9 summarizes the partial application of BASECASS to the HumanAuth CAPTCHA and the results found.

5. Conclusions

In this paper we have summarized the main CAPTCHAs design and their attacks. We have also presented a framework to assess a basic level of security for new CAPTCHA designs called BASECASS and some examples where BASECASS has been applied.

BASECASS could be implemented as a software tool. The parts of it that depend on the specific CAPTCHA could be implemented as plug-ins. The part of it that needs a few labelled examples could be implemented through third-party CAPTCHA solving services. If implemented as Open Source, we hope that the research community would find it useful and that new CAPTCHA designers would use it to assess a basic security level for their designs.

BASECASS is able to detect common weaknesses in a number of cases. More so, it is able to do so in a methodological way. The heavyweight lifting of the analysis is left to ML algorithms. In any case, we find BASECASS successful, even when using generic metrics.

Although BASECASS does not provide guidance for possible workarounds or mitigations of the possible vulnerabilities found, it is a framework for assessing that a new CAPTCHA proposal meets a basic level of security against primary and statistical side-channel attacks.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is funded by the JCLM project SBPLY/19/180501/000024 and the Spanish Ministry of Science and Innovation project PID2019-109891RB-I00, both under the European Regional Development Fund (FEDER).

The work of María D. R-Moreno was partly supported by the Spanish Ministry of Sciences and Innovation under the grant number PRX18/00563.

All the vulnerabilities discovered have been responsibly disclosed to the CAPTCHA authors.

Compliance with ethical standards

This article does not contain any studies with human participants or animals performed by any of the authors.

References

[1] Naor M. Verification of a human in the loop or Identification via the Turing Test. Tech. rep., 1996.
[2] Ahn LV, Blum M, Hopper NJ, Langford J. CAPTCHA: using hard AI problems for security. In: Proceedings of the 22nd international conference on theory and applications of cryptographic techniques. Eurocrypt'03, Berlin, Heidelberg: Springer-Verlag; 2003, p. 294–311.
[3] Athanasopoulos E, Antonatos S. Enhanced CAPTCHAs : using animation to tell humans and computers apart. Ifip Int Fed Inf Process 2006;4237:97–108.
[4] Mohamed M, Gao S, Saxena N, Zhang C. Dynamic cognitive game CAPTCHA usability and detection of streaming-based farming. In: Usable Security (USEC 2014). San Diego, CA, USA: Internet Society; 2014.
[5] Yan J, Ahmad aE. Breaking visual CAPTCHAs with naive pattern recognition algorithms. In: Twenty-third annual computer security applications conference (acsac 2007). Miami Beach, Florida, USA: IEEE; 2007, p. 279–91.
[6] Hindle A, Godfrey MW, Holt RC. Reverse engineering captchas. In *2008 15th working conference on reverse engineering*, Antwerp, Belgium, 2008.
[7] Bursztein E, Martin M, Mitchell J. Text-based CAPTCHA strengths and weaknesses. In: Proceedings of the 18th acm conference on computer and communications security. Ccs '11, New York, NY, USA: ACM; 2011, p. 125–38. http://dx.doi.org/10.1145/2046707.2046724.
[8] Nguyen DV. Contributions to text-based captcha security (Ph.D. thesis), University of Wollongong; 2014.
[9] Lillibridge M, Abadi M, Bharat K, Broder A. Method for selectively restricting access to computer systems. Tech. rep., 2001, US Patent 6, 195, 698.
[10] Leung TK, Burl MC, Perona P. Probabilistic affine invariants for recognition. In: Proceedings of the 1998 ieee computer society conference on computer vision and pattern recognition. Santa Barbara, California, USA: IEEE; 1998, p. 678–84.
[11] Mori G, Malik J. Recognizing objects in adversarial clutter: breaking a visual CAPTCHA. In: Proceedings of the 2003 ieee computer society conference on computer vision and pattern recognition, Vol. 1. Madison, Wisconsin, USA: IEEE; 2003, p. I–134–I–141.
[12] Yan J, Ahmad ASE. A low-cost attack on a microsoft CAPTCHA. In: Proceedings of the 15th acm conference on computer and communications security. Alexandria, VA, USA: ACM; 2008, p. 543–54.
[13] El Ahmad AS, Yan J, Marshall L. The robustness of a new CAPTCHA. In: Proceedings of the third european workshop on system security. Eurosec '10, Paris, France: ACM; 2010, p. 36–41. http://dx.doi.org/10.1145/1752046.1752052.
[14] Bursztein E, Aigrain J, Moscicki A, Mitchell JC. The end is nigh: generic solving of text-based captchas. In *8th usenix workshop on offensive technologies (woot 14)*, San Diego, CA, USA, 2014.
[15] Gao H, Yan J, Cao F, Zhang Z, Lei L, Tang M, Zhang P, Zhou X, Wang X, Li J. A simple generic attack on text captchas. Netw Distrib Syst Secur Sympos (NDSS) 2016;1(February):21–4.
[16] Nguyen VD, Chow Y-W, Susilo W. On the security of text-based 3D CAPTCHAs. Comput Secur 2014;45:84–99.
[17] Nguyen VD, Chow Y-W, Susilo W. Attacking animated CAPTCHAs via character extraction. In: Pieprzyk J, Sadeghi A-R, Manulis M, editors. Proceedings of the 11th international conference cryptology and network security (cans 2012). Darmstadt, Germany: Springer Berlin Heidelberg; 2012, p. 98–113. http://dx.doi.org/10.1007/978-3-642-35404-5_9.
[18] Hernandez-Castro C, Ribagorda A, Hernandez-Castro J. On the strength of egglue and other logic captchas. In *International conference on security and cryptography (Secrypt 2011)*, Seville, Spain, 2011, p. 157–67.
[19] Gigoit. HumanAuth. Tech. rep., 2006.
[20] Hernandez-Castro CJ, Ribagorda A, Saez Y. Side-channel attack on the HumanAuth captcha. In *International conference on security and cryptography (secrypt 2010)*, Athens, Greece, 2010.
[21] Fritsch C, Netter M, Reisser A, Pernul G. Attacking image recognition CAPTCHAs. In: International conference on trust, privacy and security in digital business. Bilbao,Spain: Springer; 2010, p. 13–25.
[22] Golle P. Machine learning attacks against the asirra CAPTCHA. In: Proceedings of the 5th symposium on usable privacy and security, soups 2009. ACM international conference proceeding series, Mountain View, California, USA: ACM; 2009.
[23] Datta R, Li J, Wang JZ. Imagination: a robust image-based CAPTCHA generation system. In: Multimedia '05: Proceedings of the 13th annual acm international conference on multimedia. New York, NY, USA: ACM; 2005, p. 331–4. http://dx.doi.org/10.1145/1101149.1101218.
[24] Zhu BB, Yan J, Li Q, Yang C, Liu J, Xu N, Yi M, Cai K. Attacks and design of image recognition CAPTCHAs. In: Proceedings of the 17th acm conference on computer and communications security. Ccs '10, Chicago, Illinois, USA: ACM; 2010, p. 187–200, doi: http://doi.acm.org/10.1145/1866307.1866329.
[25] Gossweiler R, Kamvar M, Baluja S. What's up captcha?: A CAPTCHA based on image orientation. In: Proceedings of the 18th international conference on world wide web. Www '09, Madrid, Spain: ACM; 2009, p. 841–50. http://dx.doi.org/10.1145/1526709.1526822.
[26] Kim J, Kim S, Yang J, Ryu J-H, Wohn K. Facecaptcha: A CAPTCHA that identifies the gender of face images unrecognized by existing gender classifiers. Multimedia Tools Appl 2014;72(2):1215–37.
[27] Sim T, Nejati H, Chua J. Face recognition CAPTCHA made difficult. In: Proceedings of the 23rd international conference on world wide web. Www '14 companion, Seoul, Korea: ACM; 2014, p. 379–80. http://dx.doi.org/10.1145/2567948.2577321, URL http://doi.acm.org/10.1145/2567948.2577321.
[28] Gosschalk K, Ford M. Funcaptcha. Tech. rep., 2016.
[29] Schryen G, Wagner G, Schlegel A. Development of two novel face-recognition CAPTCHAs. Comput Secur 2016;60(C):95–116. http://dx.doi.org/10.1016/j.cose.2016.03.007.
[30] Goswami G, Powell BM, Vatsa M, Singh R, Noore A. FaceDCAPTCHA: Face detection based color image CAPTCHA. Future Gener Comput Syst 2014;31:59–68.
[31] Goswami G, Powell BM, Vatsa M, Singh R, Noore A. FR-CAPTCHA: CAPTCHA based on recognizing human faces. PLoS One 2014;9(4):e91708.
[32] Gao H, Lei L, Zhou X, Li J, Liu X. The robustness of face-based CAPTCHAs. In: 2015 IEEE international conference on computer and information technology; ubiquitous computing and communications; dependable, autonomic and secure computing; pervasive intelligence and computing. Liverpool, UK; 2015, p. 2248–55. http://dx.doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.332.
[33] Kim H, Tang J, Anderson R. Social authentication: harder than it looks. In: International conference on financial cryptography and data security. Bonaire, Netherlands: Springer; 2012, p. 1–15.
[34] Polakis I, Lancini M, Kontaxis G, Maggi F, Ioannidis S, Keromytis AD, Zanero S. All your face are belong to us: breaking facebook's social authentication. In: Proceedings of the 28th annual computer security applications conference. Orlando, Florida, USA: ACM; 2012, p. 399–408.
[35] Paxton T, Tatoris R. How PlayThru makes CAPTCHA obsolete. Tech. rep., 2012, URL http://areyouahuman.com/benefits/ (Accessed on 2012-09-26).
[36] Mohamed M, Sachdeva N, Georgescu M, Gao S, Saxena N, Zhang C, Kumaraguru P, van Oorschot PC, bang Chen W. Three-way dissection of a game-CAPTCHA: Automated attacks, relay attacks, and usability. 2013, ArXiv Preprint abs/1310.1540.
[37] Mitra NJ, Chu H-K, Lee T-Y, Wolf L, Yeshurun H, Cohen-Or D. Emerging images. ACM Trans Graph 2009;28(5):163:1–8.
[38] NuCaptcha. Nucaptcha security feautures. Tech. rep., 2016, URL http://www.nucaptcha.com/security-features (Accessed on 2014-11-20).
[39] Kluever KA. Evaluating the usability and security of a video captcha (Master's thesis), Rochester Institute of Technology; 2008.
[40] Hernandez-Castro CJ, Ribagorda A. Video CAPTCHAs. In: IDET Security Conference - Security and Protection of Information (SPIE). Brno, Czech Republic; 2009.
[41] Qvarfordt P, Rieffel EG, Hilbert DM. Motion and interaction based CAPTCHA. (US Patent 8,601,538). 2013.
[42] Santamarta R. Breaking gmail's audio captcha. Tech. rep., 2008, URL http://blog.wintercore.com/?p=11 (Accessed on 2010-13-02).
[43] Tam J, Simsa J, Hyde S, von Ahn L. Breaking audio CAPTCHAs. In: Advances in neural information processing systems (nips). Vancouver, British Columbia, Canada: Curran Associates, Inc.; 2008, p. 1625–32.
[44] Sano S, Otsuka T, Okuno HG. Solving google's continuous audio CAPTCHA with HMM-based automatic speech recognition. In: Sakiyama K, Terada M, editors. Advances in information and computer security: proceedings of the 8th international workshop on security, iwsec 2013. Okinawa, Japan: Springer Berlin Heidelberg; 2013, p. 36–52.
[45] Sidorov Z. Rebreakcaptcha: Breaking google's recaptcha v2 using google. Tech. rep., 2017, (Accessed on 2017-08-14).
[46] Bursztein E. How we broke the nucaptcha video scheme and what we propose to fix it. Tech. rep., Google Anti-abuse Research Team; 2012.

[47] Xu Y, Reynaga G, Chiasson S, Frahm J-M, Monrose F, Van Oorschot P. Security and usability challenges of moving-object CAPTCHAs: decoding codewords in motion. In: Presented As Part of the 21st USENIX Security Symposium (USENIX Security 12). Bellevue, WA, USA: USENIX; 2012, p. 49–64.

[48] Sivakorn S, Polakis I, Keromytis AD. I am robot:(deep) learning to break semantic image CAPTCHAs. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P). Saarbrücken, Germany: IEEE; 2016, p. 388–403.

[49] Homakov E. The no CAPTCHA problem. Tech. rep., 2014, URL http://homakov.blogspot.com.es/2014/12/the-no-captcha-problem.html (Accessed on 2017-08-16).

[50] Hernández-Castro CJ, Barrero DF, R-Moreno MD. Machine learning and empathy: The civil rights CAPTCHA. Concurr Comput Pract Exp 2016;28(4):1310–23. http://dx.doi.org/10.1002/cpe.3632.

[51] Hernández-Castro CJ, R-moreno MD, Barrero DF. Side-channel attack against the capy HIP. In: Fifth international conference on emerging security technologies (est 2014). 2014.

[52] Seber GAF. The Estimation of Animal Abundance. 16, (1):Griffin London; 1974, p. 80–5,

[53] Hernandez-Castro CJ. Where do CAPTCHAs fail: A study in common pitfalls in CAPTCHA design and how to avoid them. (Ph.D. thesis), Madrid, Spain: Universidad de Alcala; 2017.

[54] Fan R-E, Chang K-W, Hsieh C-J, Wang X-R, Lin C-J. Liblinear: A library for large linear classification. J Mach Learn Res 2008;9:1871–4.