All About Uncertainties and Traps: Statistical Oracle-based Attacks on a New CAPTCHA Protection Against Oracle Attacks[⋆]

Carlos Javier Hernández-Castro[a,*], Shujun Li[b,*], María D. R-Moreno[a]

[a] *Departamento de Automática, Universidad de Alcalá, Madrid, Spain*
[b] *School of Computing & Kent Interdisciplinary Research Centre in Cyber Security (KirCCS), University of Kent, Canterbury, UK*

Abstract

CAPTCHAs are security mechanisms that try to prevent automated abuse of computer services. Many CAPTCHAs have been proposed but most have known security flaws against advanced attacks. In order to avoid a kind of oracle attacks in which the attacker learns about ground truth labels via active interactions with the CAPTCHA service as an oracle, Kwon and Cha proposed a new CAPTCHA scheme that employ uncertainties and trap images to generate *adaptive* CAPTCHA challenges, which we call "Uncertainty and Trap Strengthened CAPTCHA" (UTS-CAPTCHA) in this paper. Adaptive CAPTCHA challenges are used widely (either explicitly or implicitly) but the role of such adaptive mechanisms in the security of CAPTCHAs has received little attention from researchers.

In this paper we present a statistical fundamental design flaw of UTS-CAPTCHA. This flaw leaks information regarding ground truth labels of images used. Exploiting this flaw, an attacker can use the UTS-CAPTCHA

[⋆] This author's accepted version is released under a Creative Commons CC BY NC ND license. The full citation information of the published version is: Carlos Javier Hernández-Castro, Shujun Li and María D. R-Morenoa, "All About Uncertainties and Traps: Statistical Oracle-based Attacks on a New CAPTCHA Protection Against Oracle Attacks," *Computers & Security*, vol. 92, Article Number 101758, 2020, DOI: 10.1016/j.cose.2020.101758. © 2020 Elsevier B.V.

[*] Corresponding co-authors with equivalent contributions.

Email addresses: chernandez@ucm.es (Carlos Javier Hernández-Castro), malola.rmoreno@uah.es (María D. R-Moreno)

URL: http://www.hooklee.com/ (Shujun Li)

service as an oracle, and perform several different statistical learning-based attacks against UTS-CAPTCHA, increasing any reasonable initial success rate up to 100% according to our theoretical estimation and experimental simulations. Based on our proposed attacks, we discuss how the fundamental idea behind our attacks may be generalized to attack other CAPTCHA schemes and propose a new principle and a number of concrete guidelines for designing new CAPTCHA schemes in the future.

Keywords: CAPTCHA, uncertainty, trap images, machine learning, image classification, oracle attacks, statistical attacks

1. Introduction

Online services have become prevalent since the broad deployment of the Internet since the late 90's. The abuse of such services, using automated computer programs (called bots), can be the first step towards more sophisticated attacks that can result in significant revenue for the attackers. Naor [1] was the first to propose a theoretical security framework based on the idea of discerning humans from bots using problems that could be solved easily by humans but were thought to be hard for computers. In the early 2000s Ahn et al. [2] coined the term CAPTCHA – "Completely Automated Public Turing test to tell Computers and Humans Apart", which since then has been widely used to refer to technologies preventing automated online resource abuses. Ahn et al. also offered some suggestions for such problems: gender recognition, understanding facial expressions, filling in words in incomplete sentences, etc. Some of such suggestions were subsequently used to create CAPTCHAs for real-world systems, while other researchers developed alternative designs. The general principle behind CAPTCHAs is to use theoretically hard-AI problems as the basis of generate challenges that can only be solved by humans but not computers alone [3–5].

Many CAPTCHAs are based on image classification problems. Typically, they require the user to tell which images from a set pertain to a particular category [6]. Many CAPTCHAs of this kind have been analyzed and broken [7–10], in some cases using Deep Learning (DL). Some image-based CAPTCHAs are based on the more general problem of image or shape recognition in different contexts, often involving active user interaction in order to enhance security [11–13], however, many of such schemes have also been shown not sufficiently secure [14–16]. Other image-based CAPTCHA proposals are not based on images classification. Among these, some interesting proposals are puzzle CAPTCHAs (such as Capy CAPTCHA [17] or Key CAPTCHA [15]), Emerging Images [11] or CaptchaStar [13].

By nature all classification-based CAPTCHA schemes are susceptible to learning-based attacks, in which the CAPTCHA is used as an *oracle*. To perform such an attack, an attacker only needs a bot based on a random or a very weak classifier that can successfully pass a small fraction of the challenges, even if mostly by chance. Once a challenge is successfully solved, the bot can learn about the correct classifications of the images in that challenge. This way, the bot can *learn* incrementally based on such learned examples and (potentially dramatically) increase its success rate over time. In addition, a bot may also be able to learn from failures because failing to pass a specific CAPTCHA challenge reveals some useful information about the ground truth label of the challenge: the guessed response is not the ground truth label. If the number of possible responses is limited and the same image can be repeatedly observed, the ground truth label can be gradually revealed. Even if the ground truth label is not revealed, having a reduced set of candidate labels can also help the bot to train its classifier. Such learning-based attacks are particular relevance for the security of CAPTCHAs because of the rapid development of AI technologies especially those based on DL architectures such as DCNNs (deep convolutional neural networks) for image classification tasks [9, 18–22]. It is true that these DL-based attacks have some limitations [23, 24]. Some researchers have proposed to use some of these limitations as a base for new image-based CAPTCHAs [25]. But as of now, DL-based classifiers remain a serious threat to most image classification CAPTCHAs.

Kwon and Cha [26] proposed a new approach to preventing bots from using the CAPTCHA service as an oracle. To do so, they employ two mechanisms. First, they add uncertainties to the grading function: when the CAPTCHA service grades an answer to a challenge, it excludes a subset of so-called "neutral" images in the challenge into consideration. Second, they use what they call "trap images" to detect "all bots". Trap images are *adaptively* decided for each *specific* suspected bot if a neutral image is used but the suspected bot passed a challenge (i.e., the neutral image could have prevented the bot from passing if it was not a neutral image). Using these two mechanisms, they argued that any image-based CAPTCHA can be strengthened into what we will call an "Uncertainty and Trap Strengthened CAPTCHA" (UTS-CAPTCHA).[1] Adaptive CAPTCHA challenges are also used in other CAPTCHA schemes and the need to resist learning-based oracle attack actually requires all CAPTCHA schemes to adaptively evolve. However, the role of such adaptive mechanisms in CAPTCHA designs has been much less studied.

In this paper, we present different statistical attacks on UTS-CAPTCHA. Our attacks all exploit a fundamental weakness in the UTS-CAPTCHA design as proposed by Kwon and Cha. This weakness enables us to perform learning attacks on UTS-CAPTCHAs, rendering the proposed protection useless. We also study in detail the results from some of these attacks.

The rest of the paper is organized as follows. In Section 2, we provide a more detailed description of UTS-CAPTCHA. In Section 3 we explain an exploitable statistical flaw in UTS-CAPTCHA and a number of statistical learning-based attacks based on this flaw, which can use the UTS-CAPTCHA service as an oracle to incrementally learn about more and more ground truth labels of images. Then, the experimental results of some of the proposed attacks are presented in Section 4. Section 5 discusses some possible improvements to UTS-CAPTCHA. In Section 6, we look at how the fundamental idea behind the proposed ad hoc attacks against UTS-CAPTCHA can be generalized, leading to a new design principle and some concrete guidelines for designing future CAPTCHA schemes. The final section concludes the paper.

2. UTS-CAPTCHA

The UTS-CAPTCHA is built based on two sets of images: M – a set of images labeled by "M" ("M"ust be chosen because they are from a particular image class, e.g. images about a specific celebrity like Bill Gates), and MN – a set of images labeled by "MN" ("M"ust "N"ot be chosen as they are not from the class). To pass a given CAPTCHA challenge with c images including at least one "M" and at least one "MN" images, the user must correctly choose all "M" images but not any "MN" images. Without any other changes, such a CAPTCHA is effectively a simple one-class image classification based CAPTCHA, which is clearly vulnerable to learning based oracle attacks since

[1] Kwon and Cha [26] did not use the term UTS-CAPTCHA. We coined this term to avoid keeping using the longer name "Kwon and Cha's CAPTCHA scheme".

any accidental success will reveal true labels of all images in the corresponding challenge.

To improve security against learning based oracle attacks, Kwon and Cha proposed the following two mechanisms to strengthen the above base-line CAPTCHA.

1) *Uncertainty-based grading:* for each challenge, n images are randomly selected to form NI – a set of "neutral images" which are not considered in the grading, where n is a random number between 0 and $n_{\max}$. A challenge is passed if the user answers correctly to all $c-n$ non-neutral images, regardless how it answers to any of the n neutral images. Some *uncertainties* are introduced by the random neutral images thus can help prevent bots from learning about true labels from accidental successes.

2) *Trap images:* the UTS-CAPTCHA service also maintains a *user-specific* database of "trap images", denoted by TI_u hereinafter, where each user u denotes a human user or a bot with the same behavior. A "trap image" is an "M" or "MN" image in a CAPTCHA challenge that was incorrectly answered but is selected as an "neutral" image to allow the user/bot to pass this challenge. In other words, a trap image is a neutral image that could have prevented a bot from passing the CAPTCHA if it was not used as a neutral image. Such images are labeled as trap images and are used to prevent the specific bot (which made the wrong response before) in the future. The idea behind trap images is that once a bot has mis-classified an image for a passed challenge it will consider the classification confirmed and thus keeps repeating the same error consistently in future attempts while human users do not normally repeat accidental errors. TI_u is initially empty but will be gradually filled up when more and more trap images are observed for a user or bot. Once TI_u has at least one trap image, t trap images will be included in *each* future challenge and always be used for grading (i.e., never selected as neutral images again), where t is a random number between 1 and $\min(t_{\max}, |\mathrm{TI}_u|)$.

Since trap images are user-specific, the UTS-CAPTCHA service needs to be able to identify each user or bot and build a different set of trap images for it. To this end, Kwon and Cha [26] discussed using IP addresses and behavioral analysis. They did not explicitly explain what to do if a "user" (which may be a bot) correctly answers an existing trap image in a passed challenge. According to the nature of trap images, we make the following reasonable assumption: when that happens the UTS-CAPTCHA service considers the "user" as a human who made an accidental mistake in a past successful challenge and removes this particular trap image from TI. This assumption is meaningful since a trap image is not a trap any more if a bot can now answer it correctly. Even if the UTS-CAPTCHA service does not remove any trap images, our proposed attacks will still work in principle, although they will become less efficient (i.e., slower). This is because unknown trap images will appear with a smaller probability while TI becomes larger and more trap images are detected. Some simple tricks can be applied to solve this problem. For instance, when the efficiency drops to a specific level, a bot can simply switch to a completely different environment (different IP address, different web browser, etc.) to trigger an empty trap image database.

Kwon and Cha did not explain from when trap images start being used. We assume that this starts once there is at least one trap image in TI according to indirect evidence in their experimental results. It may be a better setting to start using trap images after TI has at least $t_{\max}$ trap images. This setting does not however influence our proposed attacks in any significant way.

It deserves noting that Kwon and Cha actually did not explicitly mention if trap images included in a challenge are always graded. When discussing their experimental results they however said (on Page 83) "Our system never missed the robots' mistakes, and later challenges included one or two such images to defeat nearly all the robot attacks." The words "never" and "all" imply that trap images must be always graded. This assumption does not have a significant influence on the effectiveness of our proposed attack.

In their prototype system of UTS-CAPTCHA, Kwon and Cha manually verified 12,388 images from the Internet, including 4,033 "M" images of Bill Gates and 8,355 "MN" images. They set the parameters as follows: $c = 22$, $n_{\max} = 8$ and $t_{\max} = 2$. They conducted some experiments to support their claims about the security of UTS-CAPTCHA.

3. Proposed attacks

In this section we propose a family of oracle-based attacks on UTS-CAPTCHA which are all built on an observable statistical difference, which has its root in the design of the UTS-CAPTCHA scheme. Such statistical attacks have been used by other researchers to break some observer-resistant password systems [27], but as far as we know this is its first application to breaking CAPTCHAs.

Before introducing the attacks, let us model a typical image classification based CAPTCHA as iCAPTCHA $= (\mathrm{M}, \mathrm{MN}, \mathcal{A}, \mathcal{D})$, where M is a set of images that pertains to a specific class, MN is a set of images that are not from the class, $\mathcal{A}$ is an algorithm generating $r \geq 1$ CAPTCHA challenge(s) $\{C_1, \ldots, C_i, \ldots, C_r\}$ when executed, and $\mathcal{D}$ is another algorithm making a binary decision ("reject" or "accept") based on responses to the r CAPTCHA challenge(s). For UTS-CAPTCHA, $r = 1$. The model does not cover trap images used in UTS-CAPTCHA explicitly, but such details can be considered as part of the algorithms $\mathcal{A}$ and $\mathcal{D}$. Without loss of generality, we assume that each image in the trap image database TI and in the rest of the set $\mathrm{M} \cup \mathrm{MN}$ is sampled independently with equal probability without repeat, considering the constraints defined by the parameters $n_{\max}$ and $t_{\max}$. Although the model is rather simple, it can cover typical settings of UTS-CAPTCHA well. It can also be easily extended to cover more complicated cases, e.g., when multiple image classes are used and challenges do not have a fixed size. The simplicity of our model does not affect the generalizability of our proposed attack, either.

3.1. The statistical difference

It seems obvious that in UTS-CAPTCHA, and as long as the intended purpose of detecting bots works, the size of TI will be much smaller than the size of the rest of image database $(\mathrm{M} \cup \mathrm{MN}) - \mathrm{TI}$. This is so because, ideally, following [26], TI will be filled once and only once per each bot since the trap images will prevent the bot from passing any future challenges.

This fact clearly presents a security problem if a trap image in TI appears in a challenge with a probability different from a non-trap image in $(\mathrm{M} \cup \mathrm{MN}) - \mathrm{TI}$, i.e., if the following inequality holds:

$$\frac{\bar{t}}{|\mathrm{TI}|} \neq \frac{c - \bar{t}}{|(\mathrm{M} \cup \mathrm{MN}) - \mathrm{TI}|}, \tag{1}$$

where $\bar{t}$ is the mean of t over all challenges and will be $(1 + t_{\max})/2$ if t is sampled uniformly from $\{1, \ldots, t_{\max}\}$. The above equation can also be interpreted in a simpler way by comparing the occurrence probability of a trap image, denoted by p_t, with the *natural* occurrence probability of an image in the whole image database $\mathrm{M} \cup \mathrm{MN}$ with uniform sampling, denoted by p:

$$p_t = \frac{\bar{t}}{|\mathrm{TI}|} \neq p = \frac{c}{|\mathrm{M} \cup \mathrm{MN}|}. \tag{2}$$

This probability difference implies that for any challenges appearing after at least one trap image is added into TI, any trap image from TI will repeat with a different probability from the rest of images. Looking at the parameters used by Kwon and Cha in their experiments, we can calculate

$$p_t = \frac{\bar{t}}{|\mathrm{TI}|} = \frac{(1+t_{\max})/2}{|\mathrm{TI}|} = \frac{1.5}{|\mathrm{TI}|} \tag{3}$$

and

$$p = \frac{c}{|\mathrm{M} \cup \mathrm{MN}|} = \frac{22}{12388} \approx 0.001776. \tag{4}$$

If one wants to make $p_t = p$, then

$$\frac{1.5}{|\mathrm{TI}|} = \frac{22}{12388} \to |\mathrm{TI}| = \frac{1.5 \times 12388}{22} \approx 844.6364. \tag{5}$$

Since $|\mathrm{TI}|$ (the number of trap images) must be an integer, the above equation will never hold. Therefore, the calculation suggests that for the parameters chosen by Kwon and Cha, it is actually *impossible* to achieve $p_t = p$ since $|\mathrm{TI}|$ can only be an integer therefore cannot be equal to 844.6364. Considering $|\mathrm{TI}|$ is always much smaller than 844.6364, we can see $p_t \gg p$, meaning that any trap image will appear in a challenge with a *much* higher probability than a normal image. Since $|\mathrm{TI}| = 0$ initially and $|\mathrm{TI}| \leq n_{\max}$ after the first time some trap images are added, we can calculate a lower bound of p_t as follows:

$$p_t \geq \frac{1.5}{n_{\max}} = \frac{1.5}{8} = 0.1875, \tag{6}$$

which is more than 105 times of $p \approx 0.001776$. Assuming that TI will remain largely unchanged after the first time some trap images are added (as we discussed above), the value of p_t will remain unchanged.

Note that Kwon and Cha actually applied two constraints on how images in M and MN are sampled for a CAPTCHA challenge: at least one from M and at least one from MN. This will lead to a change of the value of p depending on if the corresponding image comes from M or MN:

$$p^{(\mathrm{M})} = 1 - \frac{(|\mathrm{M}|-1) \cdot |\mathrm{MN}| \cdot P(|\mathrm{M} \cup \mathrm{MN}|-3, c-2)}{|\mathrm{M}| \cdot |\mathrm{MN}| \cdot P(|\mathrm{M} \cup \mathrm{MN}|-2, c-2)}, \tag{7}$$

$$p^{(\mathrm{MN})} = 1 - \frac{|\mathrm{M}| \cdot (|\mathrm{MN}|-1) \cdot P(|\mathrm{M} \cup \mathrm{MN}|-3, c-2)}{|\mathrm{M}| \cdot |\mathrm{MN}| \cdot P(|\mathrm{M} \cup \mathrm{MN}|-2, c-2)}, \tag{8}$$

where

$$P(i,j) = \frac{i!}{(i-j)!} = \prod_{k=0}^{j-1}(i-k) = i\cdot(i-1)\cdots(i-j+1). \tag{9}$$

It is impossible to numerically estimate the above values for large M and MN, but we can calculate their upper bounds to be the maximum between the probability of being the minimum one image selected from M or MN and the probability of being selected naturally otherwise:

$$p^{(\mathrm{M})} \leq \max\left(\frac{1}{|\mathrm{M}|}, \frac{|\mathrm{M}|-1}{|\mathrm{M}|}\cdot p\right), \tag{10}$$

$$p^{(\mathrm{MN})} \leq \max\left(\frac{1}{|\mathrm{MN}|}, \frac{|\mathrm{MN}|-1}{|\mathrm{MN}|}\cdot p\right). \tag{11}$$

When $p \geq \max\left(\frac{1}{|\mathrm{M}|-1}, \frac{1}{|\mathrm{MN}|-1}\right)$ (which also implies M and MN are not too small), the above two probabilities will be both very close to p. This is clearly the case for the parameters used by Kwon and Cha:

$$p \approx 0.0018 > \max\left(\frac{1}{|\mathrm{M}|-1}, \frac{1}{|\mathrm{MN}|-1}\right) \approx 0.00025. \tag{12}$$

In general p can always be used as a (good) rough estimate of $p^{(\mathrm{M})}$ and $p^{(\mathrm{MN})}$ because M and MN have to contain sufficiently many images to make the UTS-CAPTCHA practically useful (with too few images an attack will become trivial – one can just collect all images and classify them manually and then build a simple bot based on the true labels).

Since the sizes of M and MN both have to be sufficiently large to make UTS-CAPTCHA secure and c has to be sufficiently small to make UTS-CAPTCHA usable, p will be normally quite small. However, both $t_{\max}$ and $n_{\max}$ obviously have to be significantly smaller than c, so p_t will be normally quite large with $2/c$ as its lower bound if at least one trap image being used for each challenge and no more than half of images in a challenge being selected as neutral. As a whole, we believe that for all practical settings $p_t \gg p$ always holds.

Although the above calculation and discussions are based on an ad hoc design and parameters of the UTS-CAPTCHA scheme and some assumptions such as uniform sampling, the basic idea can be easily generalized. The main reason why $p_t \gg p$ appears in UTS-CAPTCHA is that trap images by definition have to be learnt from the behavior of each target bot, which means that the number of such trap images will be small initially. On the other hand, the number of the whole image pool has to be large to reduce the chance of random guess attacks. Therefore, when uniform sampling is applied $p_t \gg p$ will always hold until the point that trap images are as many as normal ones. If images are not sampled uniformly, the system has to decide what images to be sampled with a higher probability, which can only lead to more exploitable statistical differences for attackers. Even if uniform sampling can work without directly compromising security, as long as trap and normal images are sampled using two different probability density functions such statistical differences can still be inferred and exploited in a similar (although more complicated) manner. Since the probability density functions defined for two sets of different sizes will always be different, so as long as $|\mathrm{TI}| \neq |\mathrm{M} \cup \mathrm{MN}|$ detection of some such statistical differences should be always theoretically possible. Note that a human attacker can obtain probability density functions corresponding to human users (by solving CAPTCHA challenges himself or asking recruited human solvers to do so) and different bots (by training different kinds of image classifiers), so he can proactively probe the system to accumulate more information until some exploitable statistical differences are detected.

3.2. Statistical oracle-based attacks

The fact that $p_t \gg p$ holds immediately leads to a *statistical* oracle-based approach to launching an attack: once a bot passes one challenge C_i successfully by accident, it can start observing all future challenges to identify trap images which appear with a higher probability than normal images. For each trap image the bot detects, it can easily get the true label of the image by reversing the answer the bot gave for the challenge C_i. Such true labels can be used to help improve the performance of the image classifier and to increase the chance of passing future challenges. By keeping doing this, the bot can incrementally increase its success rate even up to 100%.

For a bot with an initial low image classification accuracy, it will need to request many challenges to the UTS-CAPTCHA service in order to eventually learn about enough trap images to improve its success rate. To avoid being detected the bot can try to issue such requests from multiple IP addresses. Assuming the bot can control a botnet with millions of zombie computers, it will not be difficult to launch an attack requesting millions of

CAPTCHAs in a short period of time. This makes it possible to circumvent any throttling mechanisms, so in the following we assume that requesting a large number of CAPTCHA challenges is not an issue. Launching the attack from multiple machines can also help the bot as the UTS-CAPTCHA service will create smaller trap image databases for each instance of attack which makes p_t large. While the attack is going, the fact that the bot can successfully answer to some trap images will let UTS-CAPTCHA service to remove those trap images from the database, thus keeping p_t large.

The attack idea we described above can actually be launched in several different ways. In the following, we describe a number of attacks that all follow the same idea but employ different technical approaches to detecting the statistical difference between p_t and p.

3.2.1. Attack 1: Unsupervised clustering based attack

Since trap images appear much more frequently than normal images, one straightforward approach to detecting them is to observe occurrence frequencies (or counts) of all images and then use an unsupervised clustering algorithm such as k-means [28] to classify all images into two classes: trap images around a high-frequency cluster and normal images around a low-frequency cluster. The main merit of this attack is its being non-parametric thus does not depend on estimation of system parameters other than the fact that trap images appear more frequently than normal images. It is expected that this attack can work with a small number of observed challenges due to the big difference between the two occurrence probabilities p and p_t. This attack can also be used to estimate p and p_t since the cluster centroids should be close to these two probabilities. We will discuss the estimation of system parameters later as they are needed in some other attacks.

3.2.2. Attack 2: Conditional probability based attack

This attack is based on calculation of the conditional probability of each candidate image x_i being a trap image based on its n occurrences in N observed challenges. This probability can be calculated based on Bayes' theorem if a number of key system parameters are known or can be roughly estimated. In a practical attack, it is sufficient to estimate a lower bound of the probability, which can be translated into estimation of a lower of upper bound of other system parameters. The parameters needed include p, p_t, $|\mathrm{M} \cup \mathrm{MN}|$ and $|\mathrm{TI}|$. With those parameters and assuming $\theta(x_i, N)$ denotes the number of occurrence of x_i in N observed challenges, the conditional probability can be calculated as follows:

$$\Pr\left[x_i \in \mathrm{TI} | \theta(x_i, N) = n\right] = \frac{P_{N,n,\mathrm{TI}}}{P_{N,n,\mathrm{TI}} + P_{N,n,\overline{\mathrm{TI}}}} \tag{13}$$

where

$$\begin{aligned} P_{N,n,\mathrm{TI}} &= \Pr\left[\theta(x_i, N) = n | x_i \in \mathrm{TI}\right] \Pr\left[x_i \in \mathrm{TI}\right] \\ &= \binom{N}{n} p_t^n (1 - p_t)^{N-n} \cdot \frac{|\mathrm{TI}|}{|\mathrm{M} \cup \mathrm{MN}|} \end{aligned} \tag{14}$$

and

$$\begin{aligned} P_{N,n,\overline{\mathrm{TI}}} &= \Pr\left[\theta(x_i, N) = n | x_i \notin \mathrm{TI}\right] \Pr\left[x_i \notin \mathrm{TI}\right] \\ &= \binom{N}{n} p^n (1 - p)^{N-n} \cdot \frac{|\mathrm{M} \cup \mathrm{MN}| - |\mathrm{TI}|}{|\mathrm{M} \cup \mathrm{MN}|}. \end{aligned} \tag{15}$$

Substituting Eqs. (14) and (15) into Eq. (13) and dividing the denominator by the enumerator, we can get

$$\Pr\left[x_i \in \mathrm{TI} | \theta(x_i, N) = n\right] = \frac{1}{1 + \left(\frac{p}{p_t}\right)^n \left(\frac{1-p}{1-p_t}\right)^{N-n} \left(\frac{|\mathrm{M} \cup \mathrm{MN}|}{|\mathrm{TI}|} - 1\right)}. \tag{16}$$

From $|\mathrm{TI}| \geq 1$, we can get a lower bound of $\Pr\left[x_i \in \mathrm{TI} | \theta(x_i, N) = n\right]$ as follows ($|\mathrm{TI}| \geq 1$ is not a too aggressive relaxation since TI is normally not very large):

$$\begin{aligned} \Pr\left[x_i \in \mathrm{TI} | \theta(x_i, N) = n\right] &\geq \frac{1}{1 + \left(\frac{p}{p_t}\right)^n \left(\frac{1-p}{1-p_t}\right)^{N-n} (|\mathrm{M} \cup \mathrm{MN}| - 1)} \\ &= \frac{1}{1 + \left(\frac{p}{p_t}\right)^n \left(\frac{1-p}{1-p_t}\right)^{N-n} \left(\frac{c}{p} - 1\right)}. \end{aligned} \tag{17}$$

The above lower bound can be calculated based on estimation of p and p_t only.

3.2.3. Attack 3: Simple threshold based attack

This attack is also based on counting the number of occurrences of each image in M and MN over N challenges. It labels all images whose counts are high enough than normal images as trap images. To be more precise, the detection process is about determining a threshold T between Np and Np_t so that any image occurs more than T times will be labeled as a trap image. The threshold T can be set to keep the precision very high so that false positives are minimized. False negatives are less an issue here since any missed trap images may be detected later while more challenges are observed. To determine the desired threshold, one can calculate the false negative and false positive rates from the two related binomial distributions (one for trap images and the other for normal images). When N is large enough, the calculation can be done by using the normal distributions $\mathcal{N}(Np, Np(1-p))$ and $\mathcal{N}(Np_t, Np_t(1-p_t))$ to approximate the two binomial distributions. It is expected with a large N and a proper threshold, the false positive and false negative rates can be made sufficiently small. This attack also requires estimation of p and p_t, although it is possible to set a more conservative threshold based on an estimate of p_t only and the fact that $p_t \gg p$.

3.2.4. Attack 4: Parallel χ^2 tests based attack

This attack is about running a number of parallel Pearson's χ^2 tests with the degree of freedom being 1 each on one candidate image x_i, where the random variable X_i is defined as a binary variable representing if x_i presents in a random challenge C ("present" / 1 or "absent" / 0) and the target distribution is a Bernoulli distribution $\{p, 1-p\}$. Given N observed challenges, the bot can see N samples of the random variable X_i for each image x_i and calculate the χ^2 statistic and then the corresponding p-value. An image is considered trap image if the p-value drops below a threshold (i.e., the significance level which is commonly set to be 0.05 or 0.01) as N increases. Note that a Bonferroni correction should be applied to the significance level because there are $n \leq |\mathrm{M} \cup \mathrm{MN}|$ parallel tests, which means that the actual significance level should be divided by n (thus being much smaller to reduce Type I errors – images which are normal but detected as trap images). This approach has its merit of having a threshold that is easier to determine: it is basically the standard significance level used in all statistical tests, while in Attack 2 the threshold depends on N so has to be re-calculated for each value of N and it is not always straightforward to decide how to balance false positive and false negative rates.

3.2.5. Attack 5: Sequential $\mathcal{X}^2$ tests based attack

This attack is also based on a number of Pearson's $\mathcal{X}^2$ tests but it does not require knowledge about the expected distribution, i.e., the (at least approximate) value of p. As discussed in Section 3.1 (see Eqs. (7) and (8)) each normal (non-trap) image appears in a challenge with a probability close to p and each trap image appears with a probability $p_t \gg p$. This suggests that given N challenges the occurrence frequencies of all images will approximately follow a uniform distribution when there are no any trap images, but will deviate significantly from it if there is at least one trap image. Since any segment of a uniform distribution is still a uniform distribution, we can apply a Pearson's $\mathcal{X}^2$ test to the occurrence frequencies of *any* subset of images (e.g., only those with larger occurrence frequencies to avoid mistaking some normal images as trap images). Since the uniform distribution does not have any parameters, applying the Pearson's $\mathcal{X}^2$ test this way does not require *any* knowledge of any system parameters. To run the Pearson's $\mathcal{X}^2$ test more effectively, N needs to be sufficiently large so that any trap image will appear significantly more than any normal image, for which one only needs to know a conservative lower bound of p_t (although knowing the precise value will help). Existence of trap images can be detected if the Pearson's $\mathcal{X}^2$ test gives a very small p-value, and then the images with maximum frequencies can be considered trap images. This process can be repeated until the Pearson's $\mathcal{X}^2$ test cannot reject the null hypothesis. In this attack, no Bonferroni correction is needed since the Pearson's $\mathcal{X}^2$ tests are run sequentially (one test at one time, not many in parallel at the same time). Some false positives may still be produced, which can be detected by observing the occurrence frequencies of all detected trap images in future challenges. Wrongly labelled trap images with low occurrence frequencies can then be detected and removed accordingly.

3.3. Estimation of system parameters

All the above attacks either require or can benefit from precise or at least rough knowledge of some system parameters. Although such parameters are not normally part of the security setting so could be assumed public knowledge, in real-world scenarios the UTS-CAPTCHA service has no motivation to publicize such parameters so an attacker can use them to refine the attack. In the following, we discuss how to estimate the parameters p and p_t separately. Note that another system parameter $|\mathrm{M} \cup \mathrm{MN}|$ can be derived from p: $|\mathrm{M} \cup \mathrm{MN}| = c/p$.

To estimate p, one simple method is to collect N observed challenges and use the *mean* occurrence frequency of all images observed as an approximate of p. In order to avoid trap images' much higher occurrence frequency skews the estimate, we can exclude all images appearing in at least one passed challenge so that all possible trap images are excluded. Note that excluding a small number of normal images should not influence the mean as all images are sampled independently with equal probability. We expect this method will require $O(1/p)$ observed challenges to give an accurate estimate. To reduce the number of challenges, another method can be used to derive an approximate estimate of p, based on counting the number of *unique* images appearing in N challenges. Given the fact that each normal image has an occurrence probability p in each challenge, the probability that it appears at least once in N challenges will be $p^{(N*)} = 1 - (1-p)^N$.[2] Then, the mean number of unique normal images observed in N challenges can be estimated to be[3]

$$\overline{N_{\text{normal}}} = |\mathrm{M} \cup \mathrm{MN}| \cdot p^{(N*)} = \frac{c\left(1-(1-p)^N\right)}{p} \tag{18}$$

The above relationship between p and $\overline{N_{\text{normal}}}$ is largely a monotonic function for $0 \leq p \leq 1$ when $c = 22$ (the value used by Kwon and Cha) as shown in Figure 1.

Therefore, with N_{normal} normal images observed in N challenges, we can get the approximate value of p by (numerically) solving Eq. (18). Here, note that we cannot actually observe $\overline{N_{\text{normal}}}$, so the actual observed number N_{normal} is used instead. To get $\overline{N_{\text{normal}}}$ more accurately, one can observe a number of independent groups of N challenges, and then calculate the average of all observed values of N_{normal}. The precise value of N_{normal} cannot be actually observed because of possible existence of one or more trap images. Fortunately, using the number of observed images as N_{normal} will just overestimate N_{normal} *slightly* because the number of trap images should always be much smaller than N_{normal}.

p_t can be estimated by observing images appearing with a probability

[2] We ignore the small difference between the probability of "M" images and that of "MN" images (see the discussion in Section 3.1).

[3] Here, we make some further simplifications, e.g., ignoring the fact that in each challenge no image repeats itself, but we aim at giving a rough estimate so the precision is not a major concern. It is actually possible to derive a more accurate (and more complicated) formula, but we consider it unnecessary for our purpose here.

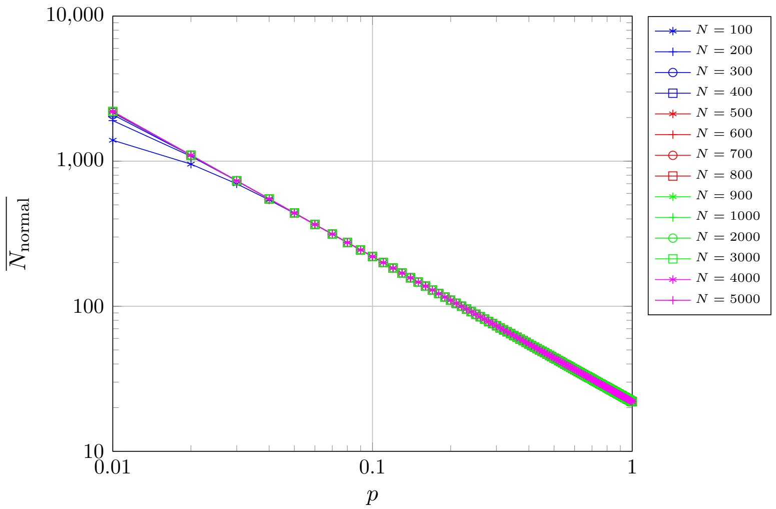


Figure 1: The relationship between p and $\overline{N_{\text{normal}}}$ for different values of N, when $c = 22$.

significantly higher than p. This can be done using a simple k-means clustering algorithm with $k = 2$ clusters, where one cluster represents normal images and the other represents trap images. Since the first cluster is significantly larger than the second, we can focus on images appearing in passed challenges only. When the precise value of p_t is not required, an alternative approach is to simply use the maximum frequency of all observed images. The maximum frequency can also be used as the starting point of the second cluster in the k-means clustering algorithm to speed it up. The starting point of the first cluster can be set by an estimated value of p. Since p_t is normally very large, it is expected that it can be estimated fairly fast within a small number of observed challenges roughly at the order of $O(1/p_t)$. Note that the process of estimating p_t using a k-means algorithm basically follows the same principle as Attack 1. Once we have an estimate of p_t, it is also possible to estimate $|\text{TI}|$ or $t_{\max}$ if either is known since the actual value of p_t should be a rational number with $|\text{TI}|$ as its denominator and $1 + \min(|\text{TI}|, t_{\max})/2$ as its numerator.

3.4. From 1-D to n-D attacks

The attack proposed above works with the probability difference of a single image x_i. Nothing prevents us from generalizing the attack to the more general n-D case: we can now look at the joint probability of an n-image tuple $(x_{i1}, \ldots, x_{in})$. While this does not seem to make much sense to make the attack more complicated, the generalization can make defense much harder as in higher-dimensional cases there are much more potential probability differences one has to consider. Since the dimensionality n is bounded only by $|\mathrm{M}\cup\mathrm{MN}|$, there are a very large number of potential attacks one has to check, which can make any defense *theoretically impossible*.

Interestingly, a very similar class of such general n-D attacks based on observable probability differences have been carefully studied by Asghar et al. [27] to prove some essential security flaws of an observer-resistant password system called Foxtail proposed by Li and Shum [29]. The attacks described by Asghar et al. [27] are not the same as the attack proposed in this paper, but conceptually they follow the same idea: some security-critical objects occurs in challenges with a probability different from others, and such probability differences are observable to attackers for revealing some important security-sensitive information. While the proposed attacks on UTS-CAPTCHA work in 1-D case, Asghar et al.'s attack works the best in 2-D case so the generalization from 1-D to n-D becomes more important.

Generalizing the 1-D attack to n-D cases is straightforward but the mathematics and experimental validation are much more complicated. Since the 1-D attack has been proved working well, we refer readers to [27] for more details on how the generalization can be done.

3.5. Additional system level attacks

It is difficult to conceive how a UTS-CAPTCHA is going to identify different clients. Just using IP addresses might not always be a good choice, as some traffic will come from privacy-protecting VPNs, or from the TOR network, and thus IPs will not be necessarily related to clients. The same happens in other scenarios, as large NAT sub-networks.

Although in general fingerprinting browsers and their environment (OS, hardware, etc.) is straightforward, it does depend on the mechanisms implemented by the browsers. As an example of a broken fingerprinting mechanism, Sivakorn et al. demonstrated a recent attack [9] on the "behavioural detection" mechanism implemented at Google's "No CAPTCHA reCAPTCHA".

4. Experimental results

To verify the proposed attacks including the methods for estimating system parameters, we implemented a simulated UTS-CAPTCHA service and a number of algorithms for parameter estimation and attacks in MATLAB[4]. We used the original parameters used by Kwon and Cha in [26], but our results can be easily generalized to other settings. We assume that the whole process starts with a bot whose initial classification accuracy for a single image is β_0, which corresponds to an overall CAPTCHA solving success rate $\approx \beta_0^{c-n_{\max}/2}$.

In the following, we first show the experimental results of estimating system parameters, and then the performance of actual attacks where a bot was able to improve its performance by using the UTS-CAPTCHA service as an oracle.

4.1. Estimation of system parameters

For the estimation of p, we tried both approaches (the one based on the mean frequency and the one based on solving Eq. (18)). We also calculated c/N_{unique} as an upper bound of p, where N_{unique} denotes the number of unique images observed (which includes both normal and trap images). The method based on Eq. (18) becomes very slow when N becomes large, so after N is larger than a threshold $N_t = 200$, we split all challenges into trunks of 200 challenges to get an estimate of $\overline{N_{\text{unique}}}$ which is then used to estimate p with $N = N_t$. Figure 2 shows one set of results we obtained, with a bot whose single-image classification accuracy is 0.8 (corresponding to a very low overall success rate of $0.8^{c-n_{\max}/2} \approx 0.018$). From the results we can see that p can be estimated with an increasing accuracy while the number of observed challenges N increases. With over 1,000 challenges the accuracy becomes sufficiently high. The number of challenges required matches our expectation that it should be at the order of $1/p \approx 563$. The mean frequency based approach performs just slightly better than the upper bound c/N_{unique}, suggesting that it may be sufficient to use the upper bound itself as an estimate. The method based on solving Eq. (18) can obviously allow a faster (but rougher) estimation of p even with a very small number of observed challenges. It has a tendency to over-estimate p which is in general not an issue for the proposed attacks as over-estimation means being more conservative.

For the estimation of p_t, we used both the k-means based approach and also the maximum frequency as its upper bound. Figure 3 shows one set of

[4] The source code of the attacks, written in MATLAB, can be downloaded from `https://github.com/hooklee/UTS-CAPTCA-cracker`.

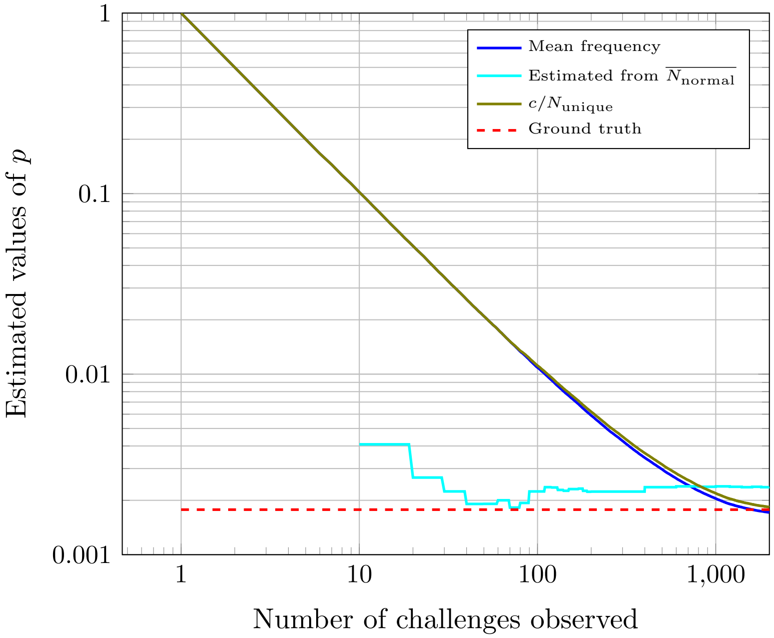


Figure 2: The results of different approaches to estimating p.

results we obtained with the same settings as the results of Figure 2. Note that the ground truth value of p_t changes while new trap images are added. The results show that the k-means based approach can give a rough estimate of p_t very quickly after just a small number of challenges. The maximum frequency also responded very quickly although it always over-estimated p_t as expected. The fast response suggests that the value of p_t can always be quickly estimated.

For both p and p_t, we repeated the experiment with different values of β_0 and the results are largely the same.

4.2. Results of actual attacks

In this subsection we report selected results of some actual attacks on UTS-CAPTCHA. We focus on the two attacks with minimum requirements on estimation of system parameters – Attacks 1 and 5. For both attacks, the bot will be able to gradually improve its performance to 100% within a reasonable number of challenges. Note that the results below are based on some parameters that can be fine tuned to reduce the number of challenges needed. Parallel attacks can also be used to improve the efficiency. We did

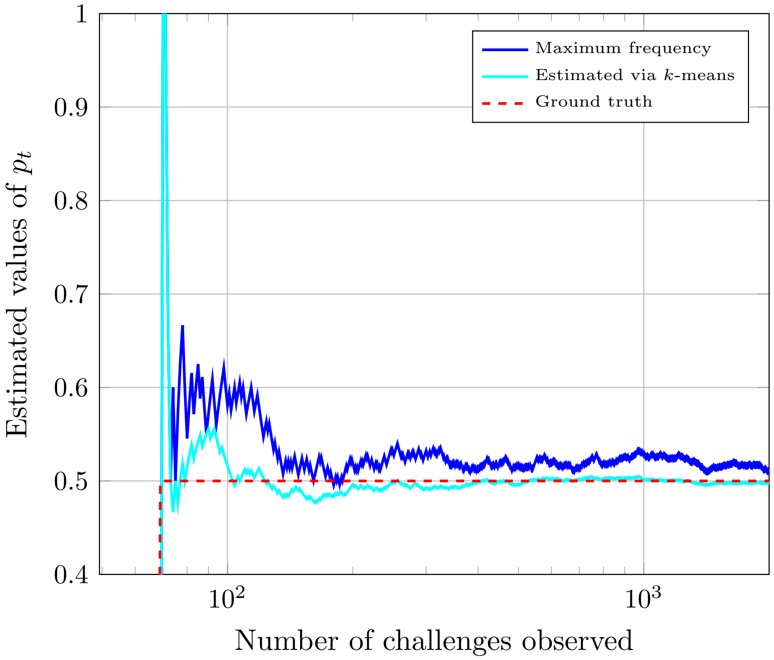


Figure 3: The results of different approaches to estimating p_t.

not implement the other three attacks because they have more requirements on estimation of system parameters and it is unlikely they will perform better than Attacks 1 and 5.

4.2.1. Attack 1

In this attack, the bot continuously monitors frequencies of all images and conducts k-means to identify possible trap images. Since every time a new trap image is added into TI, the value of p_t will change, the frequencies will be reset every time when a challenge is successfully passed. To ensure the detected high-frequency cluster indeed corresponds to trap images, we set a threshold $T_k = 10$ and only count the high-frequency cluster's centroid as valid if its is more than T_k times larger than the low-frequency centroid. To ensure the k-means algorithm will see a large enough difference between the two clusters, we let the bot observe at least $\lceil T_k \rceil$ challenges after each passed challenge. This attack is very robust and can recover from missed trap images due to the continuous monitoring. Figure 4 shows how the performance of a bot with initial single-image classification accuracy $\beta_0 = 0.8$ improves its performance over time.

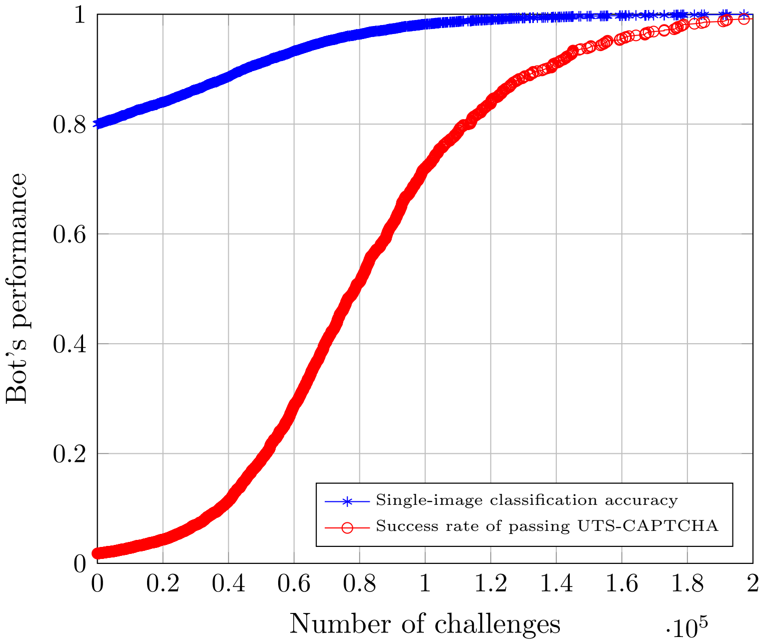


Figure 4: Success rate of Attack 1 with $\beta_0 = 0.8$ and up to 200,000 challenges.

4.2.2. Attack 5

For this attack, we implemented a bot switching between two modes: 1) active responding mode to try to meet a new passed challenge and potentially new trap images; 2) passive learning mode to try to detect the new trap images to improve its performance in classifying images and passing future CAPTCHA challenges. As mentioned before, this attack does not require estimation of any system parameters, but the bot needs to set the period of each learning phase to ensure all new trap images will be reliably detected. In our implementation, we use the number of active trap images detected plus a small margin $\delta \leq c/2$ to estimate an upper bound of $|\mathrm{TI}|$ and then multiply it by a fixed factor $\alpha > 1$ to ensure each trap image will appear for a sufficient number of times. Figure 5 shows how the performance of a bot improves over time when more challenges are observed, with parameters $\beta_0 = 0.8$, $\delta = c/4$ and $\alpha = 10$. This attack needed more challenges than Attack 1 to get to 100% success rate, as it is more conservative.

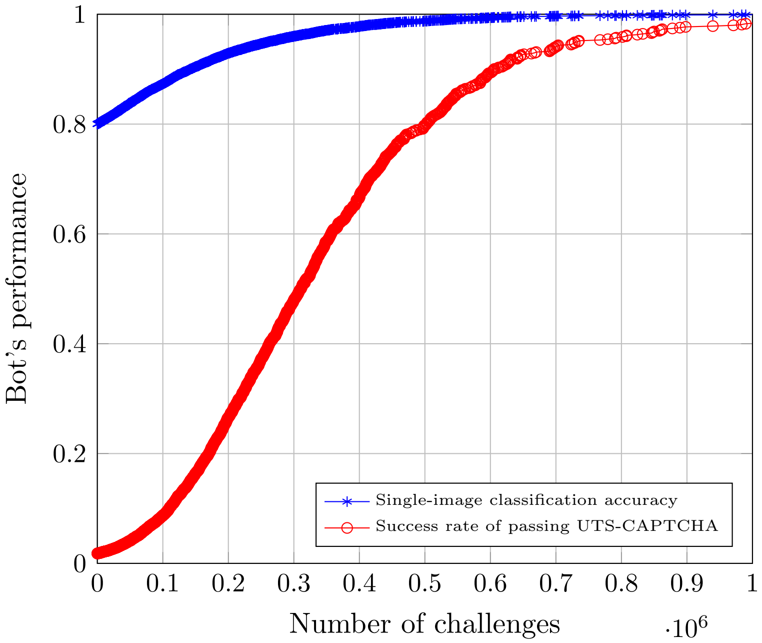


Figure 5: Success rate of Attack 5 with $\beta_0 = 0.8$ and up to 1,000,000 challenges.

5. Possible Improvements

In this section, we discuss some possible countermeasures against our attacks.

5.1. Decoy trap images

Our trap image learning attacks are all based on the possibility to correctly identify trap images due to their higher frequency of appearance in the CAPTCHAs generated.

To reduce the possibility of trap images being detected, we may try the following countermeasures:

- Changing the rate at which the trap images appear. This means having a bigger set of trap images (i.e., a larger value of $|\mathrm{TI}|$).

- Changing the rate at which (some of) the other images appear.

The first option can be implemented in several ways. We discuss one of them in the next subsection.

The second option may allow the CAPTCHA designers to increase the difficulty of the statistical analysis. To do so, they can create an additional set DTI of "decoy trap images". Every time they add or remove an image x_i to TI, they add or remove an image x_j to DTI. For every challenge in which they present n_{TI} trap images, they also present $n_{\mathrm{DTI}} = n_{\mathrm{TI}}$ decoy trap images. Thus, in every challenge, the attacker (bot) will see a set of images that appear more than the rest, but it will not be able to judge what are real trap images and what are decoy ones.

The problem with this is that the bot can do a statistical analysis on all of these images. The bot can *guess* which ones are real and which ones are decoy, then try to answer them semi-randomly and answer all other images in a challenge using its classification mechanism. This approach will possibly lower the bot's chances of passing the challenge, but it will allow it to gather statistical information on the real and decoy trap images: if a challenge is passed and there is one or more possible decoy images, then all such possible decoy image must be decoys; and if the UTS-CAPTCHA always removes an image from TI once it is correctly solved, this can be noticed by one previously observed possible trap image missing in future observations, which implies that this particular image is a real trap so its ground truth label can be learned.

The above analysis shows that it is hard to avoid trap images being detected because there are many probability differences that may be exploited by a bot. When some differences are removed, some others often appear, so it seems very difficult to remove all such exploitable statistical differences. This may be an unfortunate effect of complicated systems like UTS-CAPTCHA.

5.2. Learning trap images from the bot

One of the problems of the UTS-CAPTCHA proposal is the extreme difference in sizes between TI and $\mathrm{M} \cup \mathrm{MN}$. This difference is not necessary. It is possible for the UTS-CAPTCHA to present a series of challenges and learn possible trap images from a bot. Then, when the size of TI is at least a certain percentage of the image set ($\mathrm{TI} \geq r \times |\mathrm{M} \cup \mathrm{MN}|$, where $0 \leq r \leq 1$), the UTS-CAPTCHA starts using the images from TI.

In the particular case presented in [26], the interesting rate would be that mimics the expected one, so that $r = 1.5/22$. This particular rate is too high, though, and thus does allow the bot to learn to classify some images before trap images start being used. Having a too large TI also means it will take a longer time to start making the system work.

5.3. Uncertainty over trap images

The reason we can learn the correct classification label of a trap image is that trap images always count towards grading. We can increase the number of trap images used per challenge, but let not all of them count towards grading the challenge.

If the size of TI remains small, we will still be able to identify trap images as each trap image has a high likelihood to repeat in multiple challenges. However, the bot will need to pass more challenges with the same trap image in order to infer the correct classification label of the trap image. As mentioned before, if the UTS-CAPTCHA removes those images from TI that have been solved correctly, the bot will be able to infer which images we did solve correctly immediately.

If the UTS-CAPTCHA does not remove any correctly solved images from TI, once a bot learns about enough correct classification labels of some trap images, it will be able to drastically increase its success rate against the UTS-CAPTCHA, given that images from TI appear more frequently as long as $\frac{\mathrm{TI}}{|\mathrm{M} \cup \mathrm{MN}|} \ll \frac{1.5}{22}$, which will be true until TI becomes close to $|\mathrm{M} \cup \mathrm{MN}|$. After TI becomes very large, it is likely the bot can already solve most of the trap images (which should have been removed), so the effect of trap images will become less effective.

5.4. Adaptive challenges

One possibility is to get rid of the idea of trap images all together and replace it by another related idea: showing *more frequently* those images that the bot *typically* miss-classifies.

This cannot be implemented as such, as it will basically mimic what the trap images do so a bot can learn in more or less the same way. It may be implemented *in disguise*, so that these images are mixed with normal ones. However, this is very similar to using uncertainties in trap images as discussed in the previous section, so will unlikely work well.

Ultimately, the problem is that the *stronger* and *more efficient* you want your anti-bot mechanism to work (i.e., presenting more often miss-classified images), the more information you will be giving away to benefit the attacker.

5.5. Other possible measures?

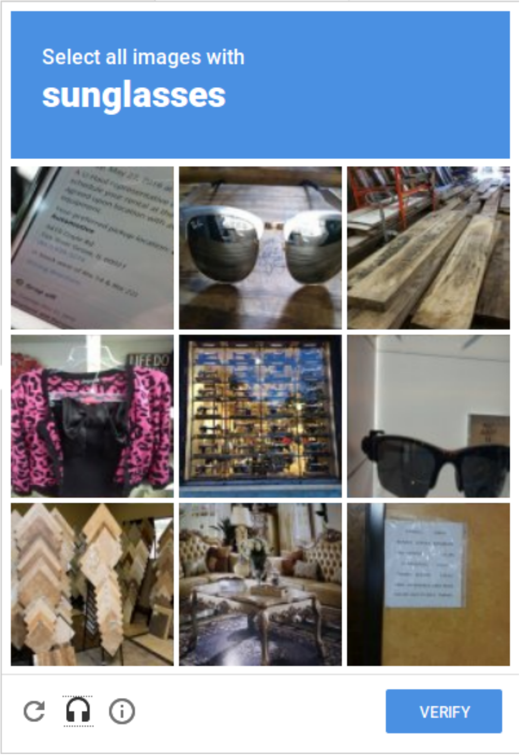
Our discussions on possible improvements so far shows that it does not seem straightforward to fix UTS-CAPTCHA to not leak information about correct classification labels of trap images. Although the general idea behind the UTS-CAPTCHA is sound, the main difficulty seems about how to hide trap images while making them work to block bots. The key here seems that, as long as there is direct feedback, anything blocking a bot can always be detected by the bot, and therefore it can learn from failures. Since direct feedback at the CAPTCHA challenge level is unavoidable by nature, maybe there is a theoretical barrier for the UTS-CAPTCHA to overcome, which is an interesting open question for further research.

6. More discussions

As reviewed in the introduction section, many CAPTCHA schemes have been broken including modern ones. The nature of CAPTCHA implies that any CAPTCHA scheme depends on the assumed hardness of some AI problem, meaning that progress of AI can lead to new attacks and the scheme has to evolve or adapt to catch up. Image classification based CAPTCHAs are not exceptions. They are generally susceptible to attacks based on modern image classification AI models such as deep convolutional neural networks (DCNNs), which have been proven capable of classifying with great accuracy images pertaining to many image categories [30]. For instance, the image recognition based second layer of Google's "No CAPTCHA reCAPTCHA" (see Fig. 6 for an example challenge) was shown insecure against deep learning based attacks [9].

To some extent, we can argue that all CAPTCHA schemes are by definition adaptive if they want to stay secure. A representative example is Google's reCAPTCHA, which has gone through several generations of evolution, starting from the original design with two distorted words in an image [31], to the combination of a distorted text and a street view image [32], then to the second generation (v2) – the so-called "No CAPTCHA reCAPTCHA" , the so-called "invisible reCAPTCHA badge" and a dedicated version for protecting Android apps – which includes a first layer based on "Advanced Risk Analysis" considering "a user's entire engagement with the CAPTCHA" and a second layer based on image recognition tasks [6], and finally to the newest version (v3) using an invisible risk score based on which the site owner can decide what to do [33]. While the wish to improve usability has played a role, the evolution of Google's reCAPTCHA has been driven by many attacks on its earlier and current versions [9, 34–36].

While CAPTCHA schemes should be adaptive, AI-based attacks can also adapt and evolve as well. For instance, an existing attack based on a trained



Select all images with
sunglasses

Figure 6: An example challenge obtained from Google's "No CAPTCHA reCAPTCHA".

AI model can adaptively evolve by continuously retraining the model using new samples via *incremental learning* or *reinforcement learning*, and it can also learn to classify new unseen labels with a small number of new examples through *transfer learning*. The adaptive nature of learning-based attacks makes it even more important that CAPTCHA schemes should be adaptively evolving.

While the statistical learning-based oracle attacks are very ad hoc and work only on UTS-CAPTCHA, the fundamental idea behind them is quite general: if an attacker can observe any statistical differences in CAPTCHA challenges and exploit such information to infer ground truth labels of items included in CAPTCHA challenges, then some learning-based attacks will be possible. This can be further extended to any information leaked, statistical or not, for inferring ground truth labels. In the following, we discuss how this fundamental idea has been or may be applied to attack some other example CAPTCHA schemes. We leave a more systematic generalization of our proposed attacks against UTS-CAPTCHA as our future work.

1) Google reCAPTCHA: Currently both v2 and v3 are being used. The probably more widely used v2 includes the following two layers: a first layer based on a checkbox "I am not a bot", and a second layer showing normally a number of image classification tasks. The whole process is adaptive in the sense that a later layer will not show if an earlier layer is passed. This adaptive display of a later layer effectively leaks information about the earlier layer. This information has been exploited by Sivakorn et al. [9] to reverse engineer how the "Advanced Risk Analysis" worked to bypass the checkbox layer. Similarly, it allowed Akrout et al. [36] to employ reinforcement learning to produce an attack of the checkbox layer, where a bot gradually refines its behavior towards the direction of being accepted by the system. The image classification layer was also found to contain some indirectly leaked information [9]: by reverse image search it is possible to retrieve tags of images from open Web, which can allow inferring the correct response with a sufficient accuracy. This attack can be re-interpreted statistically: the probabilities that each image in a CAPTCHA challenge relates to different possible tags are different, so the ground truth label can be effectively guessed. In addition, looking at Fig. 6, we can see correct images relating to the given tag appear more times, therefore can further increasing the probability of the relevant tag in the whole CAPTCHA challenge. Such statistical differences may also exist at feature level (e.g., special texture features related to a specific type of objects), which could also be exploited to infer ground truth labels.

2) The image orientation based CAPTCHA scheme proposed in [37]: This scheme has a mechanism of adaptively removing two different types of images: images that become easier to orient by computers over time, and images that are too difficult for humans. While removing images cannot help a bot to improve its orientation detector's performance directly, it could allow the bot to work out a three-class classifier to focus on its efforts: images that are easy to orient by computers, images that are hard to orient by humans, other images. Ground truth labels can be obtained by passively observing the target CAPTCHA service to see what images were removed. To detect a removed image, the bot can run a very similar attack to any one proposed in this paper, where removed images appearing with a zero probability after being removed, absolutely smaller than the occurrence probability of any image that is still in the image pool.

3) Cortcha, an image recognition based CAPTCHA scheme proposed in [8]: This scheme detaches a computer-segmented object from a challenge

image, applies image inpainting to fix the broken boundary caused by the detached object, and then searches in a large image database for similar computer-segmented objects that are used as decoy objects. The user/bot is required to move the correct detached object from all candidates to the original position in the challenge image. See Fig. 7 for an example showing how Cortcha works. Observing all candidate objects in Fig. 7, we can see the correct one has a quite different texture structure or a different color tone. In addition, we can see in the challenge images, there are two other salient objects (other two flowers) that are much more similar to the correct detached object rather than the other decoy objects, in terms of texture and color tone. The different level of similarities will allow identification of the correct object. While this will normally work only with a specific success rate, a bot could exploit the results (successes and failures) to infer useful information about how the decoy objects are generated to develop a classifier that can incrementally increase the success rate of such similarity-based attacks. According to [8], some specific features (color histogram, the complexity and potentially SIFT features) are used to find the decoy objects from all candidates, but as long as some features are not properly considered a bot can learn a classifier to capture such neglected features to increase its chance of telling the correct object from decoy ones. The authors of [8] did acknowledge this can be a problem, but argued that the system can be adapted to consider more features without explaining how this can be done.

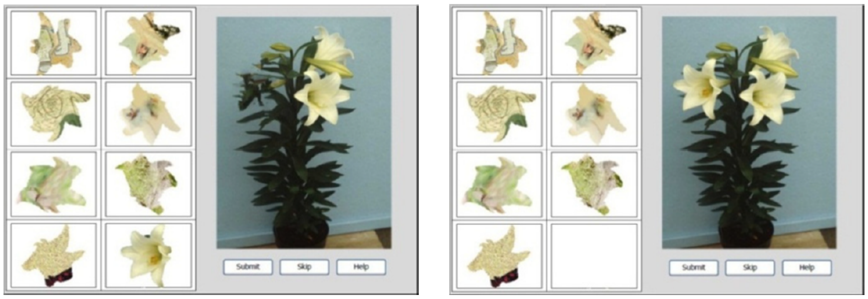


Figure 7: Left: an example challenge of Cortcha (Fig. 7 in [8]); right: after the correct detached object (the one at the right bottom corner of all candidate objects) is moved back to the original position in the challenge image (Fig. 8 in [8]).

Based on the above discussion, we propose a new basic principle for the design of future CAPTCHA schemes, supplementing the CAPTCHA design principles other researchers have proposed such as those described in [8, 38].

***Principle**: For a new CAPTCHA design, any potential statistical differences should be carefully checked for any observable objects, object groups (e.g., objects sharing a specific common feature) and detectable features of such objects and object groups, in order to avoid leaking information about ground truth labels of such objects or object groups to facilitate practical learning based attacks.*

The above principle may be too abstract to apply in practice, so we also suggest a number of concrete design guidelines, some of which have been used in existing CAPTCHA designs.

- *Checking all system parameters so that they do not lead to exploitable statistical differences*: For instance, for UTS-CAPTCHA, the parameters $\bar{t}$, c, $|\mathrm{TI}|$, and $|\mathrm{M} \cup \mathrm{MN}|$ can be set in such a way to have $p_t = \frac{\bar{t}}{|\mathrm{TI}|} = p = \frac{c}{|\mathrm{M} \cup \mathrm{MN}|}$ or to reduce $|p_t - p|$, which will lead to a very different design of UTS-CAPTCHA.

- *Using a (very) large object database*: This can help reduce the probability of specific object or feature, therefore reducing any potential statistical differences. While this cannot completely remove the possibility of statistical attacks, it can increase the time for an attack to become effective, allowing the system to remain secure for a sufficiently long time before being updated. This idea has been used in some CAPTCHA schemes such as Cortcha [8] and DeepCAPTCHA [25] by getting a large number of images from the open Internet.

- *Using a dynamic (time-varying) object database*: This can help make it harder for a bot to accumulate evidence of any statistical differences since the target keeps changing. This idea is used in DeepCAPTCHA [25] by discarding any used images and keeping retrieving new images to effectively make the database size “bottomless”.

- *Using non-public (secret) object database*: This can make it harder for a bot to collect sufficient training samples to build a classifier. However, since a bot can leverage a botnet to passively harvest objects used in a non-public database, this countermeasure has to be combined with the above two to make a botnet-based harvesting process less effective. An effective way of generating non-public objects is to generate them using computer synthesis methods, but such methods should be designed carefully to avoid future objects becoming predictable (e.g., some identical objects are reused). Most CAPTCHA schemes (e.g., old-generation CAPTCHAs with distorted images of words) actually adopt this countermeasure.

- *Avoid reusing any used source images*: Such images can be good samples for training and re-training a classifier used in a learning based attack. This idea is used in DeepCAPTCHA [25].

- *Avoiding tailoring CAPTCHA challenges against a specific suspected bot*: This is what UTS-CAPTCHA chooses to do, and such adaptation can actually help a bot to adapt rather than block it. Note that it is very easy for a bot to avoid being blocked or identified by changing its running environment (e.g., employing a botnet) or behavior (e.g., making its behavior change depending on the environment).

- *Paying attention to statistical differences between objects in a CAPTCHA challenge and publicly available objects on the open Internet*: Our analysis on some example CAPTCHA schemes given in this section shows that exploitable statistical differences can go beyond the object database used by a CAPTCHA scheme, so how the objects used statistically link to external objects that are publicly available to a bot should be considered as well.

- *Frequently applying major changes to the CAPTCHA generation process*: Considering the risk of any new learning based attacks will appear in future, it is always a good idea to keep the target (the CAPTCHA scheme) moving. Major changes should be applied from time to time, and old settings should not be reused. This is what Google reCAPTCHA is doing, but the frequency of its changes is still not high enough.

- *Using hybrid CAPTCHA schemes involving different types of CAPTCHA challenges*: It can be difficult or even impossible to avoid all statistical differences in a specific CAPTCHA scheme. By using hybrid CAPTCHA schemes where different types of CAPTCHA challenges are mixed up, we could significantly increase the difficulty of all being broken. The hybridization can be done in a way that different challenges are intermingled so they cannot be attacked following a simple “divide and conquer” manner.

It deserves mentioning that the proposed principle is not limited to image-based CAPTCHAs only. For instance, most audio based CAPTCHAs can be attacked by a similar learning-based attack as well since they often depend on added noises to defeat speech recognition. If an attacker can learn how such noises are added, he may be able to adaptively evolve a machine learning model to be more robust against such added noises. For the Google reCAPTCHA v2 and v3 based on an "invisible" risk score, it is also possible that an attacker is able to learn about how the risk score is generated and then adapts its behavior to pass the test.

The above principle and the concrete design guidelines can obviously make the design of new CAPTCHA schemes harder since they apply more constraints on what designers can do and may also negatively influence the usability of a CAPTCHA scheme. Therefore, we recommend CAPTCHA designers to consider how to best balance security and usability for their CAPTCHA schemes. As a rule of thumb, if a CAPTCHA scheme can evolve relatively fast (e.g., via frequent change of parameters and ingredients forming CAPTCHA challenges), its security can be limited to a time window therefore relaxing the security requirements. In order to be less constrained by the proposed principle, it will also help if CAPTCHA designers can consider moving away from traditional CAPTCHA designs based on hard AI problems to other approaches less vulnerable to learning-based attacks. For instance, Google reCAPTCHA v3 [33] uses a risk score and lets the website owner to decide when and what to do on suspicious activities, and the "Completely Automated Public Physical test to tell Computers and Humans Apart" (CAPPCHA) depends on some physical actions only humans can do (e.g., arranging a mobile device in a specific position) [39, 40]. Some work for solving other security problems may also be used to design new CAPTCHA-like schemes, e.g., Čagalja et al. [41] reported a paper and smartphone based "weakly unrelayable channel" against (automatic) relay attacks, which can be used to design CAPPCHAs as well.

7. Conclusion

Kwon and Cha [26] presented two mechanisms to increase the robustness of image classification based CAPTCHAs, creating a *Uncertainty and Trap Strengthened* CAPTCHA (UTS-CAPTCHA). The scheme was designed to be securer against oracle-based attacks that can learn ground truth labels of images to incrementally train an attacking classifier. A key feature of their scheme is that CAPTCHA challenges generated are adaptive to users/bots according to their responses observed. Similar adaptive mechanisms have also been proposed by other researchers and CAPTCHA designers, but no previous attacks on such adaptive CAPTCHA schemes have been reported.

In this paper, we report a fundamental statistical flaw of UTS-CAPTCHA, which allows us to identify a number of oracle-based statistical attacks against UTS-CAPTCHA. Our theoretical analyses and experimental simulations show that the attacks are effective and can convert a weak classifier with a very low success rate into one eventually reaching a close to 100% success rate. We also discuss the implications of the reported attacks on other CAPTCHA schemes and propose a general principle with a number of concrete guidelines that should be followed in future for designing new CAPTCHA schemes.

Acknowledgments

Carlos Javier Hernández-Castro wants to thank Vova Sokolov for his valuable input. In addition, he would also like to thank, in no particular order, Julio H.O., Declan J.H.O., Julio C.H.C., Jhenya S., Maya A.H.S., Carmen C.V. and Lena S. for their contributions and support.

The work of Shujun Li was partly supported by the research projects COMMANDO-HUMANS (`http://www.commando-humans.net/`) and ACCEPT (`http://accept.cyber.kent.ac.uk/`), funded by the EPSRC (Engineering and Physical Sciences Research Council) in the UK, under grant numbers EP/N020111/1 and EP/P011896/2, respectively.

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

Supplementary material

Supplementary material associated with this article (source code of implemented attacks) can be found, in the online version, at doi:10.1016/j.cose.2020.101758.

References

[1] M. Naor, Verification of a human in the loop or identification via the Turing test, online document (1996).
URL `http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.ps`

[2] L. Von Ahn, M. Blum, N. J. Hopper, J. Langford, CAPTCHA: Using hard AI problems for security, in: Advances in Cryptology – EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings, Vol. 2656 of Lecture Notes in Computer Science, Springer, 2003, pp. 294–311 (2003). doi:10.1007/3-540-39200-9_18.

[3] J. M. G. Hidalgo, G. Alvarez, CAPTCHAs: An artificial intelligence application to web security, Advances in Computers 83 (2011) 109–181 (2011). doi:10.1016/B978-0-12-385510-7.00003-5.

[4] Q.-J. Li, Y.-B. Mao, Z.-Q. Wang, A survey of CAPTCHA technology, Journal of Computer Research and Development 49 (3) (2012) 469–480 (2012). doi:10.1016/B978-0-12-385510-7.00003-5.

[5] Y.-W. Chow, W. Susilo, P. Thorncharoensri, CAPTCHA design and security issues, in: K.-C. Li, X. Chen, W. Susilo (Eds.), Advances in Cyber Security: Principles, Techniques, and Applications, Springer Nature Switzerland AG, 2019, Ch. 4, pp. 69–92 (2019). doi:10.1007/978-981-13-1483-4_4.

[6] V. Shet, Are you a robot? Introducing No CAPTCHA reCAPTCHA, blog article (2014).
URL `https://security.googleblog.com/2014/12/are-you-robot-introducing-no-captcha.html`

[7] P. Golle, Machine learning attacks against the Asirra CAPTCHA, in: Proceedings of 2009 5th Symposium on Usable Privacy and Security, ACM, 2009, pp. 535–542 (2009). doi:10.1145/1572532.1572585.

[8] B. B. Zhu, J. Yan, Q. Li, C. Yang, J. Liu, N. Xu, M. Yi, K. Cai, Attacks and design of image recognition CAPTCHAs, in: Proceedings of 2010 ACM International Conference on Computer and Communication Security, ACM, 2010, pp. 187–200 (2010). doi:10.1145/1866307.1866329.

[9] S. Sivakorn, I. Polakis, A. D. Keromytis, I am robot: (deep) learning to break semantic image CAPTCHAs, in: Proceedings of 2016 2nd IEEE European Symposium on Security and Privacy, IEEE, 2016, pp. 388–403 (2016). doi:10.1109/EuroSP.2016.37.

[10] H. Weng, B. Zhao, S. Ji, J. Chen, T. Wang, Q. He, R. Beyah, Towards understanding the security of modern image captchas and underground captcha-solving services, Big Data Mining and Analytics 2 (2) (2019) 118–144 (2019). doi:10.26599/BDMA.2019.9020001.

[11] N. J. Mitra, H.-K. Chu, T.-Y. Lee, L. Wolf, H. Yeshurun, D. Cohen-Or, Emerging images, ACM Transactions on Graphics 28 (5) (2009) 163:1–163:8, ACM SIGGRAPH Asia 2009 papers (2009). doi:10.1145/1618452.1618509.

[12] N. J. Rubenking, Are you a human? CAPTCHA and beyond, online document (2013).
URL `https://uk.pcmag.com/software/8686/are-you-a-human-captcha-and-beyond`

[13] M. Conti, C. Guarisco, R. Spolaor, CAPTCHaStar! a novel CAPTCHA based on interactive shape discovery, in: Applied Cryptography and Network Security: 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings, Vol. 9696 of Lecture Notes in Computer Science, Springer, 2016, pp. 611–628 (2016). doi:10.1007/978-3-319-39555-5_33.

[14] M. Szczys, CAPTCHA bot beats new Are You A Human PlayThru game, online document (2012).
URL `https://hackaday.com/2012/05/25/captcha-bot-beats-new-are-you-a-human-playthru-game/`

[15] C. J. Hernández-Castro, M. D. R-Moreno, D. F. Barrero, Using JPEG to measure image continuity and break Capy and other puzzle CAPTCHAs, IEEE Internet Computing 19 (6) (2015) 46–53 (2015). doi:10.1109/MIC.2015.127.

[16] T. Gougeon, P. Lacharme, How to break CaptchaStar, in: Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018), SCITEPRESS, 2018, pp. 41–51 (2018). doi:10.5220/0006577600410051.

[17] M. Okada, W. H. Saito, Capy risk-based authentication (2016).
URL `\url{https://www.capy.me/products/risk\_based\_authentication/}`

[18] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet classification with deep convolutional neural networks, in: Proceedings of 25th International Conference on Neural Information Processing Systems, Curran Associates, Inc., 2012, pp. 1097–1105 (2012).

[19] D. Ciregan, U. Meier, J. Schmidhuber, Multi-column deep neural networks for image classification, in: Proceedings of 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2012, pp. 3642–3649 (2012). doi:10.1109/CVPR.2012.6248110.

[20] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, V. D. Shet, Multi-digit number recognition from street view imagery using Deep Convolutional Neural Networks, in: Proceedings of 2014 International Conference on Learning Representations, 2014 (2014).

[21] Y. Taigman, M. Yang, M. Ranzato, L. Wolf, DeepFace: Closing the gap to human-level performance in face verification, in: Proceedings of 2014 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2014, pp. 1701–1708 (2014). doi:10.1109/CVPR.2014.220.

[22] F. Stark, C. Hazırbas, R. Triebel, D. Cremers, , in: Proceedings of 37th German Conference on Pattern Recognition (GCPR 2015) Workshop New Challenges in Neural Computation, 2015 (2015).
URL `https://vision.in.tum.de/_media/%20spezial/bib/stark-gcpr15.pdf`

[23] A. Nguyen, J. Yosinski, J. Clune, Deep neural networks are easily fooled: High confidence predictions for unrecognizable images, in: Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2015, pp. 427–436 (2015). doi:10.1109/CVPR.2015.7298640.

[24] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, A. Swami, The limitations of deep learning in adversarial settings, in: Proceedings of 2015 1st IEEE European Symposium on Security and Privacy, IEEE, 2015, pp. 372–387 (2015). doi:10.1109/EuroSP.2016.36.

[25] M. Osadchy, J. Hernandez-Castro, S. Gibson, O. Dunkelman, D. Pérez-Cabo, No bot expects the DeepCAPTCHA! Introducing immutable adversarial examples with applications to CAPTCHA, IEEE Transactions

on Information Forensics and Security 12 (11) (2017) 2640–2653 (2017). doi:10.1109/TIFS.2017.2718479.

[26] S. Kwon, S. Cha, A paradigm shift for the CAPTCHA race: Adding uncertainty to the process, IEEE Software 33 (6) (2016) 80–85 (2016). doi:10.1109/MS.2016.32.

[27] H. J. Asghar, S. Li, R. Steinfeld, J. Pieprzyk, Does counting still count? revisiting the security of counting based user authentication protocols against statistical attacks, in: Proceedings of 2013 20th Network & Distributed System Security Symposium, Internet Society, 2013 (2013). URL http://www.hooklee.com/Papers/NDSS2013_Full.pdf

[28] G. A. F. Seber, Multivariate Observations, John Wiley & Sons, Inc., 2004 (2004).

[29] S. Li, H.-Y. Shum, Secure human-computer identification (interface) systems against peeping attacks: SecHCI, IACR's Cryptology ePrint Archive: Report 2005/268 (2005). URL http://eprint.iacr.org/2005/268

[30] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2016, pp. 770–778 (2016). doi:10.1109/CVPR.2016.9.

[31] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, M. Blum, reCAPTCHA: Human-based character recognition via web security measures, Science 321 (5895) (2008) 1465–1468 (2008). doi:10.1126/science.1160379.

[32] S. Perez, Google now using ReCAPTCHA to decode street view addresses, online document (2012). URL https://techcrunch.com/2012/03/29/google-now-using-recaptcha-to-decode-street-view-addresses/

[33] W. Liu, Introducing reCAPTCHA v3: the new way to stop bots, Google Webmaster Central Blog (2018). URL https://webmasters.googleblog.com/2018/10/introducing-recaptcha-v3-new-way-to.html

[34] C. Houck, J. Lee, , presentation at DEF CON 18 (2010).
URL https://media.defcon.org/DEF%20CON%2018/DEF%20CON%2018%20slides/DEF%20CON%2018%20Hacking%20Conference%20Presentation%20By%20Chad%20Houck%20and%20Jason%20Lee%20-%20Decoding%20reCAPTCHA%20-%20Slides.mp4

[35] E. Homakov, The No CAPTCHA problem, blog article (2014).
URL http://homakov.blogspot.com.es/2014/12/the-no-captcha-problem.html

[36] I. Akrout, A. Feriani, M. Akrout, Hacking Google reCAPTCHA v3 using reinforcement learning, preprint, arXiv:1903.01003 (2019).
URL https://arxiv.org/abs/1903.01003

[37] R. Gossweiler, M. Kamvar, S. Baluja, What's up CAPTCHA? A CAPTCHA based on image orientation, in: Proceedings of 2009 International Conference on World Wide Web, ACM, 2009, pp. 841–850 (2009). doi:10.1145/1526709.1526822.

[38] C. J. Hernandez-Castro, Where do CAPTCHAs fail: A study in common pitfalls in CAPTCHA design and how to avoid them, Ph.D. thesis, University of Alcala, Spain (2017).

[39] M. Guerar, M. Migliardi, A. Merlo, M. Benmohammed, B. Messabih, A Completely Automatic Public Physical test to tell Computers and Humans Apart: A way to enhance authentication schemes in mobile devices, in: Proceedings of 2015 International Conference on High Performance Computing & Simulation, 2015 (2015). doi:10.1109/HPCSim.2015.7237041.

[40] M. Guerar, A. Merlo, M. Migliardi, Completely Automated Public Physical test to tell Computers and Humans Apart: A usability study on mobile devices, Future Generation Computer Systems 82 (2018) 617–630 (2018). doi:10.1016/j.future.2017.03.012.

[41] M. Čagalja, T. Perković, M. Bugarić, S. Li, Fortune cookies and smartphones: Weakly unrelayable channels to counter relay attacks, Pervasive and Mobile Computing 20 (2015) 64–81 (2015). doi:10.1016/j.pmcj.2014.09.002.