# Adversarial CAPTCHAs

Chenghui Shi, Xiaogang Xu, Shouling Ji, *Member, IEEE*, Kai Bu, *Member, IEEE*, Jianhai Chen, *Member, IEEE*, Raheem Beyah, *Senior Member, IEEE*, and Ting Wang, *Member, IEEE*

***Abstract*—Following the principle of to set one's own spear against one's own shield, we study how to design adversarial completely automated public turing test to tell computers and humans apart (CAPTCHA) in this article. We first identify the similarity and difference between adversarial CAPTCHA generation and existing hot adversarial example (image) generation research. Then, we propose a framework for text-based and image-based adversarial CAPTCHA generation on top of state-of-the-art adversarial image generation techniques. Finally, we design and implement an adversarial CAPTCHA generation and evaluation system, called aCAPTCHA, which integrates 12 image preprocessing techniques, nine CAPTCHA attacks, four baseline adversarial CAPTCHA generation methods, and eight new adversarial CAPTCHA generation methods. To examine the performance of aCAPTCHA, extensive security and usability evaluations are conducted. The results demonstrate that the generated adversarial CAPTCHAs can significantly improve the security of normal CAPTCHAs while maintaining similar usability. To facilitate the CAPTCHA security research, we also open source the aCAPTCHA system, including the source code, trained models, datasets, and the usability evaluation interfaces.**

***Index Terms*—Adversarial image, completely automated public turing test to tell computers and humans apart (CAPTCHA), deep learning, usable security.**

## I. Introduction

CAPTCHA is a type of *challenge-response test* in computing which is used to distinguish between human and automated programs (machines). The first generation of completely automated public turing test to tell computers and humans apart (CAPTCHA) was invented in 1997, while the term "CAPTCHA" was first coined in 2002 [1], [2]. Ever since its invention, CAPTCHA has been widely used to improve the security of websites and various online applications to prevent the abuse of online services, such as preventing phishing, bots, spam, and Sybil attacks.

Although there are many different proposals for CAPTCHA design, for example, text-based CAPTCHA [22], [23]; image-based CAPTCHA [25], [26]; audio-based CAPTCHA [29]; video-based CAPTCHA [8]; and other CAPTCHA designs [40], [41], in this article, our study mainly focuses on *text- and image-based CAPTCHAs*. The reason is evident: they are the most accepted and widely used CAPTCHAs up to now and in the foreseeable future. The study of their security and usability has more potential implications for practical applications.

*Issues of CAPTCHAs and Motivation:* Generally speaking, CAPTCHA can be evaluated according to its *security performance*, which refers to the strength and resilience of CAPTCHAs against various attacks, and *usability performance*, which refers to how user friendly the CAPTCHAs are. From the security perspective, it is not news to see reports that a CAPTCHA scheme is broken by some attacks [20], [21], [27], [39]. The evolution of CAPTCHAs always moves forward in a spiral, constantly accompanied by emerging attacks. For text-based CAPTCHAs, the security goal of its earliest version is to defend against optical character recognition (OCR)-based attacks. Therefore, many distortion techniques (e.g., varied fonts, varied font sizes, and rotation) are applied. Over the last decade, machine learning algorithms have become more powerful. Following the seminal work which demonstrates that computers turn to outperform humans in recognizing characters, even under severe distortion, many successful attacks to text-based CAPTCHAs were proposed, including both generic attacks which target multiple text-based CAPTCHAs [6], [7], and specialized attacks which targeted one kind of text-based CAPTCHAs [24]. Despite this, it is possible to improve the security of text-based CAPTCHAs by increasing the distortion and obfuscation levels, and their usability will be significantly affected [6], [7].

The same dilemma exists for image-based CAPTCHAs. With the prosperity of machine learning research, especially recent deep learning progress, deep neural networks (DNNs) have achieved impressive success in image classification/recognization, matching, or even outperforming the cognitive ability of humans in complex tasks with thousands of classes [15]. Along with such progress, many DNN-based attacks have been proposed recently to crack

Manuscript received November 29, 2019; revised July 7, 2020; accepted March 19, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB2103802; in part by NSFC under Grant 61772466, Grant U1936215, and Grant U1836202; and in part by the Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under Grant LR19F020003. This article was recommended by Associate Editor H. Lu. *(Corresponding author: Shouling Ji.)*

Chenghui Shi and Shouling Ji are with the Cancer Institute (Key Laboratory of Cancer Prevention and Intervention, Ministry of Education), the Second Affiliated Hospital, School of Medicine, and the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China (e-mail: chenghuishi@zju.edu.cn; sji@zju.edu.cn).

Xiaogang Xu is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: xgxu@cse.cuhk.edu.hk).

Kai Bu and Jianhai Chen are with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China (e-mail: kaibu@zju.edu.cn; chenjh919@zju.edu.cn).

Raheem Beyah is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: rbeyah@ece.gatech.edu).

Ting Wang is with the College of Information Sciences and Technology, Penn State University, State College, PA 16801 USA (e-mail: inbox.ting@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2021.3071395.

Digital Object Identifier 10.1109/TCYB.2021.3071395

image-based CAPTCHAs with very high success probability, as demonstrated by a large number of reports [28]. To defend against existing attacks, the intuition is to rely on high-level image semantics and develop more complex image-based CAPTCHAs, for example, recognizing an image object by utilizing its surrounding context [27]. Leaving the security gains aside, such designs usually induce poor usability. To make things worse, unlike text-based CAPTCHAs, it is difficult, if not impossible, for designers to generate specific images with required semantical meanings through certain rules. In other words, it is too labor intensive to collect labeled images in a large scale.

In summary, existing text- and image-based CAPTCHAs are facing challenges from both the security and the usability perspectives. It is desirable to develop a new CAPTCHA scheme that achieves high security while preserving proper usability, that is, it seeks a better balance between security and usability.

*Our Methodology and Contributions:* To address the dilemma of existing text- and image-based CAPTCHAs, we start from analyzing state-of-the-art attacks. It is not surprising that most, if not all, of the attacks to text- and image-based CAPTCHAs are based on the machine learning techniques, especially the latest and most powerful ones, which are mainly based on deep learning, typically, CNNs. This is mainly because the development of CAPTCHA attacks roots in the progress of machine learning research, as we discussed before.

On the other hand, with the progress of machine learning research, researchers found that many machine learning models, especially neural networks, are vulnerable to *adversarial examples*, which are defined as elaborately (maliciously, from the model's perspective) crafted inputs that are imperceptible to humans but that can fool the machine learning model into producing undesirable behavior, for example, producing incorrect outputs [31]. Inspired by this fact, is that possible for us to design a new kind of CAPTCHAs by proactively attacking existing CAPTCHA attacks, that is, "to set one's own spear against one's own shield?"

Following this inspiration, we study the method to generate text- and image-based CAPTCHAs based on adversarial learning, that is, *text-based adversarial CAPTCHAs* and *image-based adversarial CAPTCHAs*, that are resilient to state-of-the-art CAPTCHA attacks and meanwhile preserve high usability. Specifically, we have three main objectives in the design: 1) *security*, which implies that the developed CAPTCHAs can effectively defend against state-of-the-art attacks, especially the powerful deep-learning-based attacks; 2) *usability*, which implies that the developed CAPTCHAs should be usable in practice and maintain a high user experience; and 3) *compatibility*, which implies that the proposed CAPTCHA generation scheme is compatible with existing text- and image-based CAPTCHA deployment and applications.

With the above goals in mind, we study the method to inject human-tolerable, preprocessing-resilient (i.e., cannot be removed by CAPTCHA attacks) perturbations to traditional CAPTCHAs. Specifically, we design and implement a novel system *aCAPTCHA* to generate and evaluate text- and image-based adversarial CAPTCHAs.

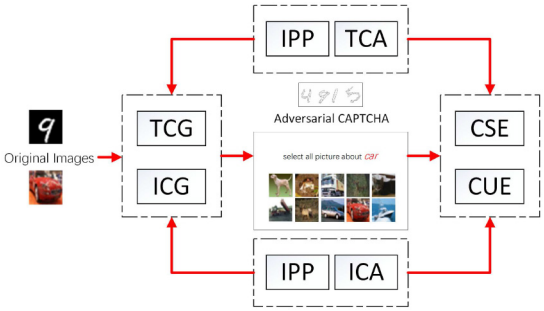



Fig. 1. System overview of aCAPTCHA.

Our main contributions can be summarized as follows.

1) Following our design principle, we propose a framework for generating adversarial CAPTCHAs on top of existing adversarial example (image) generation techniques. Specifically, we propose four text-based and four image-based adversarial CAPTCHA generation (ICG) methods. Then, we design and implement a comprehensive adversarial CAPTCHA generation and evaluation system, called *aCAPTCHA*, which integrates 12 image preprocessing (IPP) techniques, nine CAPTCHA attacks, four baseline adversarial CAPTCHA generation methods, and eight new adversarial CAPTCHA generation methods. aCAPTCHA can be used for the generation, security evaluation, and usability evaluation of both text- and image-based adversarial CAPTCHAs.
2) To examine the performance of the adversarial CAPTCHAs generated by aCAPTCHA, we conducted extensive security and usability evaluations. The results demonstrate that the generated adversarial CAPTCHAs can significantly improve the security of normal CAPTCHAs while maintaining similar usability.
3) We open source the aCAPTCHA system at [44], including the source code, trained models, datasets, and the interfaces for usability evaluation. It is expected that aCAPTCHA can facilitate the CAPTCHA security research and can shed light on designing more secure and usable adversarial CAPTCHAs.

## II. System Overview

In this section, we present the system architecture of aCAPTCHA, which is shown in Fig. 1. Basically, it consists of seven modules.

*IPP Module:* In this module, we implement 12 widely used standard IPP techniques for CAPTCHA security analysis, including nine filters: 1) BLUR; 2) DETAIL; 3) EDGE ENHANCE; 4) SMOOTH; 5) SMOOTH MORE; 6) GaussianBlur; 7) MinFilter; 8) MedianFilter; and 9) ModeFilter, two morphological operations: 1) dilation and 2) close, and one standard image binarization method. Basically, all the preprocessing techniques can be used to remove the noise in an image.

*Text-Based CAPTCHA Attack (TCA) Module:* In this module, we implement five TCAs, including two traditional

machine learning-based attacks (SVM, KNN) and three state-of-the-art DNN-based attacks (LeNet [11], MaxoutNet [12], and NetInNet [13]). In aCAPTCHA, TCA has two main functions. First, it can provide necessary model information for generating text-based adversarial CAPTCHAs, that is, for the following text-based adversarial CAPTCHA generation (TCG) module. Second, it can also be employed to evaluate the resilience of text-based CAPTCHAs against actual attacks.

*Image-Based CAPTCHA Attack (ICA) Module:* Similar to TCA, we implement four state-of-the-art ICAs in this module (NetInNet [13], VGG [14], GoogleNet [16], and ResNet [17]). It is used to provide necessary model information for generating image-based adversarial CAPTCHAs and for evaluating the resilience of image-based CAPTCHAs against actual attacks.

*TCG Module:* In this module, we first implement four state-of-the-art adversarial example (image) generation algorithms to serve as the baseline. Then, we analyze the limitations of applying existing adversarial image generation techniques to generate text-based adversarial CAPTCHAs. Finally, according to our analysis, we propose four new text-based adversarial CAPTCHA generation algorithms.

*ICG Module:* In this module, we first analyze the limitations of existing adversarial image generation techniques for generating image-based adversarial CAPTCHAs. Then, we implement four ICG algorithms by improving existing techniques.

*CAPTCHA Security Evaluation (CSE) Module:* Leveraging TCA and ICA, this module is used to evaluate the resilience and robustness of text- and image-based CAPTCHAs against state-of-the-art attacks.

*CAPTCHA Usability Evaluation (CUE) Module:* This module is mainly used for evaluating the usability of text- and image-based CAPTCHAs.

aCAPTCHA takes a fully modular design, and is thus easily extendable. We can freely add emerging attacks to TCA/ICA and/or add new proposed adversarial CAPTCHA generation algorithms to TCG/ICG.

## III. Text-Based Adversarial CAPTCHAs

With the design goals in mind and following our design principle, we show the design of TCG step by step below.

### A. Baselines

In fact, CAPTCHAs can be viewed as a special case of images. Then, following the design principle and goals, a straightforward idea is to generate text-based adversarial CAPTCHAs using exiting adversarial image generation techniques. Therefore, we implement four baseline adversarial image generation algorithms in TCG. Before delving into the details, we define some useful notations.

*1) Notations:* We first present necessary notations in the context of generating adversarial images. To be consistent with existing research, we use the same notation system as that in [10]. We represent a neural network as a function $F(x) = y$, where $x \in \mathbb{R}^{n\times n}$ is the input image[1] and $y \in \mathbb{R}^m$ is the corresponding output. Define $F$ to be the full neural network, including the softmax function and let $Z(x) = z$ be the output of all the layers except the softmax. According to $y$, $F$, which can be viewed as a classifier, assigns $x$ a class label $C(x)$. Let $C^*(x)$ be the correct label of $x$.

As in [9] and [10], we use $L_p$ norms to measure the similarity of $x, x' \in \mathbb{R}^{n\times n}$. Then, $L_p = \|x - x'\|_p = (\sum_{i=1}^{n}\sum_{j=1}^{n}|x - x'|^p)^{1/p}$. According to the definition, $L_2$ distance measures the Euclidean distance between $x$ and $x'$; $L_0$ distance measures the number of coordinates $i$ s.t. $x_{i,j} \neq x'_{i,j}$; and $L_\infty$ distance measures the maximum change to any of the coordinates, that is, $\|x - x'\|_\infty = \max\{|x_{1,1} - x'_{1,1}|, \ldots, |x_{n,n} - x'_{n,n}|\}$.

*2) Baseline Methods:* Recently, to generate adversarial examples (adversarial images in our context) against neural networks, many attacks have been proposed [30], [32]. For our purpose, those attacks can serve as our adversarial CAPTCHA generation methods. In TCG, we implement four state-of-the-art attacks as our baseline methods.

*Jacobian-based Saliency Map Attack (JSMA):* Papernot *et al.* [9] proposed the JSMA to generate adversarial images. JSMA is a greedy algorithm. Suppose $l$ is the target class of image $x$. Then, to obtain $x'$ such that $x' \neq x$ and $C(x') = l$, JSMA follows the following steps: 1) $x' = x$; 2) based on the gradient $\nabla Z(x')_l$, compute a *saliency map* in which each value indicates the impact of the corresponding pixel on the resulting classification; 3) according to the saliency map, select the most important pixel for modification to increase the likelihood of class $l$; and 4) repeat the above two steps until $C(x') = l$ or more than a set threshold of pixels have been modified.

Note that, JSMA is also capable for generating untargeted adversarial images. For that purpose, we only have to: 1) let $l = C(x)$ and change the goal as to find $x'$ such that $x' \neq x$ and $C(x') \neq l$; and 2) select the pixel to mostly decrease the likelihood of class $l$ for modification.

*Carlini–Wagner Attacks:* Aiming at generating high quality adversarial images, Carlini and Wagner [10] introduced three powerful attacks tailored to $L_2$, $L_0$, and $L_\infty$, respectively. Basically, all those three attacks are optimization based and can be targeted or untargeted. Taking the untargeted $L_2$ attack as an example, it can be formalized as the optimization problem: minimize $\|\delta\| + c \cdot F(x+\delta)$, such that $x+\delta \in [0,1]^n$, that is, for image $x$, the attack seeks for a perturbation $\delta$ that is small in length and can fool the classifier $F$ meanwhile. In the formalization, $c$ is a hyperparameter that balances the two parts in the objective function. The constraint implies that the generated adversarial image should be valid.

Previous work [19] shows that untargeted attacks are more transferable than targeted attacks in the black-box setting, the adversarial CAPTCHAs generated by untargeted attacks can achieve a better security performance than that by targeted attacks. Therefore, we focus on untargeted setting to generate adversarial CAPTCHAs. Without additional mention, all

[1]Note that, $x$ is not necessary to be a square image. The setting here is for simplicity.

TABLE I
PERFORMANCE OF BASELINE ALGORITHMS VERSUS LENET. THE ORIGINAL SAR OF LENET IS 95.87%

| Filter | $JSMA$ | | $L_2$ | | $L_0$ | | $L_\infty$ | |
|---|---|---|---|---|---|---|---|---|
| | – | **B** | – | **B** | – | **B** | – | **B** |
| – | 0.00% | 13.93% | 0.00% | 73.51% | 0.00% | 1.38% | 0.00% | 83.30% |
| BLUR | 5.15% | 8.27% | 4.22% | 20.84% | 6.25% | 19.27% | 6.25% | 22.52% |
| DETAIL | 17.80% | 11.76% | 0.00% | 78.28% | 2.22% | 4.22% | 56.79% | 83.30% |
| EDGE ENHANCE | 9.05% | 8.27% | 0.00% | 2.77% | 9.89% | 9.89% | 26.21% | 35.13% |
| SMOOTH | 43.36% | 37.71% | 0.00% | 64.70% | 24.31% | 7.54% | 28.24% | 94.15% |
| SMOOTH MORE | 37.71% | 40.46% | 0.00% | 37.71% | 20.84% | 10.79% | 19.27% | 88.58% |
| GaussianBlur | 49.70% | 16.42% | 0.35% | 35.13% | 28.24% | 22.52% | 22.52% | 73.51% |
| MinFilter | 0.15% | 1.38% | 0.05% | 0.11% | 0.02% | 0.07% | 0.06% | 0.15% |
| MedianFilter | 24.31% | 68.99% | 0.05% | 35.13% | 17.80% | 12.81% | 12.81% | 68.99% |
| ModeFilter | 20.84% | 30.40% | 0.00% | 22.52% | 30.40% | 32.69% | 0.05% | 40.46% |

adversarial CAPTCHAs are generated by untargeted setting in the remainder of this article.

### B. Datasets

For the text-based evaluation scenario, we employ Modified National Institute of Standards and Technology database (MNIST). MNIST [3] is a large database of 70 000 handwritten digit images and is widely used by the research community as a benchmark. We can generate text-based CAPTCHAs by the following ways; 1) sample $l$, for example, 6, images from MNIST randomly where each image contains a handwritten digit and each image size is $28 * 28$ and 2) concatenate these images to form a whole CAPTCHA where using $o$ to control the overlap between two images. In this article, the default value for $o$ is 0. The size of a whole CAPTCHA is $28 * 168$ for $l = 6$. Note that we have evaluated text-based CAPTCHAs with different $l$. However, due to the space limitation, we only present the results for text-based CAPTCHAs with $l = 6$.

### C. Analysis of Baselines

As discussed before, intuitively, it seems like that existing adversarial image generation algorithms, for example, JSMA and Carlini–Wagner attacks, can be applied to generate adversarial CAPTCHAs directly. Following this intuition, we conduct a preliminary evaluation as follows.

1) Leverage MNIST randomly generates 10 000 CAPTCHAs of length 6, that is, each CAPTCHA is composed of six characters from MNIST. Denote these CAPTCHAs by set $\mathcal{C}$.
2) Suppose LeNet from TCA is the employed CAPTCHA attack. Then, use LeNet (trained using 50 000 CAPTCHAs for 20 000 rounds and with batch size 50) to attack the CAPTCHAs in $\mathcal{C}$. The *Success Attack Rate* (SAR), which is defined as the portion of successfully recognized CAPTCHAs in $\mathcal{C}$, is 95.87%;
3) In terms of LeNet, generate the adversarial versions of the CAPTCHAs in $\mathcal{C}$ using JSMA, $L_2$, $L_0$, and $L_\infty$, denoted by $\mathcal{C}_J$, $\mathcal{C}_2$, $\mathcal{C}_0$, and $\mathcal{C}_\infty$, respectively.
4) Use LeNet and possible preprocessing techniques from the IPP module to attack $\mathcal{C}_J$, $\mathcal{C}_2$, $\mathcal{C}_0$, and $\mathcal{C}_\infty$. The corresponding SARs are shown in Table I, where "$-$" implies does not apply the corresponding preprocessing and $B$ denotes the *image binarization* processing.

It is worth noting that we do not consider using multiple filters at the same time. This is because combining different image filters may cause negative effects for CAPTCHA recognition. In practice, when a filter removes the noise in the image, it also damages the details of the image. Using multiple filters would loss much of image details and even make the image unrecognizable for the attack model.

From Table I, we observe that without applying IPP, the adversarial CAPTCHAs generated by all the baseline algorithms can significantly reduce the SAR of LeNet, for example, $L_2$ reduces the SAR of LeNet from 95.87% to 0%. This implies that the idea of applying adversarial CAPTCHAs to defend against modern attacks is promising.

However, unfortunately, without considering the usability, the security of these adversarial CAPTCHAs can be significantly affected by IPP either. For instance, when attacking $\mathcal{C}_\infty$, the SAR of LeNet is raised from 0% to 28.24% after applying the SMOOTH filter and to 94.15% after further applying image binarization, which is similar to its performance on normal CAPTCHAs. This implies that the perturbation in the adversarial CAPTCHAs can be removed by IPP, that is, the perturbations added by the baseline algorithms are not resilient/robust to IPP.

We analyze the reasons from two aspects. From the perturbation generation perspective, most, if not all, of the existing adversarial image generation techniques, including the baseline algorithms, are focusing on injecting perturbations to images in the space domain. However, for CAPTCHA attacks, various IPP methods are usually employed to remove irrelevant information before actual recognition. Those preprocessing methods are especially effective in removing the noise in the spatial domain. Thus, existing adversarial image generation techniques frequently behave unstable when against the CAPTCHA attacks along with IPP. From the CAPTCHA application perspective, we are on the defensive side when generating adversarial images (CAPTCHAs) instead of as their original purpose for attacking neural network models. Therefore, as long as the adversarial CAPTCHAs are usable, more perturbations can be injected to the CAPTCHAs, which is totally different from the design principle of existing adversarial image generation techniques, that are working hard for injecting human imperceptible or as less as possible perturbations.

### D. Adversarial CAPTCHA Generation

In the previous section, we analyzed the limitations of existing techniques for generating adversarial CAPTCHAs. Aiming

at generating more robust and usable text-based adversarial CAPTCHAs, we, in this section, proposed four new methods based on existing techniques.

Our design mainly follows two guidelines. First, according to our analysis, the perturbations added in the space domain are frail to IPP. Therefore, we consider to add perturbations in the frequency domain. This is because space domain perturbation can be considered as local change of images while frequency domain perturbation is a kind of global change to images, which is more difficult to remove, that is, frequency domain perturbation is intuitively more resilient to IPP. Certainly, when conducting frequency domain perturbation, we should be aware of the possible impact on the usability. Second, when generating adversarial CAPTCHAs, instead of trying to add human-imperceptible perturbations, we focus on adding *human-tolerable* perturbations. This will give us more freedom to design more secure and fast adversarial CAPTCHA generation methods. Specifically, based on JSMA, $L_2$, $L_0$, and $L_\infty$, we propose four text-based adversarial CAPTCHA generation algorithms, denoted by JSMA$^f$, $L_2^f$, $L_0^f$, and $L_\infty^f$, respectively.

*JSMA$^f$:* We show the design of JSMA$^f$ in Algorithm 1. Basically, JSMA$^f$ follows a similar procedure as the untargeted JSMA. We remark the differences as follows. First, in steps 3 and 4, we transform a CAPTCHA to the frequency domain by fast Fourier transform (FFT) and then compute a saliency map. This enables us to elaborately inject perturbations to a CAPTCHA in the frequency domain as expected.

Second, after transforming a CAPTCHA into the frequency domain, its high-frequency part usually corresponds to the margins of characters and other nonvital information, while the low-frequency part usually corresponds to the fundamental shape information of characters. Therefore, to decrease possible impacts on the usability of a CAPTCHA, we introduce a mask matrix $\varphi$ in Algorithm 1, which has the same size with $x$. $\varphi$ has values of 1 in the high-frequency part while 0 in the low-frequency part. Then, as shown in steps 5 and 6, we preserve the pixels in the low-frequency part while only considering to change the pixels in the high-frequency part.

Third, after selecting the candidate modified pixel, instead of modifying one pixel each time as in JSMA, we modify the candidate pixel and its neighbors as shown in step 7. This design is mainly based on the fact that close pixels in the frequency domain exhibit the *partial similarity* [42], that is, neighboring pixels in the frequency domain have very similar property and features. Therefore, modifying the candidate pixel and its neighbors would significantly accelerate the adversarial CAPTCHA generation process while not harmfully affect its quality (recall that, we are targeting to use user tolerable instead of as little as possible perturbations).

Finally, we make an inverse FFT (IFFT) for the CAPTCHA in the frequency domain and transform it back to the space domain as shown in step 8.

*$L_2^f$, $L_0^f$, and $L_\infty^f$:* Basically, $L_2^f$, $L_0^f$, and $L_\infty^f$ follow the similar procedures as that in $L_2$, $L_0$, and $L_\infty$ respectively, except that all the designs are finished in the frequency domain. The differences are the same as that between JSMA$^f$ and JSMA. Therefore, we omit their algorithm descriptions here while implementing them in TCG.

**Algorithm 1: JSMA$^f$**

**Input:** $x$ original CAPTCHAs; $C^*(x)$ the label of $x$; $F$ a classifier; $\varphi$ mask.
**Output:** $x'$ adversarial CAPTCHAs

1 $x' \leftarrow x$, $l \leftarrow C^*(x)$;
2 **while** $F(x') == l$ **do**
3 $\quad x'^f \leftarrow FFT(x')$;
4 $\quad$ compute a *saliency map* $\mathcal{S}$ based on the gradient $\nabla Z(x'^f)_l$;
5 $\quad \mathcal{S} \leftarrow \mathcal{S} \times \varphi$;
6 $\quad$ based on $\mathcal{S}$, select the pixel, denoted by $x'^f[i][j]$, that mostly decreases the likelihood of $l$;
7 $\quad$ modify $x'^f[i][j]$ and its neighbors to decrease the likelihood of $l$;
8 $\quad x' \leftarrow IFFT(x'^f)$;

### E. Evaluation

Now, we evaluate the security performance of JSMA$^f$, $L_2^f$, $L_0^f$, and $L_\infty^f$ and leave their usability evaluation in Section VI. Generally, the evaluation procedure is the same as that in Section III-C. In all the evaluations of this section, we employ MNIST to randomly generate CAPTCHAs of length 6. For each attack in TCA, we use 50 000 normal CAPTCHAs for training. Specifically, for the DNN-based attacks LeNet, MaxOut, and NetInNet, the batch size is 50 and each model is trained for 20 000 rounds. For each scenario, we use 1000 CAPTCHAs for testing. When generating an adversarial CAPTCHA, we set the inner $8 \times 8$ area as the high-frequency part while the rest as the low-frequency part for mask $\varphi$. Each evaluation is repeated three times and their average is reported as the final result.

First, we evaluate the performance of JSMA$^f$, $L_2^f$, $L_0^f$, and $L_\infty^f$ without any IPP. To conduct this group of evaluations, we 1) leverage JSMA$^f$, $L_2^f$, $L_0^f$, and $L_\infty^f$ to generate adversarial CAPTCHAs in terms of LeNet, MaxoutNet, and NetInNet, respectively and 2) leverage the attacks in the TCA module to attack these adversarial CAPTCHAs, respectively. The results are shown in Table II, where *Normal* indicates the SAR of each attack on the normal CAPTCHAs (nonadversarial versions).

From Table II, we have the following observations.

1) All the attacks in TCA are very powerful when attacking normal CAPTCHAs. However, when they attack the adversarial CAPTCHAs generated by JSMA$^f$, $L_2^f$, $L_0^f$, or $L_\infty^f$, none of them can break any adversarial CAPTCHA. This result is as expected and further demonstrates the advantage of applying adversarial CAPTCHAs to improve the security.
2) The generated CAPTCHAs by JSMA$^f$, $L_2^f$, $L_0^f$, and $L_\infty^f$ have very good transferability, that is, the adversarial CAPTCHAs generated in terms of one neural network

TABLE II
PERFORMANCE OF JSMA$^f$, $L_2^f$, $L_0^f$, AND $L_\infty^f$ (NO IPP)

| Attack Model | Normal | Text-based Adversarial CAPTCHA Generation | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | LeNet | | | | MaxoutNet | | | | NetInNet | | | |
| | | JSMA$^f$ | $L_2^f$ | $L_0^f$ | $L_\infty^f$ | JSMA$^f$ | $L_2^f$ | $L_0^f$ | $L_\infty^f$ | JSMA$^f$ | $L_2^f$ | $L_0^f$ | $L_\infty^f$ |
| **SVM** | 87.51% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% |
| **KNN** | 83.81% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% |
| **LeNet** | 95.87% | 0.01% | 0.00% | 0.00% | 0.00% | 0.00% | 0.01% | 0.00% | 0.00% | 0.01% | 0.00% | 0.00% | 0.00% |
| **MaxoutNet** | 95.29% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% |
| **NetInNet** | 96.45% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% |

TABLE III
PERFORMANCE OF JSMA$^f$, $L_2^f$, $L_0^f$, AND $L_\infty^f$ (FILTER + $\mathbf{B}$)

| Attack Model | Filter + $\mathbf{B}$ | Text-based Adversarial CAPTCHA Generation | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | LeNet | | | | MaxoutNet | | | | NetInNet | | | |
| | | JSMA$^f$ | $L_2^f$ | $L_0^f$ | $L_\infty^f$ | JSMA$^f$ | $L_2^f$ | $L_0^f$ | $L_\infty^f$ | JSMA$^f$ | $L_2^f$ | $L_0^f$ | $L_\infty^f$ |
| **SVM** | BLUR | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% |
| | DETAIL | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% |
| | EDGE ENHANCE | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% |
| | SMOOTH | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% |
| | SMOOTH MORE | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% |
| | GaussianBlur | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% |
| | MinFilter | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% |
| | MedianFilter | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% |
| | ModeFilter | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% |
| | Dilation | 31.30% | 19.12% | 27.21% | 26.41% | 29.09% | 25.44% | 28.45% | 27.83% | 28.88% | 27.83% | 28.45% | 27.62% |
| | Close | 15.13% | 8.57% | 13.47% | 12.60% | 14.17% | 12.07% | 15.51% | 14.28% | 13.03% | 17.37% | 14.76% | 13.59% |
| **LeNet** | BLUR | 0.32% | 0.29% | 0.26% | 0.24% | 0.30% | 0.49% | 0.30% | 0.28% | 0.38% | 0.25% | 0.35% | 0.29% |
| | DETAIL | 3.77% | 0.48% | 1.98% | 1.84% | 2.71% | 4.01% | 2.86% | 2.77% | 3.32% | 2.24% | 3.47% | 2.95% |
| | EDGE ENHANCE | 3.77% | 0.48% | 1.98% | 1.84% | 2.71% | 4.01% | 2.86% | 2.77% | 3.32% | 2.24% | 3.47% | 2.95% |
| | SMOOTH | 11.66% | 3.50% | 6.19% | 6.56% | 8.49% | 10.89% | 7.20% | 7.47% | 10.70% | 8.12% | 8.65% | 7.97% |
| | SMOOTH MORE | 8.89% | 2.71% | 5.10% | 4.81% | 6.94% | 9.13% | 5.68% | 5.85% | 8.49% | 6.49% | 6.81% | 6.56% |
| | GaussianBlur | 0.03% | 0.05% | 0.04% | 0.04% | 0.04% | 0.07% | 0.05% | 0.04% | 0.05% | 0.06% | 0.05% | 0.04% |
| | MinFilter | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% |
| | MedianFilter | 0.01% | 0.00% | 0.01% | 0.01% | 0.01% | 0.02% | 0.01% | 0.02% | 0.02% | 0.02% | 0.01% | 0.01% |
| | ModeFilter | 0.01% | 0.00% | 0.01% | 0.01% | 0.01% | 0.02% | 0.01% | 0.02% | 0.02% | 0.02% | 0.01% | 0.01% |
| | Dilation | 29.74% | 19.58% | 23.22% | 22.52% | 26.61% | 26.61% | 26.21% | 26.21% | 27.62% | 22.01% | 26.81% | 26.61% |
| | Close | 11.27% | 3.54% | 6.56% | 6.25% | 8.34% | 10.70% | 7.97% | 8.49% | 10.33% | 9.05% | 9.13% | 8.89% |

model are transferable to another neural network or traditional machine learning models. This demonstrates the good robustness of the adversarial CAPTCHAs generated by JSMA$^f$, $L_2^f$, $L_0^f$, and $L_\infty^f$.

Now, we discuss the reasons that we can achieve an excellent result even in the black-box setting. First, as stated in Section III-D, when generating adversarial CAPTCHAs, instead of trying to add human-imperceptible perturbations, we focus on adding human-tolerable perturbations. Our methods can inject a sufficient amount of adversarial perturbations into CAPTCHAs. A large amount of perturbations can provide a strong defense capability and fool different attack models even in the black-box setting. Second, despite five models in our experiments have different architectures, they are all trained by the same dataset (MNIST). This setting may limit the model difference among five models and somehow makes the generated adversarial CAPTCHAs easily transferable.

Now, on top of the above evaluation, we evaluate the resilience of the adversarial CAPTCHAs generated by JSMA$^f$, $L_2^f$, $L_0^f$, and $L_\infty^f$ against image filtering and image binarization. The evaluation procedure is the same as before except we use the filters in IPP to preprocess the adversarial CAPTCHAs before attack. The results are shown in Table III. From Table III, we have the following observations.

1) For different attacks, for example, SVM and LeNet, they become more powerful along with image filtering and binarization and can break adversarial CAPTCHAs to some extent in several scenarios.
2) The SARs of the attack model by using morphological operations increase largely than using image filters. It indicates that compared to 9 image filters, 2 morphological operations can remove more adversarial perturbations in CAPTCHAs. However, adversarial CAPTCHAs are obviously more secure than normal ones when considering the SAR rates of these attacks. Further, comparing the results in Table III with that in Table I, the adversarial CAPTCHAs generated by JSMA$^f$, $L_2^f$, $L_0^f$, and $L_\infty^f$ are also much more secure than the ones generated by JSMA, $L_2$, $L_0$, and $L_\infty$.
3) Similar as the previous evaluations, the adversarial CAPTCHAs maintain adequate transferability, which implies adversarial CAPTCHAs have stable robustness.

Finally, we further explore the security of text-based CAPTCHAs with different character numbers. We 1) leveraging JSMA$^f$ to generate adversarial CAPTCHAs in terms of LeNet; 2) using SMOOTH filter and image binarization to preprocess adversarial CAPTCHAs; and 3) leveraging the Lenet model to attack these adversarial CAPTCHAs. In this setting,

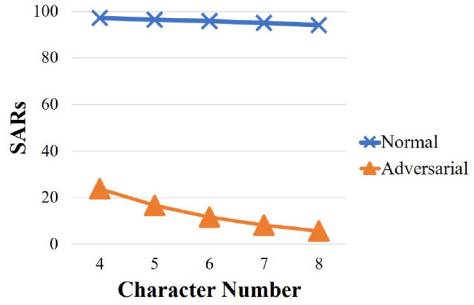



Fig. 2. Relationship between character number and SAR of the attack model.

the attack model can achieve a high SAR from Table III. The experimental results are shown in Fig. 2. From Fig. 2, we can observe that as the character number increases, the SARs of adversarial CAPTCHAs drop faster than that of normal CAPTCHAs. Because the attack model cannot recognize a single character in an adversarial CAPTCHA with a high accuracy, the SAR of a whole CAPTCHA will decrease exponentially with the increase of character number.

## IV. Image-Based Adversarial CAPTCHAs

### A. ICG Design

For ICG, we actually follow the same design principles as that for the text-based scenario. Furthermore, similar to the situation that existing adversarial image generation techniques are not suitable for generating text-based adversarial CAPTCHAs, they are not suitable for image-based adversarial CAPTCHAs due to similar reasons. Existing adversarial image generation techniques are mainly targeting to attack neural network models by adding as less as possible (human-imperceptible) perturbations to an image. However, we are standing on the defensive side to generate adversarial CAPTCHAs to improve the security. This implies that we might inject as much as possible perturbations to an image-based adversarial CAPTCHA as long as it is user tolerable (user recognizable). In addition, the adversarial example generation speed may not be a concern for existing techniques. Although it is not the main constraint for CAPTCHA generation neither, since we can generate the CAPTCHAs offline, we still expect to generate many CAPTCHAs in a fast way (since we may need to update our CAPTCHAs periodically to improve the system security). Therefore, we take efficiency as a consideration in adversarial CAPTCHA generation.

Image-based CAPTCHAs contain rich and important information which plays a key role in image classification. Thus, attackers cannot use radical IPP, such as image binarization, for image-based CAPTCHAs. When generating image-based adversarial CAPTCHAs, we can easily inject adversarial perturbations into any area of the CAPTCHA in the space domain. Therefore, we do not have to transform an image-based CAPTCHA to the frequency domain. Here, similar to the text-based scenario, we implement four ICG methods based on JSMA, $L_2$, $L_0$, and $L_\infty$, denoted by JSMA$^i$, $L_2^i$, $L_0^i$, and $L_\infty^i$, respectively.

**Algorithm 2: JSMA$^i$**

**Input:** $x$ original CAPTCHAs; $C^*(x)$ the label of $x$; $F$ a classifier; $K$ noise level.
**Output:** $x'$ adversarial CAPTCHAs

```
x' ← x, l ← C*(x);
while F(x') == l or K > 0 do
    compute a saliency map S based on the gradient ∇Z(x')_l;
    based on S, select the pixel, denoted by x'[i][j], that mostly decreases the likelihood of l;
    modify x'[i][j] and its neighbors to decrease the likelihood of l;
    K − −;
```

*JSMA$^i$:* We show the design of JSMA$^i$ in Algorithm 2, which basically follows the same procedure as JSMA. Following our design principle, we make two changes. First, we introduce an integer parameter $K$ to control the least perturbation that should be made. This implies that in our design, we try to inject as much as possible perturbations as long as the CAPTCHA is user tolerable (certainly, $K$ is an empirical value that can be decided based on some preliminary usability testing). Second, like to the text-based scenario, we modify multiple pixels simultaneously to accelerate the generation process.

*$L_2^i$, $L_0^i$, and $L_\infty^i$:* For the designs of $L_2^i$, $L_0^i$, and $L_\infty^i$, their procedures are the same as $L_2$, $L_0$, and $L_\infty$ except that we choose a small step and less iterations to accelerate the CAPTCHA generation process. This also implies that our perturbation injection scheme may not be optimal compared with the original $L_2$, $L_0$, and $L_\infty$. As we explained before, we are not targeting to add as less perturbation as possible like the original algorithms. Toward another direction, we try to inject more perturbations in a fast way when the CAPTCHA is user tolerable.

### B. Datasets

For the image-based evaluation scenario, we employ another image benchmark dataset ImageNet ILSVRC-2012 (refers to the dataset used for the 2012 ImageNet large-scale visual recognition challenge) [4]. The employed ImageNet ILSVRC-2012 contains 50 000 handlabeled photographs from 1000 categories with 50 photographs from each category.[2]

### C. Evaluation

Now, we evaluate the security performance of JSMA$^i$, $L_2^i$, $L_0^i$, and $L_\infty^i$ while leaving their usability evaluation in the next section. In the evaluation, we employ ImageNet ILSVRC-2012 to generate all the needed CAPTCHAs. Meanwhile, we use the pretrained models (all trained using the data in ImageNet ILSVRC-2012) of the attacks in ICA to examine the security performance of the generated adversarial CAPTCHAs,

[2]The used dataset here is actually a subset of ImageNet ILSVRC-2012, which is sufficient for our purpose.

TABLE IV
SECURITY OF IMAGE-BASED ADVERSARIAL CAPTCHAS

| | Normal | Image-based Adversarial CAPTCHA Generation | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | NetInNet | | | | GoogleNet | | | | VGG | | | | ResNet50 | | | |
| | | $JSMA^i$ | $L_2{}^i$ | $L_0{}^i$ | $L_\infty{}^i$ | $JSMA^i$ | $L_2{}^i$ | $L_0{}^i$ | $L_\infty{}^i$ | $JSMA^i$ | $L_2{}^i$ | $L_0{}^i$ | $L_\infty{}^i$ | $JSMA^i$ | $L_2{}^i$ | $L_0{}^i$ | $L_\infty{}^i$ |
| **NetInNet** | 41.72% | 0.0% | 0.0% | 0.0% | 0.0% | 4.6% | 20.3% | 4.2% | 1.9% | 0.7% | 5.4% | 1.9% | 2.2% | 4.1% | 3.3% | 8.8% | 4.7% |
| **GoogleNet** | 51.69% | 0.5% | 3.8% | 7.0% | 14.3% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.1% | 0.2% | 1.5% | 0.4% | 1.2% | 6.3% | 4.6% |
| **VGG** | 57.20% | 0.5% | 4.2% | 11.5% | 13.5% | 0.8% | 19.7% | 6.4% | 4.2% | 0.0% | 0.0% | 0.0% | 0.0% | 0.5% | 1.1% | 13.0% | 6.7% |
| **ResNet50** | 63.80% | 10.1% | 17.6% | 18.5% | 20.8% | 1.9% | 26.2% | 7.9% | 7.6% | 0.1% | 0.4% | 1.2% | 3.1% | 0.0% | 0.0% | 0.0% | 0.0% |

TABLE V
SECURITY OF IMAGE-BASED ADVERSARIAL CAPTCHAS VERSUS FILTERS

| | Filter | Image-based Adversarial CAPTCHA Generation | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | NetInNet | | | | GoogleNet | | | | VGG | | | | ResNet50 | | | |
| | | $JSMA^i$ | $L_2{}^i$ | $L_0{}^i$ | $L_\infty{}^i$ | $JSMA^i$ | $L_2{}^i$ | $L_0{}^i$ | $L_\infty{}^i$ | $JSMA^i$ | $L_2{}^i$ | $L_0{}^i$ | $L_\infty{}^i$ | $JSMA^i$ | $L_2{}^i$ | $L_0{}^i$ | $L_\infty{}^i$ |
| **NetInNet** | BLUR | 0.0% | 0.0% | 0.0% | 0.6% | 4.1% | 9.7% | 3.2% | 2.5% | 1.7% | 5.5% | 1.6% | 2.0% | 4.4% | 2.7% | 5.3% | 3.7% |
| | DETAIL | 0.0% | 0.0% | 0.0% | 0.1% | 1.0% | 16.7% | 2.2% | 0.8% | 0.4% | 5.7% | 0.9% | 1.3% | 1.2% | 1.3% | 4.1% | 2.8% |
| | EDGE ENHANCE | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 2.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.1% | 0.0% |
| | SMOOTH | 0.0% | 0.0% | 0.0% | 0.1% | 5.5% | 19.2% | 5.7% | 3.9% | 1.1% | 7.9% | 2.2% | 2.7% | 7.0% | 5.5% | 9.7% | 8.5% |
| | SMOOTH MORE | 0.0% | 0.0% | 0.0% | 0.1% | 6.0% | 19.1% | 6.2% | 4.9% | 1.5% | 8.2% | 3.0% | 3.2% | 7.3% | 5.7% | 9.4% | 7.6% |
| | GaussianBlur | 0.0% | 0.0% | 0.0% | 0.1% | 5.5% | 14.4% | 5.9% | 4.3% | 1.2% | 6.7% | 3.2% | 3.4% | 7.3% | 4.3% | 8.8% | 7.9% |
| | MinFilter | 0.0% | 0.0% | 0.0% | 0.3% | 5.1% | 12.3% | 3.0% | 5.5% | 2.1% | 3.1% | 1.2% | 4.4% | 5.8% | 1.7% | 5.5% | 10.0% |
| | MedianFilter | 0.0% | 0.0% | 0.0% | 0.1% | 5.1% | 16.2% | 6.5% | 2.9% | 1.2% | 7.9% | 3.5% | 2.3% | 5.5% | 5.7% | 9.4% | 5.3% |
| | ModeFilter | 0.0% | 0.0% | 0.0% | 0.1% | 4.6% | 20.3% | 4.2% | 1.9% | 0.7% | 5.4% | 1.9% | 2.2% | 4.1% | 3.3% | 8.8% | 4.7% |
| **GoogleNet** | BLUR | 0.8% | 5.7% | 5.9% | 8.2% | 0.0% | 3.9% | 1.3% | 1.3% | 0.9% | 4.3% | 2.8% | 2.9% | 6.8% | 3.4% | 6.2% | 6.5% |
| | DETAIL | 0.4% | 2.9% | 5.1% | 12.6% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.1% | 0.2% | 1.2% | 0.2% | 0.5% | 3.1% | 3.0% |
| | EDGE ENHANCE | 0.0% | 0.5% | 0.5% | 1.6% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.1% | 0.1% | 0.1% | 0.0% | 0.2% | 0.2% |
| | SMOOTH | 0.3% | 2.7% | 5.7% | 7.8% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.8% | 0.7% | 1.5% | 0.9% | 1.7% | 7.9% | 5.0% |
| | SMOOTH MORE | 0.4% | 3.8% | 7.1% | 9.1% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.8% | 0.8% | 1.5% | 1.1% | 1.2% | 8.1% | 5.7% |
| | GaussianBlur | 0.4% | 2.3% | 4.3% | 6.2% | 0.0% | 0.8% | 0.0% | 0.6% | 0.1% | 2.1% | 2.1% | 2.0% | 1.9% | 3.2% | 9.1% | 6.0% |
| | MinFilter | 2.1% | 3.8% | 4.2% | 10.8% | 0.0% | 1.2% | 0.2% | 1.7% | 0.2% | 1.1% | 0.7% | 3.4% | 1.7% | 0.7% | 5.3% | 7.6% |
| | MedianFilter | 0.3% | 1.9% | 5.1% | 5.3% | 0.0% | 0.3% | 0.0% | 0.2% | 0.1% | 1.8% | 1.6% | 1.0% | 1.7% | 3.7% | 7.3% | 2.6% |
| | ModeFilter | 0.5% | 3.8% | 7.0% | 14.3% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.1% | 0.2% | 1.5% | 0.4% | 1.2% | 6.3% | 4.6% |

that is, using the attacks in ICA to recognize the generated CAPTCHAs. These pretrained models have state-of-the-art performance and are available at Caffe Model Zoo [5]. For each evaluation scenario, we use 1000 CAPTCHAs for testing. Each evaluation is repeated three times and their average is reported as the final result.

We first evaluate the security of the adversarial CAPTCHAs generated by JSMA$^i$, $L_2^i$, $L_0^i$, and $L_\infty^i$ in the scenario of not considering any IPP. The results are shown in Table IV. *Normal* implies the SAR of each attack when against normal CAPTCHAs, and in the rest of the evaluation scenarios, we first generate adversarial CAPTCHAs in terms of the neural network model of an attack, for example, VGG, and then using different attacks to attack them. Further, the default setting is $K = 50$ for JSMA$^i$, and $K = 100$ for $L_2^i$, $L_0^i$, and $L_\infty^i$ (note that, in the original $L_2$, $L_0$, and $L_\infty$, there is also a parameter to control the noise level. We denote it by $K$ for consistence in $L_2^i$, $L_0^i$, and $L_\infty^i$).

From Table IV, we have the following observations. First, for image-based CAPTCHAs, adversarial learning techniques can significantly improve their security. This further confirms our design principle: to set one's own spear against one's own shield. Second, the generated adversarial CAPTCHAs demonstrate adequate transferability, that is, the adversarial CAPTCHAs generated in terms of one neural network model also exhibits good resilience to other attacks. Thus, they are robust.

Under the same settings with Table IV, we examine the security performance of JSMA$^i$, $L_2^i$, $L_0^i$, and $L_\infty^i$ against the attacks in ICA plus IPP. Note that, since all the CAPTCHAs are color images, we do not consider image binarization here. We show the results in Table V. Basically, same conclusions can be drawn from Table V as that from Table IV. In addition, we can find that image filtering has little impact on the security of the adversarial CAPTCHAs generated by JSMA$^i$, $L_2^i$, $L_0^i$, or $L_\infty^i$, that is, they are very robust.

Now, we consider the impact of different perturbation (noise) levels on the security of the generated adversarial CAPTCHAs. Taking JSMA$^i$ as an example, we show partial results in Table VI, from which we make the following observations. First, in most of the scenarios, when adding more noise, better security can be achieved, which is consistent with our intuition. However, according to the results, such security improvement is slight in most of the cases. Second, as before, the generated adversarial CAPTCHAs are resilient and robust to various attacks.

## V. ADAPTIVE SECURITY ANALYSIS

In this section, we analyze in depth the adaptive methods that could be applied against aCAPTCHA.

### A. Statement

In practical scenario, we assume the threat follows all of the following models.

*Knowledge of Adversarial Example Generation and Defense:* The attacker has full knowledge of adversarial example generation and defense schemes. They can get that information from the research community and other means.

TABLE VI
SECURITY OF IMAGE-BASED ADVERSARIAL CAPTCHAS VERSUS NOISE LEVEL

| | Filter | Image-based Adversarial CAPTCHA Generation | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | NetInNet | | | | GoogleNet | | | | VGG | | | | ResNet50 | | | |
| | | 20 | 30 | 40 | 50 | 20 | 30 | 40 | 50 | 20 | 30 | 40 | 50 | 20 | 30 | 40 | 50 |
| **NetInNet** | BLUR | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 1.6 % | 1.2 % | 1.1 % | 0.9 % | 0.6 % | 0.5 % | 0.4 % | 0.3 % | 2.7 % | 3.0 % | 2.4 % | 1.8 % |
| | DETAIL | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.8 % | 0.3 % | 0.1 % | 0.1 % | 0.2 % | 0.1 % | 0.1 % | 0.1 % | 0.9 % | 0.5 % | 0.2 % | 0.1 % |
| | EDGE ENHANCE | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % |
| | SMOOTH | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 2.7 % | 1.8 % | 1.2 % | 1.1 % | 0.3 % | 0.2 % | 0.2 % | 0.1 % | 3.6 % | 2.4 % | 1.6 % | 1.4 % |
| | SMOOTH MORE | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 3.0 % | 2.2 % | 1.2 % | 1.2 % | 0.4 % | 0.3 % | 0.3 % | 0.2 % | 3.6 % | 2.7 % | 2.2 % | 1.8 % |
| | GaussianBlur | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 1.8 % | 1.6 % | 1.4 % | 0.9 % | 0.3 % | 0.3 % | 0.1 % | 0.1 % | 3.3 % | 3.0 % | 1.8 % | 1.8 % |
| | MinFilter | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 2.0 % | 1.1 % | 1.2 % | 1.2 % | 0.3 % | 0.6 % | 0.4 % | 0.3 % | 2.0 % | 2.2 % | 2.0 % | 1.4 % |
| | MedianFilter | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 2.2 % | 1.8 % | 1.4 % | 1.2 % | 0.3 % | 0.4 % | 0.2 % | 0.1 % | 2.4 % | 1.8 % | 1.2 % | 1.1 % |
| | ModeFilter | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 1.8 % | 0.9 % | 0.6 % | 0.7 % | 0.2 % | 0.1 % | 0.1 % | 0.1 % | 2.4 % | 0.9 % | 0.5 % | 0.5 % |
| **GoogleNet** | BLUR | 0.2 % | 0.2 % | 0.1 % | 0.1 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.4 % | 0.1 % | 0.1 % | 0.1 % | 1.8 % | 2.0 % | 1.6 % | 1.8 % |
| | DETAIL | 0.2 % | 0.1 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.2 % | 0.1 % | 0.0 % | 0.0 % |
| | EDGE ENHANCE | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % |
| | SMOOTH | 0.1 % | 0.1 % | 0.1 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.5 % | 0.1 % | 0.1 % | 0.1 % |
| | SMOOTH MORE | 0.1 % | 0.1 % | 0.1 % | 0.1 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.5 % | 0.3 % | 0.2 % | 0.1 % |
| | GaussianBlur | 0.0 % | 0.1 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.9 % | 0.3 % | 0.2 % | 0.2 % |
| | MinFilter | 0.4 % | 0.3 % | 0.2 % | 0.2 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.1 % | 0.0 % | 0.0 % | 0.0 % | 1.8 % | 0.9 % | 0.4 % | 0.2 % |
| | MedianFilter | 0.1 % | 0.1 % | 0.1 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.6 % | 0.3 % | 0.3 % | 0.2 % |
| | ModeFilter | 0.2 % | 0.1 % | 0.1 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.0 % | 0.2 % | 0.1 % | 0.1 % | 0.0 % |

*No Knowledge of CAPTCHA Generation:* The attacker can realize that the CAPTCHAs were updated by adding adversarial noise, while they do not know the specific model and method used to generate the adversarial CAPTCHAs.

*No Access to the Source Images:* The attacker can only access to all generated adversarial CAPTCHAs but not to their source. They has no knowledge about the particular image used for generating the adversarial CAPTCHAs.

TABLE VII
PERFORMANCE OF ADVERSARIAL CAPTCHAS AGAINST ADAPTIVE ATTACK

| Adaptive Methods | Normal | Adversarial | | | |
|---|---|---|---|---|---|
| | | JSMA$^f$ | $L_2^f$ | $L_0^f$ | $L_\infty^f$ |
| – | 95.87% | 0.00% | 0.00% | 0.00% | 0.00% |
| EnAdv. Training | 96.95% | 28.40% | 21.26% | 25.84% | 23.20% |
| EnAdv. Training$^+$ | 97.12% | 48.61% | 41.37% | 39.35% | 41.37% |
| Distillation | 94.36% | 5.35% | 4.96% | 6.77% | 5.13% |
| Therm. Encoding | 92.39% | 12.19% | 7.68% | 9.89% | 11.24% |

## B. Adaptive Attack

When attackers are aware of the existence of the possible defense, they will try other methods against adversarial CAPTCHAs. Now, there are three types of state-of-the-art techniques against adversarial examples: 1) adversarial training; 2) gradient masking; and 3) input transformation. Attackers can adopt these techniques to improve their attacks. We introduce one representative method for each type of techniques respectively below.

*Ensemble Adversarial Training [36]:* This method augments a model's training data with adversarial examples crafted on other static pretrained models. As a result, minimizing the training loss implies increased the robustness to black-box attacks from some set of models.

*Defense Distillation [35]:* This method is a type of gradient masking-based defense technique. Defensive distillation modifies the softmax function to include a temperature constant $T$

$$\text{softmax}(x, T)_i = \frac{e^{x_i/T}}{\sum_j e^{x_i/T}}. \tag{1}$$

First, training a teacher model on the training set, using softmax at temperature $T$. Then using the teacher model to label each instance in the training set with soft labels (the output vector from the teacher model), using softmax at temperature $T$. Finally, training the distilled model on the soft labels from the teacher model, again using softmax at temperature $T$.

*Thermometer Encoding [37]:* The purpose of thermometer encoding is to break the linear manner of the neural networks. Given an image $x$, for each pixel color $x_{(i,j,c)}$, the $l$-level thermometer encoding $\tau(x_{(i,j,c)})$ is a $l$-dimensional vector

$$\tau\big(x_{(i,j,c)}\big) = \begin{cases} 1, & \text{if } x_{(i,j,c)} > k/l \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

For example, for a 10-level thermometer encoding, we had $\tau(0.57) = 1111100000$. Then we use thermometer encoding to train a model.

## C. Evaluation

Generally, the evaluation procedure is the same as that in Section III-E. In all the evaluations of this section, we employ MNIST to randomly generate CAPTCHAs of length 6. For each scenario, we use 1000 CAPTCHAs for testing. When generating an adversarial CAPTCHA, we set the inner $8 \times 8$ area as the low-frequency part while the rest as the high-frequency part for mask $\varphi$. Each evaluation is repeated three times and their average is reported as the final result.

Specifically, we use MaxoutNet to generate adversarial CAPTCHAs. For ensemble adversarial training, we use MaxoutNet, NetInNet and LeNet to generate adversarial examples by JSMA$^f$, $L_2^f$, $L_0^f$ and $L_\infty^f$, respectively, and use these examples to train a LeNet model. In Table VII, *EnAdv. Training* means we do not use adversarial examples crafted on MaxoutNet, while *EnAdv. Training*$^+$ do. For defense distillation, we set $T$ as 100 which is the strong defense setting. For Thermometer Encoding, we set $l$ as 16 which is the same

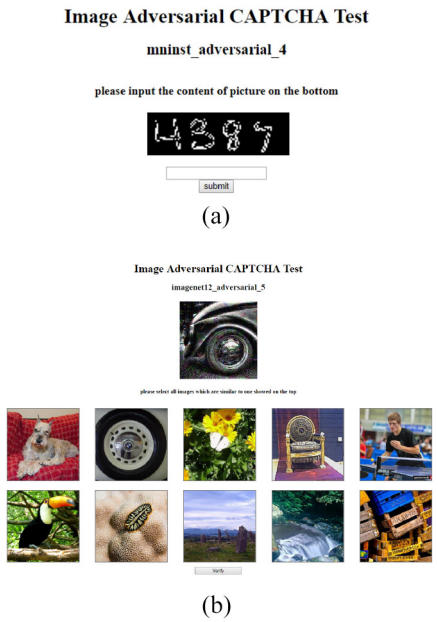



Fig. 3. Examples of aCAPTCHA. Text-based CAPTCHA is generated by JSMA$^f$ and image-based one is generated by JSMA$^i$ using $K = 50$. (a) Text-based adversarial CAPTCHAs. (b) Image-based adversarial CAPTCHAs.

as the original paper. In addition, image binarization is used in all of the tests.

The results are shown in Table VII, from which we make the following observations. First, defense distillation which is based on gradient masking cannot break adversarial captchas. The result is consistent with previous work [38] that gradient masking is not an effective solution against black-box adversarial examples. Second, thermometer encoding shows limited value to recognize adversarial examples. This may be due to the large perturbation we injected. Third, ensemble adversarial training largely improves the SAR, especially in the *EnAdv. Training*$^+$ setting. However, considering the SAR of LeNet (95.87%) for normal CAPTCHAs, adversarial CAPTCHAs are still more secure than normal ones. Moreover, in practice, attackers are hard to know what methods and models are employed in adversarial CAPTCHA generation. Attackers need to train an attack model by their own data, and then use their own methods to generate adversarial CAPTCHAs. As a result, the adversarial CAPTCHAs generated by attackers might be far different from our adversarial CAPTCHAs, which further restricts the practical performance of adversarial training. Overall, the generated adversarial CAPTCHAs are resilient to state-of-the-art defense methods.

## VI. Usability Evaluation

We have examined the security performance of aCAPTCHA from multiple perspectives in Sections III–V, respectively. In this section, we conduct experiments to evaluate the usability performance of aCAPTCHA. As in the security evaluation, we employ MNIST and ImageNet ILSVRC-2012 to generate normal and adversarial CAPTCHAs for the text- and image-based scenarios, respectively.

TABLE VIII
User Statistics

| | | | | | |
|---|---|---|---|---|---|
| gender | female | | | male | |
| | 43 | | | 82 | |
| age | [16-20] | [21-30] | [31-40] | [41-50] | [51-60] |
| | 76 | 40 | 1 | 3 | 5 |
| education | primary school | high school | B.S. | M.S. | Ph.D. |
| | 1 | 17 | 85 | 12 | 10 |

### A. Settings and Methodology

To evaluate the usability of aCAPTCHA, we set the baseline as the usability of normal text- and image-based CAPTCHAs.

*Methodology:* To conduct our evaluation, we construct a real-world website, on which the evaluation webpage is self-adapted to both PC and mobile clients, to deploy normal and adversarial CAPTCHAs and collect the evaluation data. The visualization of adversarial CAPTCHAs used in test are shown in Fig. 3. Then, we recruit volunteer users from the campus of our university to do the evaluation. Due to the space limitation, we have omitted specific evaluation steps.

### B. Results and Analysis

After moving the usability evaluation website online, we finally recruit 125 volunteer users as shown in Table VIII. Specifically, the users include 43 females and 82 males, and most of them have ages ranging from 16 to 30. Furthermore, almost all the users' education levels are high school or higher. This is mainly because we conduct experiments in the campus. Following the evaluation procedure, all the 125 users successfully finished the evaluation ($\sim$ 90% users finish the evaluation through smart phones). We then collect all the results to our server.

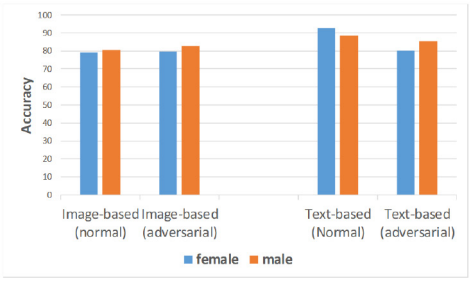
Based on the collected data, we show the main results in Table IX, where $\iota$ denotes the length of a text-based CAPTCHA, $K$ indicates the noise (difficulty) level of an image-based adversarial CAPTCHA, and *success rate*, *average time*, and *median time* measure the average successful probability, the average time consumption, and the median time consumption of all the users to finish the corresponding task, respectively. From Table IX, we have the following observations.

For text-based CAPTCHAs, although the adversarial versions can significantly improve the security performance as shown in Section III, their success rate of recognition also maintains a high level, which is only slightly lower than that of the normal versions. Meanwhile, it takes similar time for users to recognize normal and adversarial CAPTCHAs. These results suggest that text-based adversarial and normal CAPTCHAs have similar usability. In addition, given that long CAPTCHAs usually have better security than short ones [6], we also find that long text-based CAPTCHAs cost more time for recognition and have a lower success rate than that of the short ones (consistent with our intuition). This implies that there is a tradeoff between security and usability.

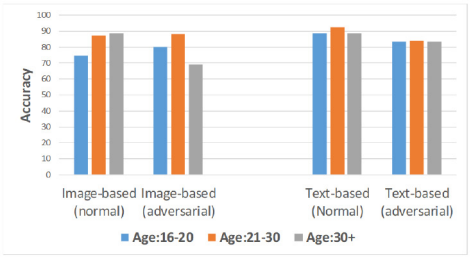
For image-based CAPTCHAs, the advantage of adversarial versions is more evident. Adversarial CAPTCHAs have similar or even better success rates as the normal ones in all the

TABLE IX
USABILITY OF ACAPTCHA

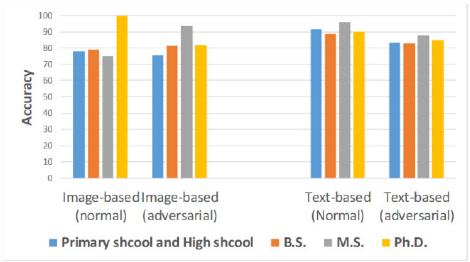
| | Text-based CAPTCHAs | | | | Image-based CAPTCHAs | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Normal | | Adversarial | | Normal | Adversarial | | | | |
| | $\iota = 4$ | $\iota = 6$ | $\iota = 4$ | $\iota = 6$ | | $K = 10$ | $K = 20$ | $K = 30$ | $K = 40$ | $K = 50$ |
| Success rate | 92.8% | 87.2% | 88.0% | 82.2% | 80.0% | 79.2% | 81.6% | 80.0% | 80.8% | 80.4% |
| Average time | 8.6s | 9.7s | 8.6s | 10.2s | 19.7s | 15.3s | 12.3s | 12.8s | 11.8s | 11.5s |
| Median time | 7.1s | 7.8s | 6.2s | 8.4s | 16.0s | 10.9s | 9.4s | 9.4s | 8.8s | 9.4s |




(a)




(b)




(c)

Fig. 4. Success rate versus statistical category. (a) Success rate versus gender. (b) Success rate versus age. (c) Success rate versus education.

cases. The success rates of adversarial CAPTCHAs with different noise (difficulty) levels are also similar. This suggests that image-based CAPTCHAs are more robust to adversarial perturbations. Given the obvious security advantage shown in Section IV, image-based adversarial CAPTCHAs are more promising compared to normal ones. Another interesting observation is that adversarial CAPTCHAs cost less time for recognition than the normal versions, which is a little bit out of our expectation. We conjecture the reasons as follows: 1) deliberately adversarial perturbation has little impact on the quality of images with respect to human recognition and 2) as the evaluation goes on, users become more and more familiar with the tasks. Thus, they can finish the tasks faster.

Now, we give a close look at the success rate of different users based on their statistical categories. The results are shown in Fig. 4. From Fig. 4, we can see that, in most of the scenarios, users from different statistical categories exhibit similar success rate over both adversarial and normal CAPTCHAs. This further demonstrates the generality of aCAPTCHA.

In summary, according to our evaluation, the CAPTCHAs generated by aCAPTCHA, especially the image-based adversarial CAPTCHAs, have similar usability as the normal versions. Recall the security evaluation of aCAPTCHA in Sections III and IV, they together demonstrate that aCAPTCHA is promising in addressing the dilemma of existing text- and image-based CAPTCHAs.

## VII. DISCUSSION

*Remarks on aCAPTCHA:* Different from traditional CAPTCHA designs, which are mainly focusing on defending against attacks in a *passive* manner, we design aCAPTCHA following a more proactive principle: to set one's own spear against one's own shield. Then, in terms of the model of state-of-the-art CAPTCHA attacks, we designed and implemented text- and image-based adversarial CAPTCHAs.

When implementing adversarial CAPTCHAs, we also follow a different methodology from that of existing adversarial image generation techniques. The main reason, as we discussed before, is because we stand on a different position. Existing adversarial image generation techniques focus on attacks in a hidden manner. For instance, some method may focus on generating an adversarial image which is only different from the original image in one pixel [18] (it is impossible for humans to identify such difference). In contrast, we follow the rule to inject as much perturbation as possible when the adversarial CAPTCHAs remain human tolerable. In this way, we would find a better balance between CAPTCHA security and usability, which can be demonstrated by our evaluation results.

One thing deserves further emphasis is that: aCAPTCHA is not designed as a replacement while is designed as an enhancement of existing CAPTCHA systems. According to our design, aCAPTCHA can be seamlessly combined with the deployed text- and image-based CAPTCHA systems. The only change is to update the normal CAPTCHAs with their adversarial versions. Therefore, we believe aCAPTCHA has a great applicability. Actually, we have contacted with several Internet companies to introduce aCAPTCHA. They are all very interested with aCAPTCHA and two of them have shown the intension to integrate aCAPTCHA to their systems.

In the design of aCAPTCHA, we only integrate the popular attacks to text- and image-based CAPTCHAs. Also, following our design principle, we propose and implement four text-based and four ICG methods, respectively. Note that,

all these designs and implementations are for demonstrating the advantages of adversarial CAPTCHAs. Furthermore, aCAPTCHA employs a modular design style, which is easy for new technique integration. Hence, we will add more attacks as well as more adversarial CAPTCHA generation methods to aCAPTCHA, especially the emerging techniques. We believe the open source nature will facilitate the improvement process of aCAPTCHA.

## VIII. Related Work

Zhang *et al.* [34] studied the effect of adversarial examples on CAPTCHA robustness (including text- and image-based CAPTCHAs). They directly used the existing adversarial example generation techniques to generate adversarial CAPTCHAs. Their experimental results demonstrate that adversarial examples have a positive effect on the robustness of CAPTCHAs. However, they did not consider the image preprocess operations which may used by attackers. In this article, we consider many widely used IPP operations for CAPTCHAs. Our study shows that for text-based CAPTCHAs, the perturbations injected by the existing adversarial example generation techniques can be easily removed by IPP operations. Therefore, aiming at generating more robust and usable text-based adversarial CAPTCHAs, we propose four new methods based on existing techniques. The experimental results show that our methods can keep the balance between usability and security for text-based CAPTCHAs.

Osadchy *et al.* [43] introduced a new image-based CAPTCHA scheme which is designed to resist machine learning attacks. It adds immutable adversarial noise (IAN) to the correctly classified images that deceive deep learning tools and cannot be removed using image filtering. However, DeepCAPTCHA is different from our approach. In general, DeepCAPTCHA is a new type of image-based CAPTCHA scheme which could provide high security. While our aCAPTCHA system is designed to enhance the existing CAPTHCA schemes. Furthermore, the proposed IAN, which is resistance to filtering attack, cannot be used in text-based CAPTCHA generation. In this work, we consider more state-of-the-art adversarial example defense strategies and propose several new methods to generate text- or image-based adversarial CAPTCHAs.

Ye *et al.* [33] proposed a GAN-based approach to break text-based CAPTCHAs. In particular, they first generated synthetic CAPTCHAs to learn a base solver and then fine-tuned the base solver on a small set of real CAPTCHAs by leveraging transfer learning. From the evaluation, their method can achieve good recognition performance with a significantly smaller set of real captchas, as compared to previous methods. Despite this method can reduce the cost of attack in term of labeling data, it cannot effectively break adversarial CAPTCHAs. This is because the solver used in [33] is a CNN model which is still vulnerable to adversarial CAPTCHAs. Our study show that adversarial CAPTCHAs can effectively defend against the attack models which are completely trained by a real dataset, let alone the model trained by a synthetic dataset. For the fine-tune process, it is similar to the process of adversarial training in Section V: using the generated adversarial CAPTCHAs to retain the original attack model. However, the results from Table VII in this article show that the adversarial CAPTCHAs are resilient to adversarial training.

## IX. Conclusion

In this article, we study the generation of adversarial CAPTCHAs. First, we propose a framework for generating text- and image-based adversarial CAPTCHAs. Then, we design and implement aCAPTCHA, a comprehensive adversarial CAPTCHA generation and evaluation system, which integrates 12 IPP techniques, nine CAPTCHA attacks, four baseline adversarial CAPTCHA generation methods, and eight new adversarial CAPTCHA generation methods, and can be used for the generation, security evaluation, and usability evaluation of adversarial CAPTCHAs. To evaluate the performance of aCAPTCHA, we conduct extensive experiments. The results demonstrate that the adversarial CAPTCHAs generated by aCAPTCHA can significantly improve the security of normal CAPTCHAs while maintaining similar usability. Finally, we open source aCAPTCHA to facilitate the CAPTCHA security research.

## Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable suggestions for improving this article.

## References

[1] *CAPTCHA*. Accessed: Aug. 7, 2018. [Online]. Available: https://en.wikipedia.org/wiki/CAPTCHA

[2] *CAPTCHA: Telling Humans and Computers Apart Automatically*. Accessed: Aug. 7, 2018. [Online]. Available: http://www.captcha.net/

[3] *The MNIST Database*. Accessed: Aug. 7, 2018. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[4] *ImageNet*. Accessed: Aug. 7, 2018. [Online]. Available: http://www.image-net.org/

[5] *Model Zoo*. Accessed: Aug. 7, 2018. [Online]. Available: https://github.com/BVLC/caffe/wiki/Model-Zoo

[6] E. Bursztein, M. Martin, and J. C. Mitchell, "Text-based CAPTCHA strengths and weaknesses," in *Proc. ACM CCS*, 2011, pp. 125–138.

[7] H. Gao *et al.*, "A simple generic attack on text captchas," in *Proc. NDSS*, 2016, pp. 1–14.

[8] K. A. Kluever and R. Zanibbi, "Balancing usability and security in a video CAPTCHA," in *Proc. SOUPS*, 2009, pp. 1–11.

[9] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. 1st IEEE Eur. Symp. Security Privacy*, 2016, pp. 372–387.

[10] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. 1st IEEE Eur. Symp. Security Privacy*, 2017, pp. 39–57.

[11] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[12] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proc. ICML*, 2013, pp. 1319–1327.

[13] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proc. ICLR*, 2014, pp. 1–12.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: arXiv:1409.1556.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. ICCV*, 2015, pp. 1026–1034.

[16] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. CVPR*, 2015, pp. 1—9.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016, pp. 770–778.

[18] J. Su, D. V. Vargas, and S. Kouichi, "One pixel attack for fooling deep neural networks," 2017. [Online]. Available: arXiv:1710.08864.

[19] X. Ling *et al.*, "DEEPSEC: A uniform platform for security analysis of deep learning model," in *Proc. IEEE Symp. Security Privacy*, 2019, pp. 673–690.

[20] E. Bursztein, J. Aigrain, A. Moscicki, and J. C. Mitchell, "The end is nigh: Generic solving of text-based CAPTCHAs," in *Proc. WOOT*, 2014, p. 3.

[21] A. S. El Ahmad, J. Yan, and M. Tayara, "The robustness of google CAPTCHAs," Sch. Comput. Sci., Newcastle Univ., Newcastle upon Tyne, U.K., Rep. CS-TR-1278, 2011.

[22] J. Yan and E. S. El Ahmad, "A low-cost attack on a microsoft CAPTCHA," in *Proc. CCS*, 2008, pp. 543–554.

[23] K. Chellapilla and P. Y. Simard, "Using machine learning to break visual human interaction proofs (HIPs)," in *Proc. NIPS*, 2005, pp. 265–272.

[24] H. Gao, W. Wang, J. Qi, X. Wang, X. Liu, and J. Yan, "The robustness of hollow CAPTCHAs," in *Proc. CCS*, 2013, pp. 1075–1086.

[25] M. Chew and J. D. Tygar, "Image recognition captchas," in *Proc. ICIS*, 2004, pp. 268–279.

[26] E. Jeremy, D. John, H. Jon, and S. Jared, "Asirra: A CAPTCHA that exploits interest-aligned manual image categorization," in *Proc. CCS*, 2007, pp. 366–374.

[27] B. B. Zhu *et al.*, "Attacks and design of image recognition CAPTCHAs," in *Proc. CCS*, 2010, pp. 187–200.

[28] S. Suphannee, I. Polakis, and A. D. Keromytis, "I am robot: (Deep) learning to break semantic image captchas," in *Proc. EuroS&P*, 2016, pp. 388–403.

[29] H. Gao, H. Liu, D. Yao, X. Liu, and U. Aickelin, "An audio CAPTCHA to distinguish humans from computers," in *Proc. ISECS*, 2010, pp. 265–269.

[30] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," 2018. [Online]. Available: arXiv:1801.00553.

[31] C. Szegedy *et al.*, "Intriguing properties of neural networks," 2013. [Online]. Available: arXiv:1312.6199.

[32] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," 2017. [Online]. Available: arXiv:1712.07107.

[33] G. Ye *et al.*, "Yet another text captcha solver: A generative adversarial network based approach," in *Proc. CCS*, 2018, pp. 332–348.

[34] Y. Zhang, H. Gao, G. Pei, S. Kang, and X. Zhou, "Effect of adversarial examples on the robustness of CAPTCHA," in *Proc. Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Disc. (CyberC)*, 2018, pp. 1–109.

[35] N. Papernot, P. D. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Security Privacy*, 2016, pp. 582–597.

[36] F. Tramèr, A. Kurakin, N. Papernot, I. J. Goodfellow, D. Boneh, and P. D. McDaniel "Ensemble adversarial training: Attacks and defenses," in *Proc. ICLR*, 2018, pp. 1–20.

[37] J. Buckman, A. Roy, C. Raffel, and I. J. Goodfellow, "Thermometer encoding: One hot way to resist adversarial examples," in *Proc. ICLR*, 2018, pp. 1–22.

[38] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," 2017. [Online]. Available: arXiv:1705.07263.

[39] P. Golle, "Machine learning attacks against the Asirra CAPTCHA," in *Proc. CCS*, 2008, pp. 535–542.

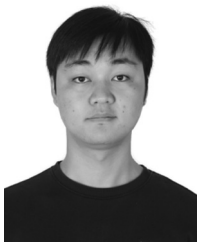
[40] Y. Xu, G. Reynaga, S. Chiasson, J.-M. Frahm, F. Monrose, and P. Van Oorschot, "Security and usability challenges of moving-object CAPTCHAs: Decoding codewords in motion," in *Proc. USENIX Security*, 2012, p. 4.

[41] S. K. Saha, A. K. Nag, and D. Dasgupta, "Human-cognition-based CAPTCHAs," *IT Prof.*, vol. 17, no. 5, pp. 42–48, 2015.

[42] R. Gonzalez, *Digital Image Processing*. Upper Saddle River, NJ, USA: Pearson Hall, 2008.

[43] M. Osadchy, J. Hernandez-Castro, S. Gibson, O. Dunkelman, and D. Pérez-Cabo "No bot expects the DeepCAPTCHA! Introducing immutable adversarial examples, with applications to CAPTCHA generation," *IEEE Trans. Inf. Forensics Security*, vol. 12, pp. 2640–2653, 2017.

[44] *aCAPTCHA*. Accessed: Aug. 7, 2018. [Online]. Available: https://github.com/ChenghuiS/aCAPTCHA

**Chenghui Shi** received the bachelor's degree in electronic and information engineering from the University of Shanghai for Science and Technology, Shanghai, China. He is currently pursuing the Ph.D. degree with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China.

His current research interests include adversarial learning and AI security.

**Xiaogang Xu** received the bachelor's degree in information engineering from Zhejiang University, Hangzhou, China, in 2018. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, the Chinese University of Hong Kong, Hong Kong.

His research interests focus on computer vision and data-driven security.

**Shouling Ji** (Member, IEEE) received the first Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, and the second Ph.D. degree in computer science from Georgia State University, Atlanta.

He is a ZJU 100-Young Professor with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China, and a Research Faculty with the School of Electrical and Computer Engineering, Georgia Institute of Technology. His current research interests include AI security, data-driven security, privacy, and data analytics.

Dr. Ji is a member of ACM and was the Membership Chair of the IEEE Student Branch at Georgia State from 2012 to 2013.

**Kai Bu** (Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2006 and 2009, respectively, and the Ph.D. degree in computer science from The Hong Kong Polytechnic University, Hong Kong, in 2013.

He is currently an Associate Professor with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. His research interests include network security and computer architecture.

Dr, Bu was a recipient of the Best Paper Award of IEEE/IFIP EUC 2011 and the Best Paper Nominee of IEEE ICDCS 2016. He is a member of the ACM and the CCF.

**Jianhai Chen** (Member, IEEE) received the M.S. and Ph.D. degrees in computer science and technology from Zhejiang University, Hangzhou, China.

He is currently an Associate Professor with the College of Computer Science and Technology, Zhejiang University. His research interests include blockchain security, high performance computing, and virtualization and cloud computing.

Dr. Chen is a member of CCF and ACM.

**Raheem Beyah** (Senior Member, IEEE) received the Bachelor of Science degree in electrical engineering from North Carolina A&T State University, Greensboro, NC, USA, in 1998, and master's and Ph.D. degrees in electrical and computer engineering from Georgia Tech, Atlanta, GA, USA, in 1999 and 2003, respectively.

He is the Motorola Foundation Professor and the Associate Chair with the School of Electrical and Computer Engineering, Georgia Tech, where he leads the Communications Assurance and Performance Group and is a member of the Communications Systems Center (CSC). Prior to returning to Georgia Tech, he was an Assistant Professor with the Department of Computer Science, Georgia State University, Atlanta; a Research Faculty Member with the Georgia Tech CSC; and a Consultant with Andersen Consulting's (currently Accenture) Network Solutions Group, Dublin, Ireland. His research interests include network security, wireless networks, network traffic characterization and performance, and critical infrastructure security.

Dr. Beyah received the National Science Foundation CAREER Award in 2009 and was selected for DARPA's Computer Science Study Panel in 2010. He is a member of AAAS and ASEE, is a Lifetime Member of NSBE, and is a Senior Member of ACM.

**Ting Wang** (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2011.

He is currently an Assistant Professor with the College of Information Sciences and Technology, Penn State University, State College, PA, USA. His current work focuses on making AI systems more practically usable through improving their security assurance, privacy preservation, and decision-making transparency.