Enhancing the Security of Image CAPTCHAs Through Noise Addition

David Lorenzi, Emre Uzun, Jaideep Vaidya, Shamik Sural, Vijayalakshmi Atluri

Enhancing the Security of Image CAPTCHAs through Noise Addition

David Lorenzi[1], Emre Uzun[1] Jaideep Vaidya[1], Shamik Sural[2], and Vijayalakshmi Atluri[1]

[1] Rutgers University, Newark NJ 07102, USA
dlorenzi,emreu,jsvaidya,atluri@cimic.rutgers.edu,
[2] Indian Institute of Technology, School of Information Technology, Kharagpur, India
shamik@sit.iitkgp.ernet.in

Abstract. Text based CAPTCHAs are the de facto method of choice to ensure that humans (rather than automated bots) are interacting with websites. Unfortunately, users often find it inconvenient to read characters and type them in. Image CAPTCHAs provide an alternative that is often preferred to text-based implementations. However, Image CAPTCHAs have their own set of security and usability problems. A key issue is their susceptibility to Reverse Image Search (RIS) and Computer Vision (CV) attacks. In this paper, we present a generalized methodology to transform existing images by applying various noise generation algorithms into variants that are resilient to such attacks. To evaluate the usability/security tradeoff, we conduct a user study to determine if the method can provide "usable" images that meet our security requirements – thus improving the overall security provided by Image CAPTCHAs.

1 Introduction

CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart) are now ubiquitously found on the web to ensure that the entity interacting with a website is indeed human. While CAPTCHAs ensure that abuse of online forms is reduced, web users are forced to suffer through increasingly convoluted and unfriendly CAPTCHAs that negatively impact their user experience. Text-based CAPTCHAs are the most common implementation in use, due to their scalability, robustness, and ease of implementation. However, given their prevalence, many techniques have been developed to break such CAPTCHAs. As a result, alternative methods of form control and human verification have been sought for by the research community. Among the several different modalities that have been explored, image based CAPTCHAs have emerged as a plausible alternative, more suitable for the smartphone/mobile touch-capable environment. However, image CAPTCHAs come with their own set of problems, particularly in terms of scalability – it is hard to find large quantities of labeled/tagged images; and robustness – there is limited variation in the challenge question and vulnerability to single style of attack. In particular, reverse image search (RIS) has emerged as a particularly insidious type of attack against image CAPTCHAs [10]. In this paper, we propose a generalized

methodology to transform existing images into more resilient variants. The basic idea is to introduce noise into the images using various noise generation algorithms which make it difficult to automatically retrieve the exact same image from the web, while still allowing humans to extract the key concepts from the image to correctly answer the CAPTCHA. Through this transformation, image CAPTCHAs can again become a viable alternative to text based CAPTCHAs having a similar level of security while providing a superior level of usability.

One interesting aspect of using noise generation algorithms to secure images is that the images produced by the algorithms we selected are very "grainy" or "pixelated" in appearance – very similar to a snowy TV picture. The noise introduced is primarily additive and multiplicative in nature, thus it tends to shift around color values in various pixels based on a threshold of our choosing. The benefit to this noise is demonstrated when the image is viewed as a matrix of numbers (as a computer would "see" the image), the values vary wildly and do not follow the patterns typical of a structured image. However, when viewed by a human eye (along with a human mind behind it), the colors blend into an image that is coherent and cognizable (the "Pointillism effect"). Strangely, this side effect of enhancing security actually does not impact usability negatively (to a point). In general, this effect is easier to achieve the further away your eye is from the image, or if the image is small in dimensions (scaled down). In honor of one of the co-creators of the Pointillism technique, 19th century French neo-impressionist painter Paul Signac, we have named our procedure SIGNAC (Secure Image Generation Noise Algorithms for CAPTCHAs).

Our method is similar to the concept of "emergence" - which refers to the unique human ability to aggregate information from seemingly meaningless pieces, and to perceive a whole that is meaningful [15]. The image CAPTCHA utilization of this idea relies on the absence of meaningful information in local image parts which largely hinders existing computer vision algorithms from recognizing emerging figures [9]. The difference between our proposed SIGNAC method and emergence is that we are using an original image (I) and altering it to a new image (I') through a series of image transformations and alterations. We believe this provides a stronger "concrete" foundation for the images generated by the SIGNAC method than would be created by emerging images without giving away too many clues to segmentation/edge detection algorithms.

The rest of the paper is structured as follows. In Section 2 we review the related work. Section 3 details attacks and defense strategies. Section 4 presents the proposed methodology. Sections 5 and 6 present the experimental evaluation and the usability study respectively. Finally, Section 7 concludes the paper.

2 Related Work

From the first introduction of the CAPTCHA [17], there has been significant work on categorizing and creating different CAPTCHA challenges based on alternate modalities such as images [2]. Usability is often a key challenge – Yan et al. [18] provide a general framework for evaluating usability. In recent years, the complexity of text-based CAPTCHAs has been steadily increasing to provide robustness against attacks, but also hampering their usability. As mentioned above,

this has led to alternatives such as image based CAPTCHAs. However, given the continuous improvement in computer vision algorithms, image CAPTCHAs have also become vulnerable to attack from methods such as edge detection, object recognition, or pattern recognition. As demonstrated by several studies [5, 19, 11, 8, 6], newly deployed real world CAPTCHAs frequently do not take into account advances in computer vision research literature and make common mistakes that could have been accounted for during their design phase.

Newer CAPTCHAs leverage the power of images to exploit the human-machine gap. For example, image orientation [7] is comparatively easy for humans but difficult for computers. Similarly, "scene tagging" [13] tests the ability to recognize a relationship between multiple objects in an image that is automatically generated via composition of a background image with multiple irregularly shaped object images. IMAGINATION [4] also uses a similar "image composition" method. Image distortion [14] has also been suggested as an alternative. One interesting recent invention is that of the GOTCHA [1], which is essentially a randomized puzzle generation protocol, which produces unique images similar to inkblot tests. While questions about its usability remain, it is a promising emerging avenue of anti-bot security methods research.

The main enemy of modern image CAPTCHAs that attempt to work at scale is the modern Content Based Image Retrieval (CBIR) engine [3]. Most major search engines today offer some CBIR capability with their "image search" feature, usually in the form of image retrieval, which can allow an attacker to find an exact match (Reverse Image Search) or other uses of an image on the web that have been used in a CAPTCHA challenge. One subset of CBIR, Automated Image Annotation, provides an extreme threat to simple naming image CAPTCHAs as it can provide an automated answer in the form of a tag for what is presented in the image. Recently, Lorenzi et al. [10] have shown how CBIR and ALA can be used to break several modern image CAPTCHAs.

3 Attack Methods and Defense Strategies

We now discuss two particular types of attack – Reverse Image Search (RIS) engine attacks and Computer Vision (CV) attacks. These are particularly strong against image CAPTCHAs. We also discuss the general defense strategy of adding noise to the image to make it more robust to these attacks. As with all CAPTCHAs, we are again faced with the challenge of balancing security with usability. Since we are utilizing a noise addition method, the image cannot be altered to the degree that a human observer loses the ability to recognize the content of the image (rendering it useless for our purposes).

3.1 Stopping Image Search Attacks

First, our noise addition algorithms must stop reverse image search engines from finding image matches indexed online (Google image search[3] and Tineye [4]). This is an important security enhancement as image CAPTCHAs traditionally have

[3] https://www.google.com/img
[4] https://www.tineye.com/

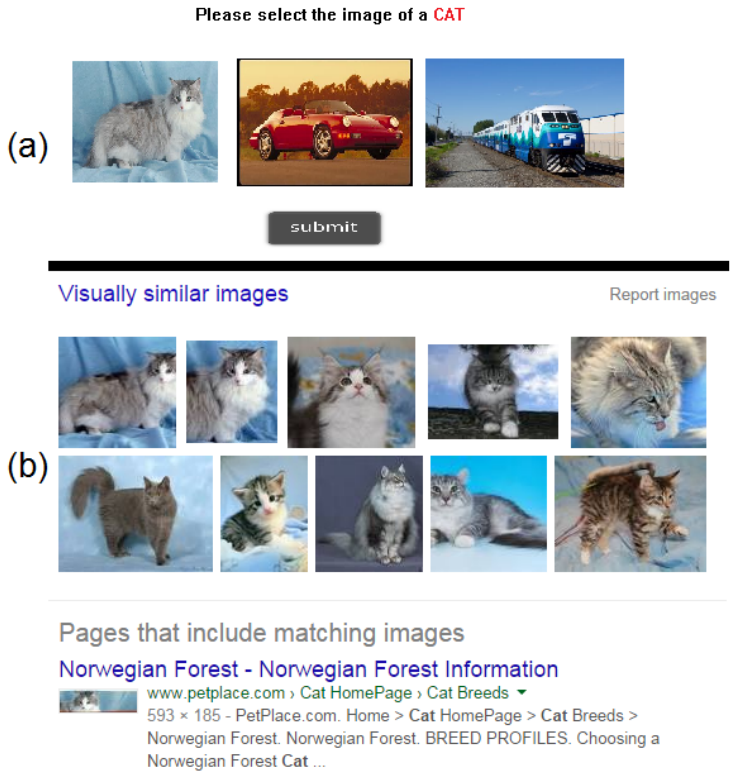

Fig. 1: Reverse image search attack with metadata. (a) depicts the CAPTCHA images without noise, (b) depicts results of a Google image search

problems in defending against database attacks and tag matching attacks, which can be viewed as a scalability issue (too few unique images). The following scenario is an example of an RIS attack in action: Imagine an image based CAPTCHA challenge asking the user to identify which image out of a set of images depicts a cat as shown in Figure 1. The attacker then: 1) Makes a copy of the images from the CAPTCHA 2) Runs them through an RIS engine to find exact matches 3) Scrapes and stores the metadata from the RIS engine 4) Uses Regular Expressions to match the keyword "cat" to the search that locates a copy of the image used somewhere else online with the filename "cat.jpg", which happened to be found on a website with the URL that contains the word "cat" e.g. http://www.coolcatpics.com.

At this point, the attacker can probabilistically determine which image is most likely the cat image (or eliminate the other image choices through the same process). The bad news for those attempting to develop security measures against RIS engine attacks is that the engines themselves are proprietary (trade secret) and closed source, forcing the CAPTCHA security developer to devise a set of experiments that attempt to probe a "black box" to learn its behavior.

The good news is that the RIS engines are available for use by the public with reasonable limits established (50 test images per day, up to 150 per week), and a security expert with access to or knowledge of "image fingerprinting" and image processing literature can use this body of knowledge to provide clues for educated guesses as to the methods that RIS engines are utilizing to identify matches. The noise generation method we propose works on the premise of introducing an amount of noise such that the image used for a CAPTCHA challenge has been altered enough from the original that the various "image fingerprinting" metrics used to determine matches have been "tricked" - that is they no longer see the image as a match as its information diverges from the original image beyond their threshold/similarity metric. Technically speaking, the image returned by the method is a different image, as the noise changes the values of the pixels in the image. A distance metric (change from original) is useful to model the noise alterations from original image to new image. However, the new image (post-noise) is still functionally depicting the same content as the original, albeit in a degraded fashion. Stopping RIS engines from finding matches means indexed images can be used as CAPTCHA challenges again, increasing the sample space of potential usable images significantly.

3.2 Stopping Computer Vision Attacks

Second, the noise algorithms must be able to alter the image enough to hinder or stop altogether, general image/object recognition algorithms that would attempt to solve image recognition challenges.

One popular CV algorithm is SIFT [12], which stands for Scale-Invariant Feature Transform. While it has previously been used in many applications, we are interested only in its ability to perform object recognition tasks. ASIFT [16], which stands for affine-SIFT, is an improvement over SIFT. It considers the lattitude and longitude angles that are ignored by SIFT and then combines that information with SIFT to provide a more complete analysis than SIFT alone. As such, it significantly outperforms SIFT and is more of a challenge to defeat. By adding noise to the image, it should throw off the keypoints calculations so that when it compares two images, the noised image does not have the same keypoints and it fails to return a match. Note that the web application uses grayscales and resizes the images before the CV algorithm is run.

Another important point to consider is that we used an online service to perform the SIFT and ASIFT analysis [16]. The above computation could be completed in approximately 7 seconds through a web form. As more of these services move online, an attacker no longer needs to run local image matching or CV tools, and can script a live attack that pipes the CAPTCHA challenge through the appropriate tools to generate and even submit a correct response.

For example, in our aforementioned cat image scenario, imagine in this case the attacker decides to use image/object recognition with a CV toolkit. The attacker has trained and tested their algorithm of choice (e.g., SIFT) on various images of cats gathered from around the web and can recognize them with a good degree of accuracy. When he feeds the CAPTCHA challenge image into the algorithm, it returns a high probability of the image being of a cat. Using

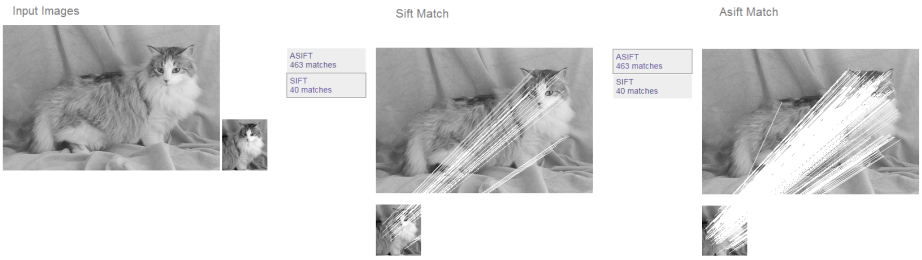

Fig. 2: CV Attack with SIFT & ASIFT

the noise generation algorithms, the image of the cat can be altered enough so that the CV algorithms return a low probability or cannot determine what the image is depicting, but a human can still determine it is showing a cat. The intention is to use the noise to distort the edges of scenes/objects and alter the patterns within the image enough such that various commonly used CV techniques fail to provide meaningful results for an attacker. Also image filters can be used to distort and move pixel neighborhoods such that detection and mapping algorithms fail to achieve matches and/or detect similarity. Figure 2 shows that both SIFT and ASIFT can overcome scaling issues (mappings are found to a smaller, cropped image of the cat), and ASIFT typically provides more mappings than SIFT.

4 Methodology

Our method is designed to work with existing image CAPTCHAs that rely on a database of images for challenges. After application of SIGNAC, we demonstrate that the same database of images provides better security against RIS and CV based attacks. The MATLAB image processing toolbox is used to generate the new secure images. The function imnoise is used to add noise to the images. The test image set contains 100 images in total, 10 images in each of 10 different categories: airplane, bird, car, cat, doll, fish, flower, monkey, robot, and train. The categories are deliberately made "concrete" instead of abstract, as this makes it easier to create naming and distinguishing image CAPTCHAs that will be straightforward for user/usability testing. This also provides the CV algorithms with an "object" to recognize.The noise functions utilized in the method are the four generalized noise functions available in the MATLAB IPT[5].

4.1 The SIGNAC Approach

As discussed above, SIGNAC is implemented using the MATLAB Image Processing Toolbox. The script below gives an idea of the method in action. X is the image at the initial starting point when it is read into the IPT. *c1* through *c5* represent the image at various stages of its alteration. Note that this example is a multimethod output, as different noise and filter functions are being used to generate an image at each step. It is important to note that ordinality plays a

[5] http://www.mathworks.com/help/images/ref/imnoise.html

large factor in the outcome of the image's success or failure in defeating an RIS engine, as discussed in the following section. This script is designed to create the image filter, read in the image file, apply noise, filter the image, then apply noise 3 more times before writing the image to a file.

```
f=fspecial('motion',11,3)
x=imread('1.jpg')
c1=imnoise(x,'salt & pepper',0.35)
c2=imfilter(c1,f)
c3=imnoise(c2,'speckle',0.35)
c4=imnoise(c3,'gaussian',0,0.35)
c5=imnoise(c4,'poisson')
imwrite(c5,'1', 'jpg');
```

This script represents the final script used to create the secure image set used in our experiments. Salt and pepper noise can be considered the most destructive type of noise, as it is the most extreme - changing pixel values to 0's and 1's. The motion filter is then applied to the image with a len of 11 pixels and a theta of 3 degrees counterclockwise, which serves to relocate the pixels that were changed with the addition of the salt and pepper noise to new areas around the image. This aids in obfuscation of clues about pixel values in a particular neighborhood, i.e., multiple pixels will now be distorted with values that differ from the original. After the filter is applied, multiplicative noise in the form of the speckle noise function distributes its noise in a uniform fashion throughout the image, followed by the addition of white Gaussian noise. The final step involves using the Poisson noise function, which does not add artificial noise, instead it generates noise from the image data and then applies it to the image using a Poisson distribution. This serves to further obfuscate the artificial noise that was added during previous steps by shifting the pixel values around.

4.2 RIS Engine Probing

Figure 3 shows an example of a single image test working against the RIS engine Tineye. For the original figure (3a), Tineye provides exact match results.

Single Noise Function, Single Stage Currently, the initial image returns 16 exact matches from across the web. These results were gathered using a single image noise function in a single step on the original image to produce an image that returns 0 exact matches. Note that these values are unique to this image, and vary based on the image properties.

Ordinality Test This test demonstrates the ordinality of noise functions. These tests were run with the default settings of each noise generation function to see if the order in which the functions are applied affects the results. Table 1 gives the results with the different orders. Note that these results were obtained using the original cat image, and these results apply only to that image. From these results, it is clear that order makes a difference as the range for matches is +-2 matches, with a high of 11 matches and a low of 9 matches. We then further investigate the chain of methods that produce the least amount of matches.

(a) RIS Probe Test Image (b) Gaussian Noise with 0.2 Mean (c) Salt & Pepper Noise with 0.3 Mean (d) Speckle Noise with 0.4 Mean

Fig. 3: RIS Engine Probing

Table 1: Results of Ordinality Test

Primary Noise Function	Permutations	# of Results	Primary Noise Function	Permutations	# of Results
Gaussian (G)	GKPS	10	Speckle (K)	KGPS	09
	GKSP	10		KGSP	11
	GPKS	11		KPGS	10
	GPSK	10		KPSG	10
	GSKP	10		KSGP	11
	GSPK	10		KSPG	11
Salt and Pepper (S)	SGKP	11	Poisson (P)	PGKS	10
	SGPK	11		PGSK	09
	SKGP	09		PKGS	10
	SKPG	10		PKSG	10
	SPGK	11		PSGK	10
	SPKG	09		PSKG	09

Threshold Determination After deciding on an appropriate ordinality for noise methods, it must be determined the minimum threshold at which zero matches are reached - the key to our anti-RIS security criterion. A rough metric is used first, incrementing each mean value by 0.1 until zero matches are found. Then it decrements by 0.05 until a match and then increments or decrements by 0.01 until the minimal value is reached with zero matches. Figure 4 shows two scripts that embody these principles in action. Note that both scripts provide zero matches, however, the second script produces a clearer image because less noise overall is added during the application of additional functions. This is important for usability reasons - as the clearer the image is, the easier the chance a real human will have in recognizing what it depicts.

However, the values are extremely sensitive. For example, decrementing the mean in the initial noise function of the previous script by 0.01 to 0.1 produced 5 matches. Decrementing both means c3 and c4 by 0.01 each produced 8 matches. It is a painstaking and involved process to tune each image for a working minimum. Unfortunately, this process must be done for each image on an individual basis and cannot be generalized beyond offering a rough threshold for which any series will return zero matches, and this threshold is usually quite high and may impact usability.

```
x=imread('cat.jpg')
c1=imnoise(x,'salt & pepper',0.11)
c2=imnoise(c1,'poisson')
c3=imnoise(c2,'speckle',0.11)
c4=imnoise(c3,'gaussian',0,0.11)
imshow(c4)
```
(a) Minimal Equal Noise via (SPKG)

```
x=imread('cat.jpg')
c1=imnoise(x,'salt & pepper',0.11)
c2=imnoise(c1,'poisson')
c3=imnoise(c2,'speckle',0.05)
c4=imnoise(c3,'gaussian',0,0.05)
imshow(c4)
```
(b) SPKG Current Working Minimum

Fig. 4: SIGNAC MATLAB Scripts

(a) Original Image Edge Detection

(b) Edge Detection after Noising

Fig. 5: Edge detection tests

As such, this script serves as the endpoint for security against RIS engines, as zero matches are returned with these values. Computer vision based attacks are an entirely different subject, and there are no guarantees that this RIS minimum will have any impact on the ability of CV tools to perform recognition tasks.

4.3 Noise for Anti-Computer Vision

While stopping RIS engines was the primary challenge, CV tools are powerful and have been successfully used to defeat image based CAPTCHAs in the past. Thus, we aim to make it as difficult as possible to use them in performing object recognition tasks. One such CV attack case is that of edge detection. This is a key component of object recognition, and being able to foil it will go a long way in stopping any CV attacks from performing this task on an image recognition CAPTCHA challenge.

In Figure 5a, we can see the results of a Sobel edge detection run on the test image. It clearly depicts a cat, while also picking up some of the wrinkles in the sheet behind the cat. Enough detail of the cat comes through that a CV algorithm could make a decision about what is depicted in the image. Note that when edge detection is performed, the image is first converted from RGB to grayscale, and then to binary (hence the black & white) after the edge detection algorithm is run. Figure 5b shows the same Sobel edge detection method run on the image of the cat that has been noised. It can be seen that the cat has completely disappeared - only white dots on a black background appear. No useful information can be gained from this image.

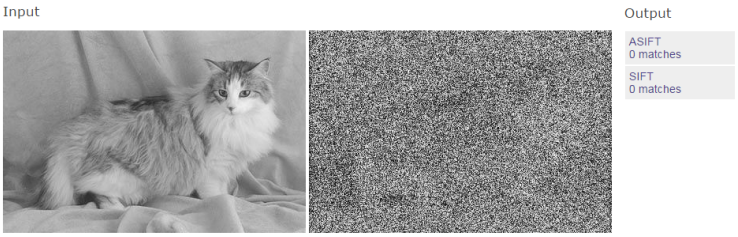

Fig. 6: SIFT & ASIFT image matching

Table 2: RIS Engine Testing

CategoryID	Category	Noise Functions at 0.25 Mean		Noise Functions at 0.30 Mean	
		Pass	Fail	Pass	Fail
1	airplane	T,G		T,G	
2	bird	T,G		T,G	
3	car	G	T22	T,G	
4	cat	T	G32	T,G	
5	doll	T,G		T,G	
6	fish	T	G57	T,G	
7	flower	T,G		T,G	
8	monkey	T	G77,G79	T,G	
9	robot		T84,T85,G81,G84,G85		T84,G84
10	train	T,G		T,G	

5 Experimental Results and Analysis

We now describe the experimental evaluation to test image security against both RIS engine attacks and CV attacks using the aforementioned online tools. We have gathered 100 random indexed images from 10 categories and applied the method described in Section 4. Note that the image filter values did not change during the course of the experiments, only the mean values of the noise functions.

5.1 RIS Engine Testing

The goal of this experiment was to establish a baseline for which a set of noise functions can provide zero exact matches against both Google (G) and Tineye's (T) reverse image search engines. As mentioned in the Methodology section, the approach we use is more conservative from the security perspective, in that many of the images are no longer returning matches at much lower levels of noise overall. We consider even 1 match a failure - thus we do not report specific numbers of matches for each image failure. The number following the search engine designation is the number in the set of 10 for that category, e.g., car contains 10 images total, numbered 21-30 - T22 means that image 22 failed to produce zero matches as matches were found on Tineye (but not Google).

Table 2 shows that at 0.25 mean noise, we have 8 out of 100 unique images returning matches. Tineye has one unique hit (T22) and Google has five unique hits (G32,G57,G77,G79,G81). There is overlap on image 84 and 85 as both engines returned matches. This means that out of our random sample of 100 indexed images, 92 out of 100 returned zero matches. At 0.30 mean noise, we

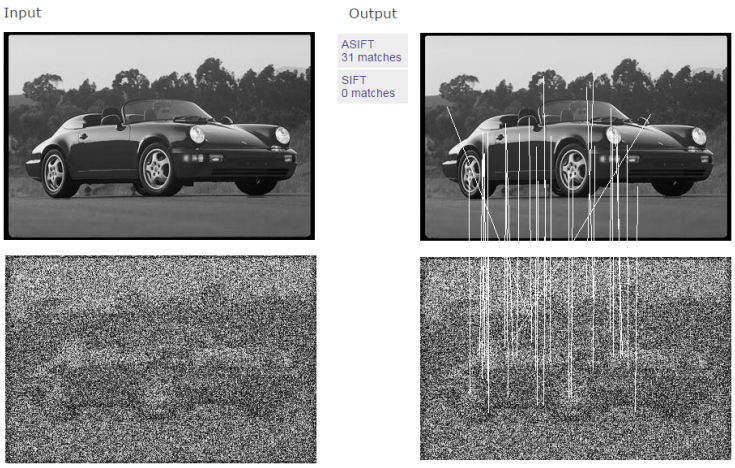

Fig. 7: Exact Match Test: Original Vs. Noise with false positives

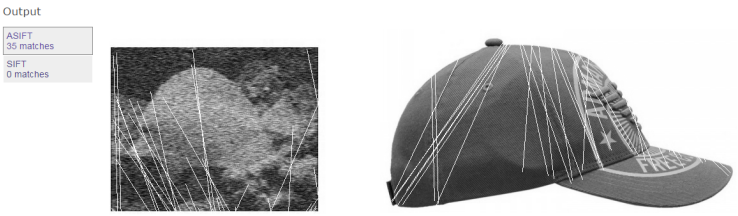

Fig. 8: Shape Test: Noised image with similar shape image

have 1 out of 100 unique images returning a match. Tineye and Google both return matches for the same image. This means that out of our random sample of 100 indexed images, 99 out of 100 returned zero matches.

5.2 Computer Vision Testing

In this section, we evaluate the effectiveness of SIFT and ASIFT to provide object detection and image matching. The key takeaway is to fool the keypoints calculator into examining incorrect correspondences by inflating the number of keypoints in an image or not finding any matches due to an insignificant matching value. Figure 6 demonstrates failure to find matching keypoints on an exact image match (original clear image vs. noised image).

It may be observed from Figure 7 that SIFT has returned zero matches, but ASIFT has returned 31 matches. However, upon further investigation, we can see that some of the matches are false positives. More specifically, we can see that some of the points provided are incorrect, as it seems the noise has been mistaken for keypoints. However, in both cases, SIFT has returned zero matches, and caused enough doubt in the ASIFT responses, thus discouraging potential attackers. In Figure 8, we can see that ASIFT was fooled by a similarly shaped image. In this case the fish and the hat have a similar shape, and it was enough to return matches, even though clearly the two images are quite different. It

is worth noting that in this example, as we scaled the size of the hat image, the ASIFT results dropped to zero. Note that the ASIFT and SIFT engines are sensitive to slight changes in any images (noised or otherwise), and thus generalizing the results to all images will require more study.

5.3 Limitations

As with all noise generation functions, there exists the possibility that their alterations to the image can be significantly decreased with smoothing functions/image filters or reversed entirely by an appropriate function. Sufficiently advanced attackers with image processing experience may be able to reverse some of the distortion effects that come as a result of the noise generation to the degree that the image becomes vulnerable to RIS or CV attacks again. We attempt to minimize this weakness by using randomness in the function when applying the algorithms to the images, as well as using image specific properties to provide alterations within the image. We believe the method has enough merit to be explored further and that the CAPTCHA security community will provide the appropriate level of vetting of our methodology in due time.

Secondly, there exist images that cannot be satisfactorily "noised" – more specifically, the image will either fail to be recognizable by a human due to the excessively high level of noise added to the image to provide the security guarantee, or it will be recognizable to a human but fail to meet the security guarantee because the noise level is too low. This tends to occur when the image does not have colors (e.g. it is mostly composed of black and white.) We plan to explore this behavior in the future and propose solutions for it.

6 Usability Study

To test the usability of the noise method on human users, we designed 4 different styles of image CAPTCHA with varying degrees of difficulty. Style 1 displays an image and asks the user to describe it by entering a description (freeform response). Style 2 displays an image and provides a dropdown box with 4 responses, with 1 of the 4 choices being the correct answer. Style 3 displays a dropdown box with 5 responses, 4 choices and not here. Style 4 asks the user to select the image from 3 images that best represents X, where X is an image category. The order of difficulty is (1,3,2,4) from most difficult to least difficult. All example of each is depicted in Figure 9. For this experiment, 100 noised images were generated in total, 10 images gathered from a web search in each of 10 different categories. The 10 image categories were chosen to be "concrete", to lower ambiguity for the user (airplane(1), bird(2), car(3), cat(4), doll(5), fish(6), flower(7), monkey(8), robot(9), train(10)). All 4 styles have the option to click a link to serve up a new CAPTCHA if the user cannot understand/decipher/solve the one they have been given. This is tallied as "no response" by our database. The study contained approximately 60 undergraduate students who provided anonymous responses to a random series of CAPTCHAs served in the various styles of previously described. The results are shown in Table 3. Not surprisingly, style 1 shows the highest percentage of no response (N), which means that

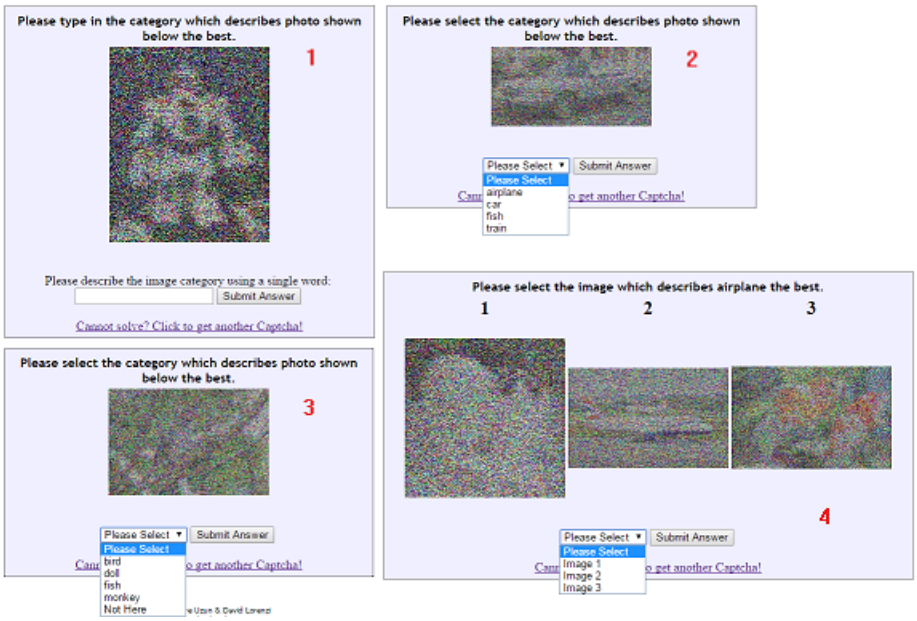

Fig. 9: CAPTCHA Styles

Table 3: Results for CAPTCHA Style Responses

CAPTCHA Style	% Right	% Wrong	% No Response
1	45%	20%	35%
2	64%	11%	24%
3	67%	21%	13%
4	78%	12%	10%

the user was unable to decipher the image or felt unable to answer the question. This is followed by style 2 at 24% and 3 and 4 at 13% and 10% respectively. It is important to understand that styles 2(20%), 3(25%), and 4(33%) provide clues for a chance to guess correctly to the user. As such, one can expect a higher level of correct responses as guessing plays a factor in the results. With the case of 3 and 4, the correct response is provided within the framework of the CAPTCHA. From the data we gathered from users, there is an initial period where the users familiarize themselves with the different challenge styles and give more incorrect responses. As they continue to answer questions and become acquainted with various styles, their accuracy and correctness increase significantly. While the number of images used for testing (100) is not very large, the results amply demonstrate that this is a valid method that can be used by humans to a successful degree.

7 Conclusions and Future Work

In this paper we have shown how noise addition can serve as an effective method to solve the scalability problem of image CAPTCHAs and effectively foil Re-

verse Image Search and Computer Vision attacks. In the future, we plan to test various additional methods of image alterations. We also plan to develop and test multimodal CAPTCHAs, that is, CAPTCHAs that utilize one or more test methods (text based, image based, audio based, etc.) using a combination of protection methods and usability enhancements to provide a comfortable user experience with the maximum level of security possible given those criteria.

References

1. J. Blocki, M. Blum, and A. Datta. Gotcha password hackers! In *AISec '13*, pages 25–34, 2013.
2. M. Chew and J. Tygar. Image recognition captchas. In K. Zhang and Y. Zheng, editors, *Information Security*, volume 3225 of *LNCS*, pages 268–279. 2004.
3. R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.*, 40(2):5:1–5:60, May 2008.
4. R. Datta, J. Li, and J. Z. Wang. Imagination: A robust image-based captcha generation system. In *MULTIMEDIA '05*, pages 331–334, 2005.
5. A. S. El Ahmad, J. Yan, and L. Marshall. The robustness of a new captcha. In *EUROSEC '10*, pages 36–41, 2010.
6. C. Fritsch, M. Netter, A. Reisser, and G. Pernul. Attacking image recognition captchas. In S. Katsikas, J. Lopez, and M. Soriano, editors, *Trust, Privacy and Security in Digital Business*, volume 6264 of *Lecture Notes in Computer Science*, pages 13–25, 2010.
7. R. Gossweiler, M. Kamvar, and S. Baluja. What's up captcha?: A captcha based on image orientation. In *WWW '09*, pages 841–850, 2009.
8. C. J. Hernandez-Castro, A. Ribagorda, and Y. Saez. Side-channel attack on the humanauth captcha. In *SECRYPT '10*, pages 1–7, 2010.
9. M.-F. Jian, H.-K. Chu, R.-R. Lee, C.-L. Ku, Y.-S. Wang, and C.-Y. Yao. Emerging images synthesis from photographs. In *ACM SIGGRAPH '13*, pages 97:1–97:1, 2013
10. D. Lorenzi, J. Vaidya, S. Sural, and V. Atluri. Web services based attacks against image captchas. In *ICISS '13*, pages 214–229, 2013.
11. D. Lorenzi, J. Vaidya, E. Uzun, S. Sural, and V. Atluri. Attacking image based captchas using image recognition techniques. In *ICISS '12*, pages 327–342, 2012.
12. D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV '99*, pages 1150–1157, 1999.
13. P. Matthews, A. Mantel, and C. C. Zou. Scene tagging: Image-based captcha using image composition and object relationships. In *ASIACCS '10*, pages 345–350, 2010.
14. M. Mehrnejad, A. Bafghi, A. Harati, and E. Toreini. Multiple seimcha: Multiple semantic image captcha. In *ICITST '11*, pages 196–201, 2011.
15. N. J. Mitra, H.-K. Chu, T.-Y. Lee, L. Wolf, H. Yeshurun, and D. Cohen-Or. Emerging images. In *ACM SIGGRAPH Asia '09*, pages 163:1–163:8, 2009.
16. J.-M. Morel and G. Yu. Asift: A new framework for fully affine invariant image comparison. *SIAM J. Img. Sci.*, 2(2):438–469, Apr. 2009.
17. L. von Ahn, M. Blum, and J. Langford. Telling humans and computers apart automatically. *Commun. ACM*, 47(2):56–60, Feb. 2004.
18. J. Yan and A. S. El Ahmad. Usability of captchas or usability issues in captcha design. In *SOUPS '08*, pages 44–52, 2008.
19. B. B. Zhu, J. Yan, Q. Li, C. Yang, J. Liu, N. Xu, M. Yi, and K. Cai. Attacks and design of image recognition captchas. In *CCS '10*, pages 187–200, 2010.