SenCAPTCHA: A Mobile-First CAPTCHA Using Orientation Sensors

YUNHE FENG, The University of Tennessee, Knoxville
QING CAO, The University of Tennessee, Knoxville
HAIRONG QI, The University of Tennessee, Knoxville
SCOTT RUOTI, The University of Tennessee, Knoxville

CAPTCHAs are used to distinguish between human- and computer-generated (i.e., bot) online traffic. As the amount of online traffic generated by mobile devices increases, it is necessary to design CAPTCHAs that work well on these devices. In this paper, we present SenCAPTCHA, a mobile-first CAPTCHA that leverages the device's orientation sensors to allow for easy completion of the CAPTCHA on devices with small screen sizes (e.g., smartphones, smartwatches). SenCAPTCHA works by showing users an image of an animal and asking them to tilt their device to guide a red ball into the center of that animal's eye. In this paper, we describe the design of SenCAPTCHA and demonstrate that it is resilient to various machine-learning based attacks. We also report on two usability studies of SenCAPTCHA involving a total of 472 mobile device users; our results show that SenCAPTCHA is viewed as an "enjoyable" CAPTCHA and that it is preferred by half of the participants to other existing CAPTCHA systems.

Additional Key Words and Phrases: CAPTCHA, mobile, orientation, user study

1 INTRODUCTION

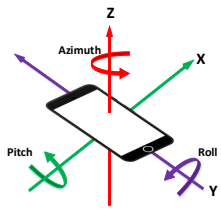
Bots (autonomous Internet programs and scripts) are an important piece of the Internet—for example, Googlebot crawls the Internet to gather data necessary to power Google' search engine. However, over 55% of bot traffic is malicious, with 94.2% of websites experiencing at least one bot attack every 90-days [25]. These malicious bots are responsible for a range of undesirable actions: comment spamming, malicious website registrations, click frauds, dictionary attacks, etc. As such, differentiating between human- and bot-initiated actions is a critical task on today's Internet.

To fight against malicious bots, von Ahn et al. [34] proposed the Completely Automated Public Turing Test To Tell Computers and Humans Apart (CAPTCHA) system in 2000. AltaVista [32] implemented one of the first practical CAPTCHA schemes and used it to block fraudulent account registrations. Today, there are many different CAPTCHA systems, where individual systems can be categorized into text-based, image-based, audio-based, video-based and puzzle-based CAPTCHAs (see Section 2 section for details). While recent updates to Google's ReCAPTCHA system (i.e., NoCAPTCHA) can reduce the frequency with which users need to solve a CAPTCHA, they are far from removing the need for CAPTCHAs altogether. As such, research is still needed to increase the effectiveness and usability of CAPTCHA systems.

Most existing CAPTCHAs were designed with desktop users in mind. This is problematic for a couple of reasons. First, many CAPTCHAs are not designed to work well when screen size or resolution is limited. While mobile phone screen sizes and resolutions continue to increase in size, this is not true of wearable technology (e.g., smart watches). For example, the 44mm Apple Watch has a screen size of just 9.77 cm^2 and a resolution of just 368 by 448 pixels. Second, CAPTCHAs designed for desktops fail to take advantage of the sensors available on modern mobile devices. This is unfortunate, as leveraging the phones sensors could help compensate for the limited screen size and resolution.

Authors' addresses: Yunhe Feng, Anonymous, The University of Tennessee, Knoxville; Qing Cao, Anonymous, The University of Tennessee, Knoxville; Hairong Qi, Anonymous, The University of Tennessee, Knoxville; Scott Ruoti, Anonymous, The University of Tennessee, Knoxville.

(a) SenCAPTCHA



(b) Orientation axes

Fig. 1. The red moving ball is controlled by built-in orientation sensors (X and Y axes) on mobile devices. Users tilt their devices to move the ball to target positions to pass the SenCAPTCHA test.

In this paper, we present SenCAPTCHA, a sensor-based CAPTCHA scheme that addresses both limitations. SenCAPTCHA works by displaying a picture of an animal and asking users to tilt their phones to guide a red ball into the center of that animal's eye (see Figure 1). Using the trajectory of the ball, SenCAPTCHA can distinguish between human- and bot-generated solutions. SenCAPTCHA requires very little space to display and complete the CAPTCHA—just enough to display an image—allowing it to work well on mobile and wearable devices with small screens (e.g., smart watches). Additionally, using the orientation sensor avoids the need for users to type text or select images, tasks which might be difficult on smaller screens (e.g., awkward keyboards, fat finger problem).

The contributions of this work can be summarized as follows:

(1) The design of SenCAPTCHA, a mobile-first CAPTCHA that uses a mobile device's orientation sensor to compensate for limited screen space. It also helps users avoid input modalities (i.e., typing, selecting images) that might be more difficult on devices with small screens (e.g., smartwatches).
(2) We demonstrate that by rotating and tiling the animal image shown to user by SenCAPTCHA, we can prevent machine learning algorithms from locating the animal's eye (i.e., identifying the solution to the puzzle). By examining the trajectory used to solve SenCAPTCHA, we can also differentiate between bots which are searching for the animal's eye and users who know where it is, helping prevent bot-based attacks against SenCAPTCHA.
(3) We conduct a usability study of SenCAPTCHA with 270 users. The results of this study identify how various features related to SenCAPTCHA—how the image is modified, starting point for the red ball, and how close the ball has to get to the eye—affect completion time and perceived usability.
(4) We conduct a usability study with 202 users comparing SenCAPTCHA to four other common CAPTCHA schemes, demonstrating that SenCAPTCHA outperforms these systems in regard to perceived usability, time to complete, and selection as users' favorite system.
(5) We provide an open source implementation of SenCAPTCHA that is available to download at [removed for blind peer review]. This website also contains research artifacts, study instruments, and results from our two user studies and our analysis of SenCAPTCHA's resistance to bots.

2 RELATED WORK

Many different categories of CAPTCHA schemes have been proposed in the literature. In this section we discuss several of these approaches.

Google's NoCAPTCHA: Recently, Google updated their CAPTCHA systems to try and avoid user interaction. This new system—reCAPTCHA v2/v3 (i.e., NoCAPTCHA) [23]—attempts to

identify bots by examining information regarding the user's network connection, browser settings, cookies, and previous browsing behavior. NoCAPTCHA can also track the user's mouse movement to serve as a biometric identifier for human users. In many cases, NoCAPTCHA is able to completely avoid user interaction or only ask that the user click a checkbox that says "I'm not a robot"

Still, there are many instances in which NoCAPTCHA is unable to automatically determine whether a user is human. This can be because of technical limitations (e.g., when cookies are deleted, an incognito web browser session is used, or JavaScript is disabled [45]) or because the algorithm behind NoCAPTCHA is unable to confirm whether the user is human. In these cases, NoCAPTCHA falls back to showing users a traditional CAPTCHA. As such, **NoCAPTCHA does not remove the need for other CAPTCHAs**, only reduces how often they must be used.

Text-Based CAPTCHAs: Text distortion-based CAPTCHAs take advantage of the inherent difficulty of Optical Character Recognition (OCR). They assume that current computer programs are difficult to read distorted texts, but humans can do it easily. The first practical distorted text-based CAPTCHA was invented by Altavista [32] in 1997 before the notation of CAPTCHA was proposed. Later, von Ahn et al. [52] proposed the reCAPTCHA, which deciphers and transcribes scanned text from old books with a word accuracy over 99% in 2008. To enhance the security of text-based CAPTCHAs, many efforts have aimed to frustrate OCR attacks. Baird et al. [1] proposed the ScatterType CAPTCHA, which resisted the segmentation of characters from a word. Hsieh et al. [22] adopted the scale-invariant feature transform (SIFT) algorithm to make characters indiscernible for bots but distinguishable for humans.

However, in recent years, text-based CAPTCHAs are more likely to be solved automatically, thanks to the progress of deep learning frameworks. In 2013, Goodfellow et al. [19] achieved over 90% accuracy of recognizing multi-digit number from street view imagery using deep convolutional neural networks. In 2014, Bursztein et al. [7] presented a machine learning based generic solving of text-based CAPTCHAs by segmenting and recognizing characters simultaneously. In 2015, Starostenko et al. [46] cracked the text-based CAPTCHAs with a three-color bar-based segmentation and a SVM-based learning classifier. The future of text-based CAPTCHA seems uncertain [44], as companies like Google are gradually phasing out text CAPTCHAs.

Audio-Based CAPTCHAs: To make CAPTCHAs available for individuals with visual impairments, some projects [8, 15, 21, 28, 30] explored and developed audio-based CAPTCHAs, which is based on the difficulties for computers in recognizing distorted and noise-filled spoken languages [10]. However, audio-based CAPTCHA has its innate drawbacks: i) users must be proficient in English vocabularies since most existing audio-based CAPTCHA are in English; ii) some characters have the similar pronunciations, leading to confusions [48].

Video-Based CAPTCHAs: In 2009, Kluever et al. [27] proposed the first video-based CAPTCHA, where users watch a video and select the correct labels associated with the video. The first widely deployed video-based CAPTCHA was released by NuCaptcha [24] in 2010. Different from [27], NuCaptcha provides a video of codewords moving across a dynamic scene. Later, several efforts [6, 53] claimed to have undermined the security of NuCaptcha by adopting moving-image object recognition (MIOR) techniques. Compared to text-based and image-based CAPTCHAs, video-based CAPTCHAs are believed to have a higher level of security [48]. On the other hand, loading large files may adversely affect user experiences, especially for mobile users where bandwidth is limited.

Image-Based CAPTCHAs: Bongo, developed by Mikhail M. Bongard [2], is one of the earliest visual pattern recognition problems used as CAPTCHAs [51]. Chew et al. [9] were among the first to study labeled photograph-based CAPTCHAs. Since then, many image recognition-based works have been proposed. Asirra [14] gave users a challenge to choose cats from 12 photographs of both cats and dogs. IMAGINATION [13] exploited the difficulty for bots to recognize composite

distorted images. Gossweiler et al. [20] proposed CAPTCHAs by requiring users to identify an image's upright orientation. Matthews et al. [35] proposed a scene tagging test, which relies on the ability of users on recognizing the relationships between irregularly shaped objects embedded in a background image. Image-based CAPTCHAs have the following disadvantages. First, some users find it difficult to identify images because of low vision or blurred pictures [48]. Second, they are likely to suffer from image recognition attacks. For example, one year after Asirra [14] was proposed, Philippe Golle [18] trained a machine learning-based classifier and achieved the 82.7% accurate in distinguishing the images of cats and dogs used in Asirra.

Puzzle-Based CAPTCHAs: Existing approaches in this category have mostly considered conventional methods. Gao et al. [16] proposed the jigsaw puzzle-based CAPTCHA by dividing an image into misplaced pieces. Kaur et al. [26] designed a boolean algebra puzzle-based CAPTCHA using circuit diagrams consisting of basic logic gates like OR, AND, NOR, among others. Conti et al. [11] implemented the CAPTCHaStar system based on interactive shape discoveries. All of these CAPTCHAs share in common that they mostly rely on conventional puzzle designs and are not easy for humans because of additional efforts required to pass these puzzles compared to image or text-based ones.

3 SENCAPTCHA

In this section, we first introduce the design principles of SenCAPTCHA. We then describe how SenCAPTCHA is resilient to bot-based attacks. Finally, we present the system overview and detailed implementation of SenCAPTCHA.

3.1 Design Principles

SenCAPTCHA is a mobile-first CAPTCHA scheme leveraging the orientation sensors available on mobile devices to provide a CAPTCHA that works even on devices with constrained screen real estate (e.g., smartphones, smartwatches). In SenCAPTCHA, users are shown a picture of an animal and are asked to tilt their phones to guide a red ball into the center of that animal's eye (see Figure 1). The design of SenCAPTCHA was guided by three principles:

- **Low effort.** SenCAPTCHA requires no expert knowledge and is designed to be easily solved by most users. Instead of relying on conventional I/O interface (e.g., keyboard, mouse, touchscreen), SenCAPTCHA utilizes orientation sensors, offering a natural, novel, and efficient user experience for mobile devices.
- **Enjoyable.** We modeled SenCAPTCHA after sensor-based games on mobile devices that ask users to complete various tasks by rotating their phones. By adopting a more game-like approach, we hoped to reduce the burden of completing SenCAPTCHA (similar to other puzzle-based CAPTCHAs).
- **Clean UI.** SenCAPTCHA avoids any blurred visual elements, often found in text-based and video-based CAPTCHAs, increasing its visual appeal and ease of use.

3.2 Resilience to Bots

To prevent bots from solving SenCAPTCHA, we take advantage of the fact that detecting animal facial keypoints is a hard problem, as most existing literature on facial keypoint detection focuses on humans [12, 47]. One challenge for animal facial keypoint detection is that, due to the high annotation and collecting cost for training, the amount of annotated data of animals is too limited to directly deploy Convolutional Neural Networks (CNNs), which have shown impressive performance on human faces. For example, the AFLW [29] consists of 25,993 human faces, while the horse and sheep datasets from previous work [39, 54] only have 3,717 and 600 images, respectively. The

lack of data has caused CNNs to be much less effective for animal faces compared to human faces. Further, before deploying CNNs on the target image to detect facial keypoints, the animal species must be identified, which also requires a very large labeled dataset containing all types of animals involved, even fictional animals. Another challenge is that animal images are much more diverse in their surroundings. For example, animal camouflage images are very difficult to be processed successfully based on normal training steps. Given these challenges, although a few state-of-the-art studies have attempted to localize the facial keypoints of animals by extracting triplet interpolated features [54] or transferring knowledge gained from human faces [39], they admit their localization methods may not perform well in many situations.

To further improve SenCAPTCHA's resilience to attacks, we randomly rotate, zoom and translate, or tile the animal picture as shown in Figure 4. This makes keypoint detection harder by changing the shape of the animal, modifying facial features, or partially obscuring the face preventing the creation of an accurate bounding box [39, 54], even though labeled datasets are available for training models. For example, we observed the average failure rate across keypoints of a deep learning model proposed in [39] increased from 8.36% to over 90% when image mutations were applied on a large dataset containing 3717 horse images. We measured the security and usability impact of these changes, and this is discussed later in the paper (see Section 4 and Section 5 respectively).

3.3 System Overview

SenCAPTCHA works in a client-server model and is broken up into two components. A server-side component provides the background image and determines when the CAPTCHA is solved. A `HTML5`/JavaScript-based client-side component displays the CAPTCHA to the user and records the ball's movement to send to the server for verification. Importantly in this separation of duties, the client-side portion never learns the location of the eye, as the client-side component runs inside the attacker's environment.

As shown in Figure 2, the workflow of SenCAPTCHA can be broken up into three primary phases: (1) SenCAPTCHA is initialized; (2) the users tilts their devices to move the ball into the animal's eye; and (3) the ball's trajectory is analyzed to distinguish between authentication human users and bots. Most operations are performed on the server, with only the UI-rendering and sensor-reading handled by the client.

3.3.1 Initialization. During the initialization phase, the server needs to create a puzzle image, determine how close the ball must get to the target, selects a starting position for the ball, and sends this information the client.

Creating Puzzle Image. To create SenCAPTCHA's puzzle image, the server first selects an image from its image corpus. This corpus contains a set of images along with their respective keypoints. It then mutates the selected image using one of the following three techniques:

- **Rotate.** The image is rotated by a random angle. The image is then scaled such that no white-space is shown to the user. If the eye falls outside the area shown to the user or is too close to the edge, then another randomly selected rotation angle is chosen.
- **Zoom and translate.** A random horizontal scaling factors and a separate random vertical scaling factor are used. After scaling the image, it is translated to ensure that the eye is within the area shown to the user and is not too close to the edge of the image.
- **Tile.** The image is broken into several tiles (currently fixed at 9 [3x3], though could also be randomly selected), then those tiles are randomly arranged and displayed to the user. As with other methods, if the eye is too close to the edge of the image, it will be moved to another random location.

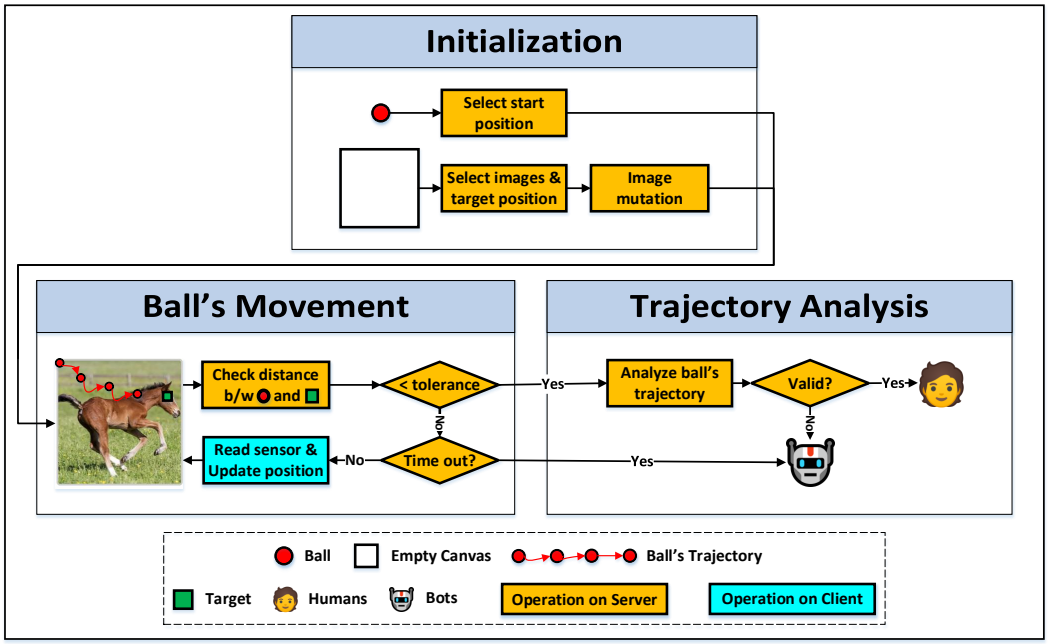


Fig. 2. SenCAPTCHA's workflow

As part of this mutation, the server will also calculate the new location of the image's keypoint. Note that each mutation can take a single image from the image corpus and generate a large number of unique puzzles. For example, the tile mutation takes a single image, breaks it into nine tiles, and then randomly orders those tiles, producing one of 9!=362,880 different puzzles. This process ensures that even with a small corpus of images, we can generate a massive number of puzzles for SenCAPTCHA's use.

Calculate Target Distance. Next, the server calculates how close the center of the ball must be to the center of the puzzle's keypoint for the puzzle to be solved. This distance (d) is measured in and is calculated based on a server-side *tolerance* value that indicates how close the two centers must be as a percentage of the overall image size. Note that there is an inverse relationship between tolerance and the difficulty of solving the puzzle. The formula for calculating d from tolerance is given in Equation 1.

$$d = tolerance * \frac{width + height}{2} \tag{1}$$

The radius of the ball is made equal to distance d to ensure that if the user sees the ball touch the keypoint, the puzzle will also be solved. We do impose a minimum radius of 5 pixels for the ball to ensure that it is visible on small form factors with limited resolution.

Select Starting Position. The ball the user needs to move starts at a random position. This position is picked by selecting one option from each of the following sets and combining them: {top, middle, bottom} and {left, center, right}.

Send Information to the Client. The server-side component sends the client-side component the following data: (1) the mutated image, (2) the ball's starting location, and (3) the ball's radius. The

client-side code will then handle rendering the image to a `HTML Canvas` element. The ball is also rendered to this element at the indicated starting point with the given radius. The ball itself is colored red, with a solid black outline.

3.3.2 Moving the Ball. To move the ball, users tilt their devices.[1] To track the device's tilt, we use the azimuth-pitch-roll convention to represent the mobile device's orientation angles, as shown in Figure 1 (b). Among the three angles, only the pitch angle (degrees of the rotation about the X axis) and the roll angle (degrees of the rotation about the Y axis) are used to compute the position of the ball, since the azimuth angle denotes a compass-like direction value and contributes nothing to the ball's movement. The pitch angle β and roll angle γ change when tilting mobile devices upwards/downwards and rotating mobile devices around their left/right edges respectively. β reports values between $-180°$and $180°$, while γ reports values between -90 °and $90°$.

To access the current pitch angle β and roll angle γ through mobile browsers, the client-side component listens to the `deviceOrientation` event, which report fresh orientation sensor readings whenever the device is rotated (i.e., tilted). Each time this event is fired, we calculate the δ between the previous rotation and the current rotation and use this to move the ball. If the rotation crosses an axis (e.g., goes from $79°$to $-79°$), we catch these large changes and transition them to the small change they actually represent (e.g., $2°$). The actual distance the ball moves is controlled by a ball speed parameter set by the server (currently, this is $\frac{1}{30}$ of the image for each degree of rotation). The movement of the ball itself is limited to within the image.

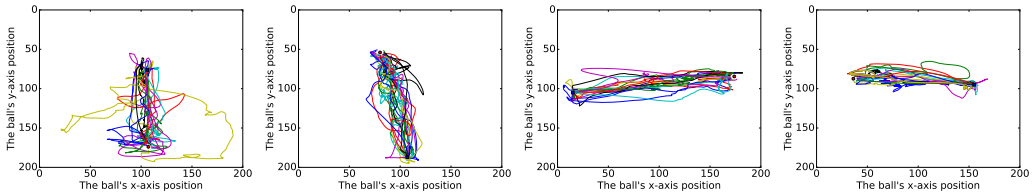
Each time the ball moves by at least a single pixel (modern devices are sensitive to record many sub-pixel moves), the new location is sent the server which checks whether the CAPTCHA has been completed—i.e., whether the distance between the center of the ball and the center of the keypoint are less than d pixels apart. If the two are within d pixels, the puzzle is solved, and the client is notified of the puzzle's completion. Note that the user does not need to hover over the keypoint or click a button to finish the puzzle, just move within the appropriate distance of the keypoint. In Section 9.3.4 we discuss other potential options for puzzle completion.

If the user fails to complete the CAPTCHA within a time limit (e.g., 1 minute), they are considered to have failed the CAPTCHA.

3.3.3 Trajectory Analysis. After the ball has been moved into the center of the animal's eye, the server will check the history of ball locations (i.e., trajectory) to determine if the CAPTCHA was completed by a human or by a bot. When completing SenCAPTCHA, we expect that humans will move the ball directly towards the animal's eye. In contrast, the design of SenCAPTCHA makes it difficult for bots to determine the eye's location with high confidence, therefore they will need to employ heuristic search strategies to find the eye from a set of possible locations. As such, it becomes possible to distinguish between human- and bot-based completion by comparing how closely the ball's recorded trajectory matches the optimal trajectory (a straight line from the starting location to the animal's eye). This comparison is done using a Dynamic Time Warping (DTW) algorithm [38, 42], with the attempt being considered human-generated if the calculated warping distance is below a given threshold.[2]

To confirm that human trajectories resemble the ideal trajectory, we completed hundreds of SenCAPTCHA puzzles within our research group and recorded those trajectories. As shown in Figure 3, while these trajectories do show deviation from the optimal path, they generally stay close to that path. This is different from bot-based approaches which will have to use various search

[1]SenCAPTCHA also supports moving the ball using touch or the mouse, but this is disabled by default.
[2]Thresholds for our DTW-based classification are based on both the behaviors of bots (see Section 4) and empirical data gather from our first user study (see Section 5).

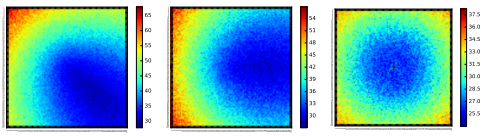


(a) Top to bottom (b) Bottom to Top (c) Left to right (d) Right to left

Fig. 3. Example human-generated trajectories

(a) Rotate (b) Zoom&tran. (c) Tile

Fig. 4. Image mutations to improve attack resilience



(a) Corner (b) Mid edge (c) Center

Fig. 5. Values of $\mu - 2.5\sigma$ for random guessing attacks

heuristics to find the eye. These results were further confirmed by the results from our user study (see Section 5).

To further differentiate between humans and bots, it may be possible to train a machine learning algorithm on the user's trajectory and see if it looks human like and resembles past attempts from the user (similar to what NoCAPTCHA does with user mouse movement). Such approaches could help detect bots even if they get lucky and correctly detect where the eye is.

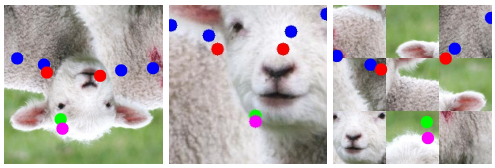
Still, we note that it is likely that an advance adversary could trick the trajectory analysis if it knew the location of the image's keypoint. As such, it is important to remember that SenCAPTCHA's security is primarily based on the difficulty of finding the keypoint.

4 SECURITY ANALYSIS

In this section, we analyze possible attack approaches on SenCAPTCHA. We categorize possible attacks into brute-force attacks and machine learning based attacks. Specifically, brute-force attacks check every possible position for the eye until the correct position is found. Machine learning based attacks take advantage of visual recognition techniques to identify the eye's position automatically.

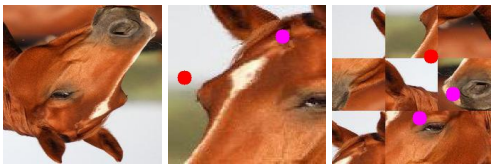
4.1 Brute-force Attacks

The most intuitive brute-force attack is random guessing, which checks arbitrary locations one by one until the target point is reached. For each pair of the starting and ending point, we can estimate its distribution of the random guessing DTW distance through sampling. Recall that SenCAPTCHA allows the ball launching from one position of the four corners, the four middle point at the edge, and the center of the canvas. Considering the symmetries of the canvas, we only need to study three representative starting locations: corner, middle point at edge, and center. Suppose the canvas size is 100 by 100, and the ball starts from the corner. Then we randomly select (x, y) as the target point, where both x and y are integers. Thus, we have 9,999 target points since the start location cannot be the target point at the same time. For each (x, y), we run the random guessing method for 1000 times, generating 1000 trajectories randomly.



(a) Rotation (b) Zoom (c) Tile

Fig. 6. TIF attacks. ● (eye), ● (nose), ● (ear), ● (mouth)



(a) Rotation (b) Zoom (c) Tile

Fig. 7. IKT attacks. ● (eye), ● (nose), ● (mouth)

Then we calculate all DTW distances between each simulated trajectory and the optimal path, i.e., the straight line connecting the start location and the target location (x, y). Next, we use the normal distribution $f(d|\mu, \sigma^2)$ to fit the probability density of DTW distances, where d represents the DTW distance, μ denotes the mean of DTW distances, and σ is the standard deviation. The cumulative distribution function of $f(d|\mu, \sigma^2)$ is denoted by $F(d)$. If we take the $\mu - 2.5\sigma$ as the threshold to distinguish humans from bots (i.e., if the DTW distance is larger than the threshold, it is classified as bots), the successful attack rate will be $F(\mu - 2.5\sigma)$, which is less than 0.63%. Thus, for all the 9,999 target points with a starting location from the corner, we plot their $\mu - 2.5\sigma$ in Figure 5 (a). Similarly, we simulate and calculate the $\mu - 2.5\sigma$ for starting location from the middle point of edges and the center, as shown in Figure 5 (b) and (c) respectively. Base the $\mu - 2.5\sigma$ values in Figure 5, a DTW threshold of 25 will be safe to distinguish bots from humans. We also checked 2385 trajectories by participants in Section 5 and found 93% responses were classified correctly.

4.2 Machine Learning based Attacks

In this subsection, we deployed three machine learning based attacks, namely Scale-Invariant Feature Transform (SIFT) model [33], Triplet Interpolated Feature (TIF) model [54], and Interspecies Knowledge Transfer (IKT) model [39], to localize the animal facial keypoints on mutated images which are used in SenCAPTCHA.

	LOE	LIE	LE	RE	RIE	ROE	N	M	ALL
Zoom	99.2%	94.5%	91.4%	90.4%	94.2%	97.0%	93.4%	94.4%	92.7%
Rotate	99.3%	98.9%	99.0%	98.7%	98.8%	99.2%	98.6%	98.8%	98.9%
Tile	96.7%	95.1%	95.8%	95.3%	95.6%	96.8%	94.7%	95.1%	95.6%

Table 1. SIFT's detection failure rate on the sheep dataset

	LE	RE	N	LM	RM	ALL
Zoom	96.4%	97.0%	97.2%	97.4%	97.1%	96.9%
Rotate	96.2%	96.4%	96.2%	96.5%	96.7%	96.4%
Tile	96.9%	97.0%	96.7%	96.9%	96.6%	96.8%

Table 2. SIFT attacks on the horse dataset

Datasets and Evaluation Metric: In TIF paper [54] and IKT paper [39], the authors published 50 labeled sheep images and 3717 labeled horse images respectively. Each sheep image contained a face bounding box and eight labeled keypoints, i.e., Left Outer Ear (LOE), Left Inner Ear (LIE), Right Outer Ear (ROE), Right Inner Ear (RIE), Left Eye (LE), Right Eye (RE), Nose (N), and Mouth (M). Each horse image was labeled five keypoints, i.e., Left Eye (LE), Right Eye (RE), Nose (N), Left Mouth Edge (LM), and Right Mouth Edge (RE). We observed the released horse images had been cropped accordingly, and used the rectangle composed by four image edges as the bounding box. When measuring the detection failure rate, we followed the same metric used in [39, 54]: If the Euclidean distance between the detected keypoint and its corresponding ground-truth keypoint was more than 10% of the face bounding box size, it was regarded as a failure. We evaluated SIFT on both datasets, TIF on the sheep dataset, and IKT on the horse dataset.

SIFT Attacks: As SIFT is robust to uniform scaling, orientation, and illumination changes [33], it has been widely used for object recognition. We applied SIFT on both the sheep and horse datasets

to detect facial keypoints. For each dataset, we calculated the average detection failure rate for three types of mutated (zoomed, rotated, and tile) images, Considering the relatively small size of the sheep dataset, we generated ten zoomed and tile images randomly for each sheep image to expand the dataset. Note that we did not perform data augmentation on rotated images because SIFT is invariant to orientation. The average detection failure rates of SIFT on the sheep and horse datasets are illustrated in Table 1 and Table 2 respectively. All failure rates for any facial keypoints on the two datasets are above 90%. The lowest average failure rates for all eight sheep keypoints and five horse keypoints are higher than 92.7% and 96.4% respectively (see the last columns in Table 1 and Table 2.) These high failure rates indicate SenCAPTCHA is able to resist SIFT attacks with a high probability.

TIF Attacks: The TIF method extracts triplet interpolation features and calculates a shape-indexed feature using three anchor landmarks [54]. TIF is sensitive to the given face bounding box when identifying keypoints and it will not work effectively if we intentionally crop partial animal faces due to the incorrectly computed bounding box, as shown in Figure 6 (b). When an image is rotated or tiled randomly, TIF fails to detect facial keypoints as well. Examples are shown in Figure 6 (a) and (c). Similar with data augmentation adopted in SIFT attack analysis, we generated ten randomly zoomed, rotated, and tile images for each sheep image to increase the diversity of data. Table 3 illustrates keypoint detection failure rates (% of predicted keypoints whose Euclidean distance to the corresponding ground-truth keypoint is more than 10% of the face bounding box size) regarding eight different facial keypoints on three types of image mutations. The average failure rates on both zoomed and rotated images are above 95%. Although the average failure rate of tiles is below 90%, the tolerance in SenCAPTCHA (ranging from 2% to 3% of the bounding box size) is much smaller than the failure detection threshold, i.e., 10% of the bounding box size. It implies a higher failure rate when applying TIF to solve SenCAPTCHA.

	LOE	LIE	LE	RE	RIE	ROE	N	M	ALL
Zoom	99.0%	94.0%	93.2%	93.4%	95.6%	99.0%	93.0%	95.8%	95.4%
Rotate	100%	100%	100%	100%	100%	100%	99.8%	100%	100%
Tile	84.8%	83.4%	85.6%	87.4%	86.0%	87.4%	90.8%	89.6%	86.9%

Table 3. TIF's detection failure rate on the sheep dataset

	LE	RE	N	LM	RM	ALL
Zoom	90.6%	93.9%	88.4%	90.0%	88.5%	90.8%
Rotate	97.8%	97.9%	99.5%	98.6%	98.0%	98.5%
Tile	97.2%	97.4%	96.6%	95.8%	96.0%	96.6%

Table 4. IKT attacks on the horse dataset

IKT Attacks: As a deep transfer learning model, IKT achieves state-of-the-art animal facial keypoint detection performance by warping the animal face shape to make them more human-like, and then adapting the pre-trained human keypoint detector [39]. However, due to the assumption of the known face bounding boxes, IKT cannot deal with the animal images with partial faces, as shown in Figure 7 (b). In addition, Figure 7 (a) and (c) demonstrate that IKT's performance is affected dramatically by rotations and misplaced keypoints. For each image mutation, we first applied it on the horse dataset, and then trained a new IKT model from scratch. We used the same testing images that were used in the IKT paper [39] for evaluation. Note that the testing images were not unseen by our trained IKT models, although it might improve the detection accuracy. Table 4 summarizes the failure rates regarding five different facial keypoints for three types of image mutations. It is interesting to note that the average failure rate on zoomed images is 90.8%, which is much lower than that of rotated images (98.5%) and tiles (96.6%). This low failure rate on zoomed images implies one limitation of IKT: to find an angle of interest to build a mapping with human faces, IKT requires at least three keypoints existing within the bounding box, which indicates the impossibility of training IKT on images with less than three keypoints.

5 FIRST USABILITY STUDY—METHODOLOGY

We conducted an IRB-approved, Amazon Mechanical Turk-administered usability study of SenCAPTCHA. In this study, we measured overall impressions regarding SenCAPTCHA as well as how various parameters affect SenCAPTCHA's perceived usability and time to complete: four image manipulations, three puzzle tolerances, and nine starting positions. The study ran over the course of five hours on October 9th, 2019. In total, 308 participants completed our usability study, with data from 270 of them used in our analysis of SenCAPTCHA.

5.1 Study Setup

Participants took 1–15 minutes to complete the study, with an average time of 3 minutes. Participants were paid $1.25 USD for their participation. Before beginning the study, participants read an IRB-approved consent statement that informed them that they would need to use a mobile phone or tablet to complete the study. The survey itself also checked whether participants were using a mobile phone or tablet and would not let them complete the study if they were not.

After consenting to be a part of the study, participants were then asked a set of simple demographic questions (gender and age). They were also asked how often they check their mobile phones. The survey software also automatically gathered information about the user's mobile device (e.g., OS version, browser version, screen resolution).

After providing this information, users were told that they would complete a series of where they "will be shown a picture of an animal and will tilt your phone to move a red ball into the middle of the animal's eye." On the page with these instructions, they were given a chance to practice moving a ball around an image. When they were ready to continue, they could click the next button.

At this point, participants were randomly assigned two different treatments:

(1) Image mutation: no mutation, rotation, zoom and translate, or tile.
(2) Puzzle tolerance: 0.02, 0.025, 0.03

Participants then completed nine SenCAPTCHA puzzles. The base image used in each puzzle was randomized, but we ensured that participants were never shown the same image more than once in the study. Each of the nine images used a different starting position for the ball, with the order of these starting images randomized. For the image mutations, all parameters were randomized each time (see Section 3).

Participants were given one minute to solve each CAPTCHA. If they did not solve the CAPTCHA within that minute, they were shown a message (using JavaScript's `alert` function) telling them that the puzzle had timed out. When the click Ok to this message, the next puzzle was automatically started. The time it took participants to complete the CAPTCHA was recorded along with the trajectory used to complete the CAPTCHAs was recorded for each puzzle, including whether the user timed out.

After completing the nine puzzles, participants were told that the puzzle they just completed was intended to be used as a CAPTCHA. This was done so that users could contextualize their answers in the remainder of the study, as feedback for SenCAPTCHA as a fun puzzle is likely to be very different than feedback for SenCAPTCHA as a CAPTCHA. After being informed of SenCAPTCHA's intended use, participants were asked to complete the ten questions from the System Usability Scale (SUS) [4, 5]. We chose to use the SUS scale for several reasons. First, SUS has been shown to be a good indicator of perceived usability [49], is consistent across studies [41], as has reported norms that we can compare our results against [43].

Finally, participants were asked if they had any other feedback regarding SenCAPTCHA. After completing the survey, participants received a code that they would input into the Amazon MTurk site to get paid.

Category	Demographic	Count	%
Gender	Male	170	63.0%
	Female	100	37.0%
	Prefer not to say	0	0.0%
Age	18–24	55	20.4%
	25–34	145	53.7%
	35-44	50	18.5%
	45–54	12	4.4%
	55–64	8	3.0%
	Prefer not to say	0	0.0%
	65 years or older	0	0.0%
Mobile Phone Usage	Constantly	31	11.5%
	6–15 times an hour	32	11.9%
	3–5 times an hour	71	26.3%
	1–2 times an hour	86	31.9%
	Less than once an hour	47	17.4%
	Prefer not to say	3	1.1%
Country	India	139	51.5%
	USA	122	45.2%
	Other	9	3.3%

Table 5. Demographics for first usability study

Category	Demographic	Count	%
Gender	Male	127	62.9%
	Female	74	36.6%
	Prefer not to say	1	0.5%
Age	18–24	16	7.9%
	25–34	132	65.3%
	35–44	37	18.3%
	45–54	11	5.4%
	55–64	3	1.5%
	Prefer not to say	1	0.5%
	65 years or older	2	1.0%
Mobile Phone Usage	Constantly	28	13.9%
	6–15 times an hour	31	15.3%
	3–5 times an hour	61	30.2%
	1–2 times an hour	57	28.2%
	Less than once an hour	25	12.4%
	Prefer not to say	0	0.0%
Country	India	73	36.1%
	USA	117	57.9%
	Other	12	5.9%

Table 6. Demographics for second usability study

5.2 Data Cleanup

In total, 308 participants completed the survey, but several were unable to get SenCAPTCHA working. Of the participants, 266 completed every puzzle, 2 participants completed all but one puzzles, 2 completed all but two puzzles. The remaining 38 participants failed to complete any puzzles. Based on feedback from these 38 participants, we learned that in many cases SenCAPTCHA would simply not run on their devices. There was a variety of reasons for this: (a) In Safari on iOS 13, by default websites are not allowed to use sensor data, (b) some distributions of Android do not report screen orientation events to the browser if screen rotation has been locked, (c) and other distributions always refuse to provide sensor data to the browser (likely tied to a distribution-specific setting). As we were using this study to evaluated SenCAPTCHA, we remove data from the 38 participants that were unable to ever use SenCAPTCHA, but did include the data from participants where there was only a problem once or twice, leaving us with 270 participants for which we report results. For those 270 participants, we report all available data, though for the four participants that had trouble completing SenCAPTCHA at least once, we only included the completion times for the puzzles where SenCAPTCHA worked for them.

5.3 Qualitative Analysis

To analyze the qualitative feedback provided by users, we used an open coding approach based on the constant-comparative method [17]. Two coders together reviewed every response, assigning one or more codes to describe the sentiment expressed in the response. Codes were only assigned if both reviewers agreed on the assignment. At the end of the coding process, the coders made a second pass over the data to merge related codes and ensure consistent usage across the data set. They also selected representative quotes to be used to help describe each code.

5.4 Demographics

Participants in our study were pulled from Amazon Mechanical Turk. Other than the restriction that participants must complete the study with a smartphone or tablet, there were no other limitations on who could participate in the study. Demographics from our study are given in Table 5.

			Mean	Mdn	SD	Mean	Mdn	SD
Category	Value	Count	SUS			Time (seconds)		
Overall		270	66	61	19	4.7	3.7	3.5
Mutation	None	70	67	63	20	4.6	3.4	3.1
	Rotate	68	64	59	18	4.9	3.5	5.0
	Zoom	70	66	61	20	4.6	3.9	2.5
	Tile	62	64	60	19	4.8	4.0	2.7
Tolerance	0.02	92	65	60	19	5.5	4.1	4.8
	0.025	105	67	63	20	4.1	3.5	2.2
	0.03	73	64	58	18	4.5	3.5	2.7
Age	18–24	55	58	58	19	4.2	3.4	3.7
	25–34	145	63	58	19	5.0	3.8	3.9
	35-44	50	79	74	18	4.2	3.7	1.0
	45–54	12	81	81	15	4.3	4.2	1.5
	55–64	8	87	89	8	5.8	4.7	3.3
Mobile Phone Usage	Constantly	31	71	65	19	3.6	3.3	1.6
	6–15 times an hour	32	73	75	19	3.9	3.5	2.4
	3–5 times an hour	71	61	53	20	4.4	3.5	2.8
	1–2 times an hour	86	66	63	18	5.2	4.0	3.5
	Less than once an hour	47	63	58	19	5.1	4.0	5.0
	Prefer not to say	3	42	40	8	11.5	11.2	2.3
Country	India	139	53	50	11	5.2	4.0	3.9
	USA	122	78	84	18	5.6	3.6	7.8
	Other	9	81	90	17	4.1	3.5	2.2

Table 7. Quantitative results from the first usability study

5.5 Limitations

While we informed participants that SenCAPTCHA was intended to be used as a CAPTCHA, we did not actually have users test it in this capacity. As such, the results might not fully reflect SenCAPTCHA's true usability. While such a measurement was sufficient for our intended purposes, future work could try SenCAPTCHA in-situ to see if reactions change.

6 FIRST USABILITY STUDY—RESULTS

In this section, we report on the System Usability Scale (SUS) scores for SenCAPTCHA, the time taken to complete SenCAPTCHA puzzles, and qualitative feedback. Quantitative results are summarized in Table 7. We use a one-way ANOVA as our omnibus test with Tukey's as our post-hoc pairwise statistical test. Where appropriate, we have used a Bonferroni correction to adjust for family-wise error in our statistical tests.

6.1 System Usability Scale

The differences between mutations ($F(3, 266) = 0.52$, $p = 1.0$) and tolerances ($F(2, 267) = 0.27$, $p = 1.0$) were not statistically significant. SUS scores from India are higher than scores from other countries (omnibus—$F(2, 267) = 96.03$, $p < 0.001$, India vs. USA—$p < 0.001$, India vs. Other—$p < 0.001$, USA vs. Other—$p = 0.87$). We initially hypothesized that this might be due to individuals in India having older devices but found no support for this in our data. There is also statistically significant difference for SUS scores based on age ($F(4, 265) = 10.47$, $p < 0.001$), with users 18–34 giving lower scores than users 35–64 (all pairwise tests within these groups are not statistically significant, all pairwise tests between these groups are). Finally, there is a statistically significant difference for scores based on mobile phone usage ($F(5, 264) = 3.38$, $p = 0.03$), though post-hoc testing finds only a single statistically significant difference between "6–15 times and hour" and "3–5 times an hour" ($p = 0.04$).

Comparing our SUS results to norms reported by Sauro and Lewis [43], we see that the SUS score for SenCAPTCHA falls in line with the mean norm reported Sauro and Lewis. This is encouraging as we would expect CAPTCHAs to be annoying and fall below the mean norm. Additionally, we

note that for older participants and participants not from India, the average SUS scores are much higher and fall between the 90^{th} and 97^{th} percentile for SUS scores, a truly great achievement.

6.2 Time to Complete

The only statistically significant difference in completion time is for mobile phone usage ($F(5, 264) = 4.03$, $p < 0.01$), inasmuch as users who "Prefer not to say" are slower than all other groups (all pairwise tests with "Prefer not to say" are statistically significant, no others are). The differences between mutations ($F(3, 266) = 0.12$, $p = 1.0$), tolerances ($F(2, 267) = 4.302$, $p = 0.07$), ages ($F(4, 265) = 1.12$, $p = 1.0$), and countries ($F(4, 265) = 3.55$, $p = 0.15$) are not statistically significant.

6.3 Qualitative Feedback

Just under two-thirds of participants left some feedback in the optional feedback field, though around half of this feedback was about the survey—i.e., generic comments about it being a good survey. In general, participants had positive perceptions of SenCAPTCHA: 17 (6.3%) participants indicated that it was enjoyable, 11 (4.1%) that it was easy, 8 (3.0%) that it was interesting, and 58 (21.5%) with another generic positive response. 5 (1.9%) participants explicitly stated that it was better than other CAPTCHAs. For example, one participant stated,

> *"This would be SO much better compared to the traditional captchas. I love it, seriously whoever came up with, this great job I really hope it is integrated on sites very soon. I would love to see traditional ones be completely replaced by this. It's much less annoying, and it seems a lot more consistent. Thank you for letting me be a part of this."*

There were very few negative comments. 5 (1.9%) participants had a generic comment about not liking SenCAPTCHA and 8 (3.0%) participants worried that SenCAPTCHA might not work in some situations: for users with accessibility needs—2 (0.7%) participants, on older phones—1 (0.4%) participant, if screen rotation was locked—1 (0.4%) participant, sensor might fail—1 (0.4%) participant, and unspecified worry—3 (1.1%) participants. For example, one participant said,

> *"I found this easy to use but I could see some people, especially older people, having a hard time with it."*

5 (1.9%) participants also mentioned needing to change their iOS settings to get SenCAPTCHA working, which helped us track down and fix that issue in our second user study.

7 SECOND USABILITY STUDY—METHODOLOGY

In our first study, we found that users generally had positive attitudes towards SenCAPTCHA and that they completed the CAPTCHAs very quickly. To understand how SenCAPTCHA's performance compares to other existing CAPTCHA schemes, we conducted an IRB-approved, Amazon Mechanical Turk-administered, within-subject usability study test of five different CAPTCHA systems. The study ran over the course of seven hours on November 12th, 2019. In total, 202 participants completed our second usability study.

The five systems we tested are,

- **SenCAPTCHA.** Based on our security evaluation, we chose to use the rotation and tile image mutations. Based on our first usability study, we set the difficulty parameter to 0.25.
- **Text-based.** We used Securimage [37], an open-source free PHP CAPTCHA script, to generate text-based CAPTCHA challenges. To complete the CAPTCHA, users had to enter the two words displayed. These words were pulled from a dictionary consisting of 1,488 common six-letter English words. They were also slightly distorted with added background noise.

- **Audio-based.** We again used Secureimage to generate audio-based CAPTCHA challenges. To complete the CAPTCHA, users had to enter six random alphanumeric characters that were read aloud. The audio sample had background noise inserted.
- **Image-based.** We implemented an image-based CAPTCHA that displays 16 images to users in a four-by-four gird and asks them to select images matching some noun (e.g., horses, dogs). Each CAPTCHA contains three to six different types of images, with the number of target images ranging from one to five. Images were pulled from the CIFAR-10 dataset and filtered to select the clearest images.
- **Video-based CAPTCHA:** We implemented a video-based CAPTCHA that displayed five capital letters moving across the image from left to right. The letters loop around every 2.7 seconds and rotate independently within a range of 30 degrees.

In each case, we implemented the non-SenCAPTCHA systems to be of comparable difficulty (i.e., same level of distortion or obfuscation) to what we have observed online. In each case, we also erred on the side of making these CAPTCHAs easier to provide a fair comparison to SenCAPTCHA. For consistency, we had all the systems allow users to refresh the CAPTCHA and get a new challenge. We also standardized the user interface for each CAPTCHA to be as similar as possible. If users gave an incorrect answer for the CAPTCHA, they were informed of the mistake and a new CAPTCHA challenge was generated for them to solve.

7.1 Study Setup

Participants took 3–27 minutes to complete the study, with an average time of 9.5 minutes. Participants were paid $1.25 for their participation. Before beginning the study, participants read an IRB-approved consent statement, which informed that that they would need to use a mobile phone or tablet to complete the study. The survey itself also checked whether participants were using a mobile phone or tablet and would not let them complete the study if they were not.

After providing this information, participants completed tasks for each of the five different types of CAPTCHAs (order of the different types was randomized). For each type of CAPTCHA, participants completed three consecutive CAPTCHA instances, with the option to skip a given instance if it took them more than one minute to complete. After completing all three instances, participants provided feedback on their experience using the After-Scenario Questionnaire (ASQ) [31], which consists of three questions answered using a seven-point Likert scale:

Q1: Overall, I am satisfied with the ease of completing this type of CAPTCHA
Q2: Overall, I am satisfied with the amount of time it took to complete this type of CAPTCHA
Q3: Overall, I am satisfied with the support information (online help, messages, documentation) when completing this type of CAPTCHA

After completing all five types of CAPTCHAs, participants were asked to indicate which of the five was their favorite and why it was their favorite. They were also asked to indicate which of the five, if any, they wouldn't want to use in practice and why they wouldn't want to use them. For each of these questions, the ordering of the systems was randomized. Finally, they were given a chance to provide any additional feedback they might want to give.

7.2 Data Cleanup

In total, 202 participants completed the study, though for 36 participants the sensor API failed to initialize leaving them unable to use SenCAPTCHA. As with our first usability study, we are unsure of what causes this, though it looks like it might be related to device-specific limitations on

the use of the orientation sensor by websites.[3] For those 36 participants, we do not report their SenCAPTCHA ASQ scores or timing data nor use this data in statistical (repeated-measure) tests as these participants didn't use SenCAPTCHA and as such can't compare it to other CAPTCHA systems. We also do not report on their favorite system or most disliked systems, as the purpose of these questions was to compare performance against SenCAPTCHA. We do report their ASQ scores and timing for the other CAPTCHAs, as we believe these data points are unaffected by the problems with SenCAPTCHA and can serve as meaningful norms to be compared against by future CAPTCHA research.

7.3 Qualitative Analysis

To analyze the qualitative feedback provided by users, we used the same open coding approach based on the constant-comparative method [17] as for the first user study.

7.4 Demographics

Participants in our study were pulled from Amazon Mechanical Turk. Other than the restriction that participants must complete the study with a smartphone or tablet, there were no other limitations on who could participate in the study. Demographics from our study are given in Table 6.

8 SECOND USABILITY STUDY—RESULTS

In this section, we report on After-Scenario Questionnaire (ASQ) scores, time taken to complete each system, participants' favorite and most disliked systems, and qualitative feedback. Results are summarized in Figure 8. We use a one-way, repeated measures ANOVA as our omnibus test (except where noted) with Tukey's as our post-hoc pairwise statistical test. For brevity, we only report on the significance of these pairwise tests, not their individual p-values.

8.1 After-Scenario Questionnaire (ASQ)

SenCAPTCHA receives the highest rating of any of the CAPTCHA systems (16.14), with audio-based CAPTCHAs receiving the lowest score (9.58). There was a statistically significant omnibus effect for ASQ scores (two-way, repeated measures ANOVA—$F(4, 2907) = 123.61$, $p < 0.001$), with all pairwise tests between SenCAPTCHA and other CAPTCHAs being statistically significant. All pairwise tests between Audio and other CAPTCHAs were also statistically significant. None of the other pairwise tests were statistically significant. The interaction effect is also statistically significant ($F(8, 2907) = 6.68$, $p < 0.05$), though the differences are not practically meaningful.

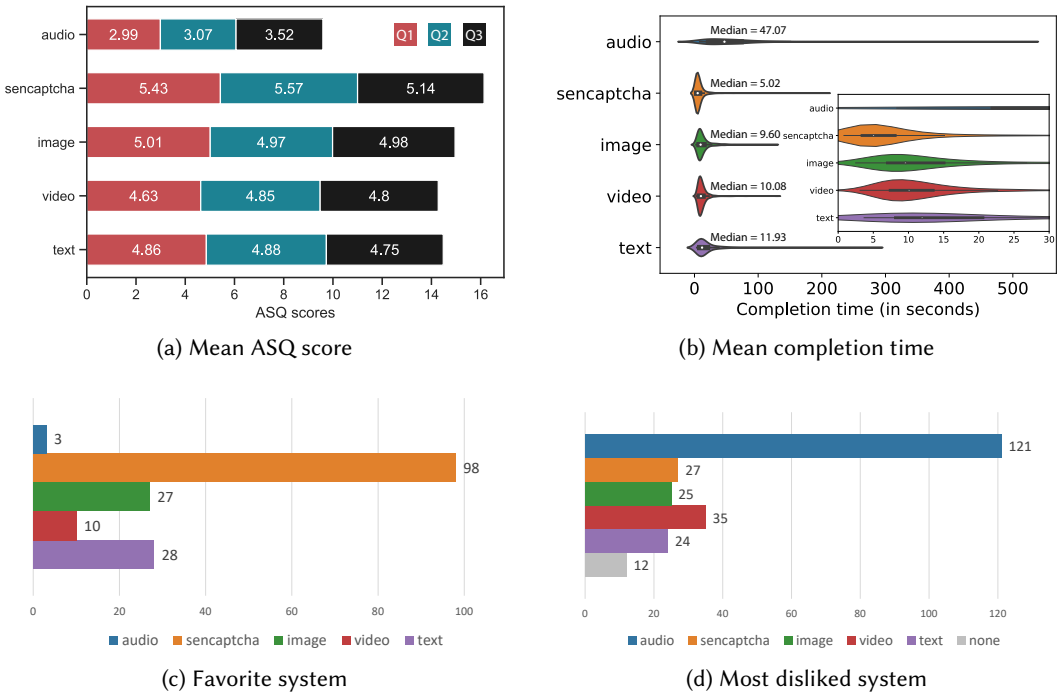
8.2 Time to Complete

SenCAPTCHA had the lowest average completion time (5.02 seconds), while audio had the longest average completion time (47.07 seconds). The remaining systems had similar average completion times (around 10 seconds). These differences were statistically significant ($F(4, 198) = 288$, $p < 0.01$), with all but two pairwise tests (SenCAPTCHA vs. video-based and image-based vs. video-based) being statistically significant.

8.3 Favorite System

Of the 166 participants that used all five CAPTCHAs, 98 (59.0%) indicated that SenCAPTCHA was their favorite system. This significantly outperformed the other four CAPTCHAs: text-based—28 (16.9%) participants, image-based—27 (16.3%) participants, video-based—10 (6.0%) participants,

[3]For example, iOS 13.1 disallows the use of the orientation sensor by websites unless this is changed in the device's settings. iOS 13.2 added an API to request that permission programmatic (which we do).



(a) Mean ASQ score

(b) Mean completion time

(c) Favorite system

(d) Most disliked system

Fig. 8. Quantitative results from the second usability study

audio-based—3 (1.8%) participants. This difference was statistically significant (χ^2 test—$\chi^2 = 134$, $p < 0.001$)

Participants strongly disliked the audio-based CAPTCHAs, with 121 (72.9%) expressing this sentiment. Participants dislikes in regard to other CAPTCHAs was evenly split: video-based—35 (21.1%) participants, SenCAPTCHA—27 (16.3%) participants, image-based—25 (15.1%) participants, text-based—24 (14.5%) participants. 12 (7.2%) participants indicated that didn't dislike any of the CAPTCHA systems. These differences were statistically significant (χ^2 test—$\chi^2 = 197$, $p < 0.001$), with audio being the only pairwise statistically significant difference ($p < 0.001$).

8.4 Refreshes, Failures, and Skips

Participants rarely refreshed the provided CAPTCHAs (means: 0.00–0.08 refreshes/attempt, median: 0). Failures were a little more common, though still uncommon for CAPTCHAs other than audio-based (mean failures/attempt: audio—1.66, SenCAPTCHA—N/A, image—0.46, video—0.11, text—0.59). Skipping (available after trying for one minute) was also rare, except for audio-based CAPTCHAs (mean number of instances skipped: audio—0.68, SenCAPTCHA—0.00, image—0.02, video—0.00, text—0.07).

8.5 Qualitative Feedback

Most participants provided feedback regarding their favorite and least favorite systems. In general, 121 (72.9%) participants indicated that their favorite system was easy, 44 (26.5%) that it was fast, and 23 (13.9%) that it was enjoyable.

SenCAPTCHA received the lion's share of positive comments: easy—54 (32.5%) participants, fast—27 (16.3%) participants, enjoyable—21 (12.7%) participants, novel—10 (6.0%) participants,

interesting—7 (4.2%) participants. Additionally, 5 (3.0%) participants indicated that they liked how SenCAPTCHA did not require any typing and 4 (2.4%) indicated that SenCAPTCHA was mobile friendly.

For the remainder of the systems, the following were the more common answers (more than 5 responses):

- 8 (4.8%) participants though that image-based CAPTCHAs were fast.
- 26 (15.7%) participants thought image-based CAPTCHAs were easy, 26 (15.7%) though the same about text-based CAPTCHAs, and 9 (5.4%) though so about video-based CAPTCHAs.
- 6 (3.6%) participants felt that image-based CAPTCHAs were understandable.

For example, participants said,

> *"It was the fastest to get passed and required the least amount of effort on my part."*
>
> *"It was novel; and I felt it might offer more security since the movements are additional info that can be taken into account when deciding if a captcha was met."*
>
> *"I did like [SenCAPTCHA] because if you're on your phone you can dip your phone around and not cause any fuss if you're in an office or something."*

In regards to things they disliked about the CAPTCHA systems, the overwhelming majority of feedback was in regards to audio-based CAPTCHAs: hard to hear or understand—66 (39.8%), too hard in general—24 (14.5%), too slow—20 (12.0%), can't be used in a public place—16 (9.6%), too much background noise making it difficult to understand—12 (7.2%), and annoying—11 (6.6%). For example, one person stated,

> *"I could not hear the letters and numbers with all the static in the background; plus the computerized voice was awful. In short; I hated it and if I ever encountered it in life; I would just quit that page or application."*

The only other comment appearing more than five times, was that 14 (8.4%) participants felt that SenCAPTCHA was hard. One participant indicated that,

> *"It was so annoying and not easy to do if you have a squirmy baby on your lap."*

Participants largely ignored the final optional feedback field and there is no meaningful data to report in that regard.

9 DISCUSSION

In this section, we summarize the lessons learned, discuss the viability of deploying SenCAPTCHA in production, and identify future research directions.

9.1 Lessons Learned

The results from our research are rather straightforward. Users enjoyed SenCAPTCHA, with nearly two-thirds of participants preferring SenCAPTCHA to other CAPTCHA schemes. Participants indicated that they found SenCAPTCHA to be easy and enjoyable, demonstrating that SenCAPTCHA achieved its three design goals. Our results also demonstrate that users are open to sensor-based CAPTCHAs, which was unclear before this research.

The fact that users choose SenCAPTCHA over existing schemes that they are familiar with also indicates that users are not satisfied with the current state of CAPTCHAs. While this might be intuitive, it is still illustrative to see demonstrated in practice. In particular, our research shows that users especially dislike audio-based CAPTCHAs. This is somewhat surprising as audio-based CAPTCHAs are commonly found as a backup method for other types of CAPTCHAs. This suggests that it might be worthwhile exploring different backup CAPTCHAs.

9.1.1 Need for Additional CAPTCHA Research. An interesting omission in the results for our second user study was that no users indicated that they no longer saw a need for CAPTCHAs or that they did not need SenCAPTCHA because they did not encounter CAPTCHAs on their mobile devices. This suggests that even with the emergence of NoCAPTCHA, that users are still encountering CAPTCHAs, including on their mobile devices. Taken together with the positive reception of SenCAPTCHA, this suggests that there remains a need for additional research into building better CAPTCHAs. Not only are users likely still encountering CAPTCHAs, but they seem to be very open and interested in the development of CAPTCHAs that are easier, faster, and more enjoyable than current solutions. This runs counter to the current trend in the field, which has seen a sharp decline in the amount of research on usable and secure CAPTCHA schemes.

9.1.2 Relationship to Prior Work. Our results regarding the relative ease of use and completion times for text-based, image-based, and video-based CAPTCHAs are in line with prior work. For example, Reynaga et al. [40] evaluated Picatcha [36], an image-based CAPTCHA and showed that completion of this CAPTCHA was faster than text-based CAPTCHAs, similar to our own results.

One notable contrast between our results and past results is in relationship to work by Brodić et al. [3]. In their research, they found that young Internet users were able to solve CAPTCHAs more quickly than older users. Based on data from both our user studies, this pattern does not seem to hold true for SenCAPTCHA, which was equally quick for users to complete regardless of age.

9.1.3 Implementation Details for Audio-Based CAPTCHAs. While implementing our audio-based CAPTCHA using the Secureimage library, we noticed that when we programmatically started the playback of the audio file using JavaScript (in response to clicking a play button), that it was difficult to hear the first few fractions of a second of the playback. While this might seem minor, it significantly decreased the success rate for solving the audio-based CAPTCHAs. In contrast, if we used the native playback interface to begin playback, then these first few fractions of a second were audible and success rates were much higher (up to 50% higher). We are not sure what made this difference but mention it here as a potential problem with the design of the Secureimage library, as the programmatic playback approach is its default.

9.2 SenCAPTCHA in Production

As with most research, the current version of SenCAPTCHA is not production-ready, but rather a proof-of-concept implementation used to demonstrate the viability of a sensor-based CAPTCHA. To prepare SenCAPTCHA for deployment, it would be necessary to address several challenges.

First, it would be necessary to have a larger corpus of images. Leveraging the mutations, we are able to generate a large number of puzzles from a single image—for example, using the tile mutation, each image from our corpus can be used to generate 9!=362,880 puzzles. Still, the base image corpus should likely number between several hundred to a couple thousand. This would provide a rich data source from which to generate mutated images and would increase the difficulty of trying to reverse the mutations. Based on our experience of generating the current image corpus, we estimate that this process would take no more than 100 hours, something that is a very reasonable time frame for the development of production-ready software.

Second, work needs to be done to understand why SenCAPTCHA sometimes fails to get data from the sensor API. In our second user study, 36 (21.7%) participants were unable to use SenCAPTCHA due to this problem. Additionally, development work is needed to locate this bug and fix it. It might also be necessary to collaborate with browser and mobile phone vendors to ensure that the sensor is available for SenCAPTCHA's use. Alternatively, SenCAPTCHA could be deployed as is, but fallback to another CAPTCHA if it is unable to access the sensor. If SenCAPTCHA were to

(a) Text Based (b) Audio Based (c) Video Based (d) Image Based (e) SenCAPTCHA

Fig. 9. How current CAPTCHAs would appear on a smartwatch.

become popular with this type of deployment, it would incentivize vendors to directly support sensor-based CAPTCHAs.

Third, it would be necessary to ensure that the current ball (a red ball with a black border) is easily visible when placed over all images in the corpus. If not, it might be necessary to calculate the color of the ball to use such that it will contrast with the underlying image.

9.3 Future Research Directions

In the remainder of this section, we identify future research directions for SenCAPTCHA.

9.3.1 SenCAPTCHA on Smartwatches. We believe that as smartwatches continue to increase their functionality, the likelihood that users will encounter CAPTCHAs on their smartwatches will likewise increase. For example, (a) viewing a website that will only display its content after presenting a CAPTCHA to prevent content scraping; (b) submitting a comment (using dictation) to a website that uses a CAPTCHA to prevent comment spam; (c) voting in an online pool that uses CAPTCHAs to prevent ballot stuffing. In each of these cases, the websites may have a responsive UI that displays cleanly on a smartwatch but will still need to show a CAPTCHA to protect against bots who would pretend to be using smartwatches if CAPTCHAs were disabled for that form factor. While the frequency of CAPTCHAs on smartwatches will likely never match their frequency on desktops or mobile phones (where account creation and management is handled), it is still worthwhile to prepare for a possible (and we believe likely) future where CAPTCHAs are occasionally encountered on smartwatches.

We believe that SenCAPTCHA would be an ideal solution for CAPTCHAs on smartwatches. We illustrate the display of multiple CAPTCHA designs on a 46mm Moto 360 smartwatch in Figure 9. Observe that those CAPTCHAs relying on users' text input have a very bad performance, because the on-screen keyboard occupies more than half of the screen. Some sub-figures in image-based and puzzle-based CAPTCHAs are hidden, which makes it hard for users to identify what the whole image looks like. On the other hand, users can still play SenCAPTCHA as it fits on the small screens more easily, and it does not require much screen space to display its content properly.

Future user studies of SenCAPTCHA running on smartwatches could help confirm our intuitions regarding SenCAPTCHA's performance on smartwatches. It could also help identify requirements and issues that are unique to smartwatches, helping the design of future mobile-first CAPTCHAs.

9.3.2 Adversarial Machine Learning. Recent advances in adversarial machine learning have shown that it is possible to make small perturbations in images such that humans cannot detect the change, but machine learning classifiers are rendered unable to classify the image. Such techniques could also be applied to SenCAPTCHA allowing it to show images without any discernable mutations, further increasing its usability and security. This approach would not only benefit SenCAPTCHA,

but other CAPTCHAs that also rely on image recognition being hard for machine learning (e.g., the image-based CAPTCHA from our user study).

9.3.3 SenCAPTCHA Using Mouse and Touch. Our implementation of SenCAPTCHA had support for moving the ball using mouse and touch, but we did not enable this functionality in our user studies as we were primarily interested in seeing how a sensor-based CAPTCHA would be received by users. During development, we extensively used the mouse-based ball movement for testing and found it to be extremely efficient and easy to use. Future research is needed to validate that users would prefer this modality, but it seems likely considering the positive reception of SenCAPTCHA on mobile devices.

It would also be possible to implement a version of SenCAPTCHA that allowed users to directly tap on the eye. We chose not to do this as we wanted to avoid the "fat finger" problem on small form factor devices, for which we think the sensor version of SenCAPTCHA is well suited. Nevertheless, future research could evaluate whether a tap-based version of SenCAPTCHA would work well on larger mobile devices such as tablets.

9.3.4 Alternative Completion Criteria. In the current implementation of SenCAPTCHA, the puzzle is completed as soon as the center of the ball and the target are within a set distance d. While this increases the speed and ease of solving SenCAPTCHA, it also somewhat limits its security as an attacker will solve the puzzle as long as the real target is on the path to the guessed target (modulo the fact that they have to pass the DTW check). We identify three alternate completion criteria that could be used to further increase the security of SenCAPTCHA. For each of these criteria, future research would be needed to establish the usability impact of adopting these approaches.

First, users could be required to hover on or near the keypoint once they have moved the ball to it. The amount of time spent hovering would likely be very short, likely no longer than half a second. This would ensure that the user is confident of the keypoint's position. With these criteria, it would likely be necessary to have the hover radius be greater than d to accommodate users' hands shaking during the hover period. Both the hover radius and timing would need to be established through additional user testing.

Second, instead of having users move a ball over a single target, we could ask them to move the ball over two different targets. After moving the ball over the second target, SenCAPTCHA would be completed immediately (i.e., without hover) just as it is now. The trajectory would then be examined to validate that the user had correctly changed trajectory after moving to the first keypoint. This would also increase the challenge of solving SenCAPTCHA by requiring an adversary not to detect just one, but two keypoints. The order in which the user would select the keypoints could likely be left up to the user. Care would need to be taken when selecting images to ensure that the keypoints are spaced a reasonable distance from each other.

Third, users could confirm that the ball is near the keypoint by clicking the screen. We choose to use a screen click as opposed to clicking a button because in our informal testing we found that asking a user to click a button causes them to rotate their screen to position their thumb over the button, whereas asking them to tap anywhere on the screen allows them to complete the CAPTCHA with no additional wrist movement. Still, usability testing would be needed to confirm that this approach works in practice.

9.3.5 Using SenCAPTCHA to Create Training Data Sets. In Von Ahn et al.'s [50, 52] on ReCAPTCHA, they described how ReCAPTCHA could be used to have users label text in images, then using those labels to help digitize books. Such an approach is especially brilliant as it takes work that is otherwise meaningless to the end-user (CAPTCHAs help servers not users) and helps produce an outcome (book digitization) that may be of use to the end-user. The question naturally arises

then if SenCAPTCHA could be used to label keypoints on images to similar help with research in computer vision.

Unfortunately, the current implementation of SenCAPTCHA does not lend itself to accomplish that goal. In contrast, the two alternative completion strategies listed above have the potential to be used to label keypoints. For the first (hovering) and third (clicking) completion criteria, we can use unlabeled images and label the point that the user selects as the keypoint. Based on measurements from multiple users, we can calculate a more precise position for the keypoint. Still, care must be taken that only a small number of unlabeled images are presented to users, as the system has no way of verifying the correctness of the presented solution.

10 CONCLUSION

In this paper, we propose SenCAPTCHA, a novel mobile-first CATPCHA. Our security analysis demonstrates that SenCAPTCHA is resilient to the best existing animal facial detection algorithms, even when those algorithms are trained with the images used by SenCAPTCHA. While this does not prove that SenCAPTCHA is impervious to attack, it does suggest that it is resilient to the best current methods and that successfully attacking SenCAPTCHA will be non-trivial. Our second usability study also demonstrate that compared to other existing CAPTCHAs (audio-, image-, text-, and video-based), SenCAPTCHA has higher perceived usability and faster completions times. Additionally, it is identified as the favorite CAPTCHA system by over half of our participants. These results are very encouraging and suggest that SenCAPTCHA successfully achieved its design goals and that it represents a meaningful contribution to the space of proposed CAPTCHAs.

REFERENCES

[1] Henry S Baird, Michael A Moll, and Sui-Yu Wang. 2005. ScatterType: a legible but hard-to-segment CAPTCHA. In *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*. IEEE, 935–939.

[2] M.M. Bongard. 1970. *Pattern Recognition*. Spartan Books, Rochelle Park, NJ.

[3] Darko Brodić, Alessia Amelio, and Radmila Janković. 2018. Exploring the influence of CAPTCHA types to the users response time by statistical analysis. *Multimedia Tools and Applications* 77, 10 (2018), 12293–12329.

[4] John Brooke. 2013. SUS: a retrospective. *Journal of usability studies* 8, 2 (2013), 29–40.

[5] John Brooke et al. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.

[6] Elie Bursztein. 2012. How we broke the NuCaptcha video scheme and what we proposed to fix it. *See https://www.elie.net/blog/security/how-we-broke-the-nucaptcha-video-scheme-and-what-we-propose-to-fix-it/, Accessed March* (2012).

[7] Elie Bursztein, Jonathan Aigrain, Angelika Moscicki, and John C Mitchell. 2014. The End is Nigh: Generic Solving of Text-based CAPTCHAs.. In *WOOT*.

[8] Tsz-Yan Chan. 2003. Using a test-to-speech synthesizer to generate a reverse Turing test. In *Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on*. IEEE, 226–232.

[9] Monica Chew and J Doug Tygar. 2004. Image recognition captchas. In *International Conference on Information Security*. Springer, 268–279.

[10] Sarika Choudhary, Ritika Saroha, Yatan Dahiya, and Sachin Choudhary. 2013. understanding CAPTCHA: Text and Audio Based Captcha with its Applications. *International Journal of Advanced Research in Computer Science and Software Engineering* 3, 6 (2013).

[11] Mauro Conti, Claudio Guarisco, and Riccardo Spolaor. 2016. CAPTCHaStar! A novel CAPTCHA based on interactive shape discovery. In *International Conference on Applied Cryptography and Network Security*. Springer, 611–628.

[12] Matthias Dantone, Juergen Gall, Gabriele Fanelli, and Luc Van Gool. 2012. Real-time facial feature detection using conditional regression forests. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2578–2585.

[13] Ritendra Datta, Jia Li, and James Z Wang. 2005. IMAGINATION: a robust image-based CAPTCHA generation system. In *Proceedings of the 13th annual ACM international conference on Multimedia*. ACM, 331–334.

[14] Jeremy Elson, John R Douceur, Jon Howell, and Jared Saul. 2007. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization.. In *ACM Conference on Computer and Communications Security*, Vol. 7. Citeseer, 366–374.

[15] Haichang Gao, Honggang Liu, Dan Yao, Xiyang Liu, and Uwe Aickelin. 2010. An audio CAPTCHA to distinguish humans from computers. In *Electronic Commerce and Security (ISECS), 2010 Third International Symposium on*. IEEE,

265–269.
[16] Haichang Gao, Dan Yao, Honggang Liu, Xiyang Liu, and Liming Wang. 2010. A novel image based CAPTCHA using jigsaw puzzle. In *Computational Science and Engineering (CSE), 2010 IEEE 13th International Conference on*. IEEE, 351–356.
[17] Barney G Glaser. 1965. The constant comparative method of qualitative analysis. *Social problems* 12, 4 (1965), 436–445.
[18] Philippe Golle. 2008. Machine learning attacks against the Asirra CAPTCHA. In *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 535–542.
[19] Ian J Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. 2013. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082* (2013).
[20] Rich Gossweiler, Maryam Kamvar, and Shumeet Baluja. 2009. What's up CAPTCHA?: a CAPTCHA based on image orientation. In *Proceedings of the 18th international conference on World wide web*. ACM, 841–850.
[21] Jonathan Holman, Jonathan Lazar, Jinjuan Heidi Feng, and John D'Arcy. 2007. Developing usable CAPTCHAs for blind users. In *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility*. ACM, 245–246.
[22] Chen-Chiung Hsieh and Zong-Yu Wu. 2013. Anti-SIFT images based CAPTCHA using versatile characters. In *Information Science and Applications (ICISA), 2013 International Conference on*. IEEE, 1–4.
[23] Google Inc. 2018. *Google reCAPTCHA*. https://www.google.com/recaptcha/intro/
[24] Leap Marketing Technologies Inc. 2010. Video-Based Captchas Now Available for Sites and Blogs. *See www.prnewswire.com/news-releases/video-based-captchas-now-available-for-sites-and-blogs-97471319.html* (2010).
[25] Imperva Incapsula. 2016. *Bot Traffic Report*. https://www.incapsula.com/blog/bot-traffic-report-2016.html
[26] Ramanpreet Kaur and Pooja Choudhary. 2015. A Novel CAPTCHA Design Approach using Boolean Algebra. In *2015 5th International Conference on IT Convergence and Security (ICITCS)*. Citeseer, 1–7.
[27] Kurt Alfred Kluever and Richard Zanibbi. 2009. Balancing usability and security in a video CAPTCHA. In *Proceedings of the 5th Symposium on Usable Privacy and Security*. ACM, 14.
[28] Greg Kochanski, Daniel P Lopresti, and Chilin Shih. 2002. A reverse turing test using speech.. In *INTERSPEECH*.
[29] Martin Koestinger, Paul Wohlhart, Peter M Roth, and Horst Bischof. 2011. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. IEEE, 2144–2151.
[30] Jonathan Lazar, Jinjuan Feng, Tim Brooks, Genna Melamed, Brian Wentz, Jon Holman, Abiodun Olalere, and Nnanna Ekedebe. 2012. The SoundsRight CAPTCHA: an improved approach to audio human interaction proofs for blind users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2267–2276.
[31] James R Lewis. 1991. Psychometric evaluation of an after-scenario questionnaire for computer usability studies: the ASQ. *ACM Sigchi Bulletin* 23, 1 (1991), 78–81.
[32] Mark D Lillibridge, Martin Abadi, Krishna Bharat, and Andrei Z Broder. 2001. Method for selectively restricting access to computer systems. US Patent 6,195,698.
[33] David G Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision* 60, 2 (2004), 91–110.
[34] Nicholas J. Hopper Luis von Ahn, Manuel Blum and John Langford. 2000. *The CAPTCHA Web Page*. http://www.captcha.net
[35] Peter Matthews, Andrew Mantel, and Cliff C Zou. 2010. Scene tagging: image-based CAPTCHA using image composition and object relationships. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*. ACM, 345–350.
[36] D Mujumdar and S Polisetti. 2011. A platform to monetize usable & secure CAPTCHAs for desktop and mobile devices (PICATCHA). *Retrieved Dec* 20 (2011), 2014.
[37] Drew Phillips. 2019. *Securimage*. https://www.phpcaptcha.org
[38] Lawrence R Rabiner and Biing-Hwang Juang. 1993. Fundamentals of speech recognition. (1993).
[39] Maheen Rashid, Xiuye Gu, and Yong Jae Lee. 2017. Interspecies Knowledge Transfer for Facial Keypoint Detection. *arXiv preprint arXiv:1704.04023* (2017).
[40] Gerardo Reynaga, Sonia Chiasson, and Paul C van Oorschot. 2015. Exploring the usability of captchas on smartphones: Comparisons and recommendations. In *NDSS Workshop on Usable Security USEC*.
[41] Scott Ruoti, Brent Roberts, and Kent Seamons. 2015. Authentication melee: A usability analysis of seven web authentication systems. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 916–926.
[42] Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing* 26, 1 (1978), 43–49.
[43] Jeff Sauro and James R Lewis. 2016. *Quantifying the user experience: Practical statistics for user research*. Morgan Kaufmann.

[44] Suphannee Sivakorn, Iasonas Polakis, and Angelos D Keromytis. 2016. I am robot:(deep) learning to break semantic image captchas. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 388–403.

[45] Suphannee Sivakorn, Jason Polakis, and Angelos D Keromytis. 2016. I'm not a human: Breaking the Google reCAPTCHA. *Black Hat* (2016).

[46] Oleg Starostenko, Claudia Cruz-Perez, Fernando Uceda-Ponga, and Vicente Alarcon-Aquino. 2015. Breaking text-based CAPTCHAs with variable word and character orientation. *Pattern Recognition* 48, 4 (2015), 1101–1112.

[47] Yi Sun, Xiaogang Wang, and Xiaoou Tang. 2013. Deep convolutional network cascade for facial point detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3476–3483.

[48] Pranal C Tayade and Mahip M Bartere. 2015. Comprehensive study on performance analysis of various CAPTCHA systems. *International Journal of Current Engineering and Technology* 5, 1 (2015).

[49] Thomas S Tullis and Jacqueline N Stetson. 2004. A comparison of questionnaires for assessing website usability. In *Usability professional association conference*, Vol. 1. Minneapolis, USA.

[50] Luis Von Ahn. 2008. Human computation. In *2008 IEEE 24th international conference on data engineering*. IEEE, 1–2.

[51] Luis Von Ahn, Manuel Blum, and John Langford. 2004. Telling humans and computers apart automatically. *Commun. ACM* 47, 2 (2004), 56–60.

[52] Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. 2008. recaptcha: Human-based character recognition via web security measures. *Science* 321, 5895 (2008), 1465–1468.

[53] Yi Xu, Gerardo Reynaga, Sonia Chiasson, Jan-Michael Frahm, Fabian Monrose, and Paul C van Oorschot. 2012. Security and Usability Challenges of Moving-Object CAPTCHAs: Decoding Codewords in Motion.. In *USENIX security symposium*. 49–64.

[54] Heng Yang, Renqiao Zhang, and Peter Robinson. 2016. Human and sheep facial landmarks localisation by triplet interpolated features. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*. IEEE, 1–8.