DotCHA: An Interactive 3D Text-based CAPTCHA

Suzi Kim and Sunghee Choi*

School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea
E-mail: kimsuzi@kaist.ac.kr; sunghee@kaist.edu
**Corresponding Author*

Received 30 August 2019; Accepted 18 December 2019;
Publication 22 January 2020

Abstract

We introduce a new type of 3D text-based CAPTCHA, called DotCHA, which relies on human interaction and overcomes the limitations of existing 2D and 3D CAPTCHAs. DotCHA asks users to rotate a 3D text model to identify the correct letters. The 3D text model is a twisted form of sequential 3D letters around a center pivot axis, and it shows different letters depending on the rotation angle. Because each model consists of many small spheres instead of a solid letter model, DotCHA is classified as a scatter-type CAPTCHA and resists character segmentation attacks. Moreover, DotCHA is resistant to machine learning attacks because each letter is only identified in a particular direction. We demonstrate that DotCHA is resistant to existing types of attacks while maintaining usability.

Keywords: CAPTCHA, 3D CAPTCHA, text-based CAPTCHA, 3D typography, mental rotation, security, usability.

1 Introduction

Completely Automated Public Turing tests to tell Computers and Humans Apart (CAPTCHA) [37] was developed to protect systems from denial of

Journal of Web Engineering, Vol. 18_8, 837–864.
doi: 10.13052/jwe1540-9589.1884

service (DoS) attacks by malicious automated programs. It is a Turing test to discriminate human users from malicious bots by using tasks that humans can perform easily but machines cannot. Two-dimensional (2D) CAPTCHAs are the most commonly used form of CAPTCHA to identify humans. However, the rapid enhancement of machine learning has enabled bots to overcome the 2D text-based CAPTCHAs [7, 9, 16] and 2D image-based CAPTCHAs [35] with high accuracy. In particular, with advances in Optical Character Recognition (OCR), sophisticated attacks have been introduced to break 2D text-based CAPTCHAs. These attacks have led to the development of three-dimensional (3D) CAPTCHAs that are relatively hard to be decoded by computers.

3D CAPTCHAs are categorized into two types: 3D model-based CAPTCHAs and 3D text-based CAPTCHAs. Previous 3D text-based CAPTCHAs were formed by a simple extrusion of alphabets, making them easily recognizable with a single glance. However, the simple extrusion is vulnerable to OCR attacks because it has the same visual effects as distorted 2D text. 3D model-based CAPTCHAs have improved security by reducing usability, which is the strength of 3D text-based CAPTCHAs. They are based on the mental rotation ability [32, 33], which enables users to find answers by inferring from the direction of various 3D objects [18, 31, 38], such as vehicles and animals. However, 3D model-based CAPTCHAs have a low correct response rate, because they require the users to not only recognize the 3D object but also judge and infer the answer by performing elaborate operations. As a result, it takes a long time to obtain the right answer for 3D model-based CAPTCHAs compared with 3D text-based CAPTCHAs. Moreover, for those who are accustomed to conventional text-based CAPTCHAs, 3D model-based CAPTCHAs cause usability issues.

In a previous work [22], we proposed a new type of 3D CAPTCHA, called *DotCHA*, which satisfies both security and usability. DotCHA combines the strengths of text-based CAPTCHAs and 3D model-based CAPTCHAs. It presents different alphabets that are rotated at different angles. The alphabets are composed of several small spheres instead of being shown as a single solid model so that it is resistant to segmentation attack. DotCHA provides usability to users who are familiar with text-based CAPTCHAs while preserving the security of 3D model-based CAPTCHAs. In this paper, we extend our previous work by presenting further experiments validating the security and usability of DotCHA. In Section 2, we briefly review previous works. Section 3 describes the generation of DotCHA, and Section 4 evaluates DotCHA using some attack scenarios. The prototype implementation and source code are available at https://github.com/suzikim/dotcha.

2 Related Work

In this section, we briefly discuss three main types of CAPTCHAs which are closely related to DotCHA: 2D text-based, 3D model-based, and interactive CAPTCHAs (Figure 1).

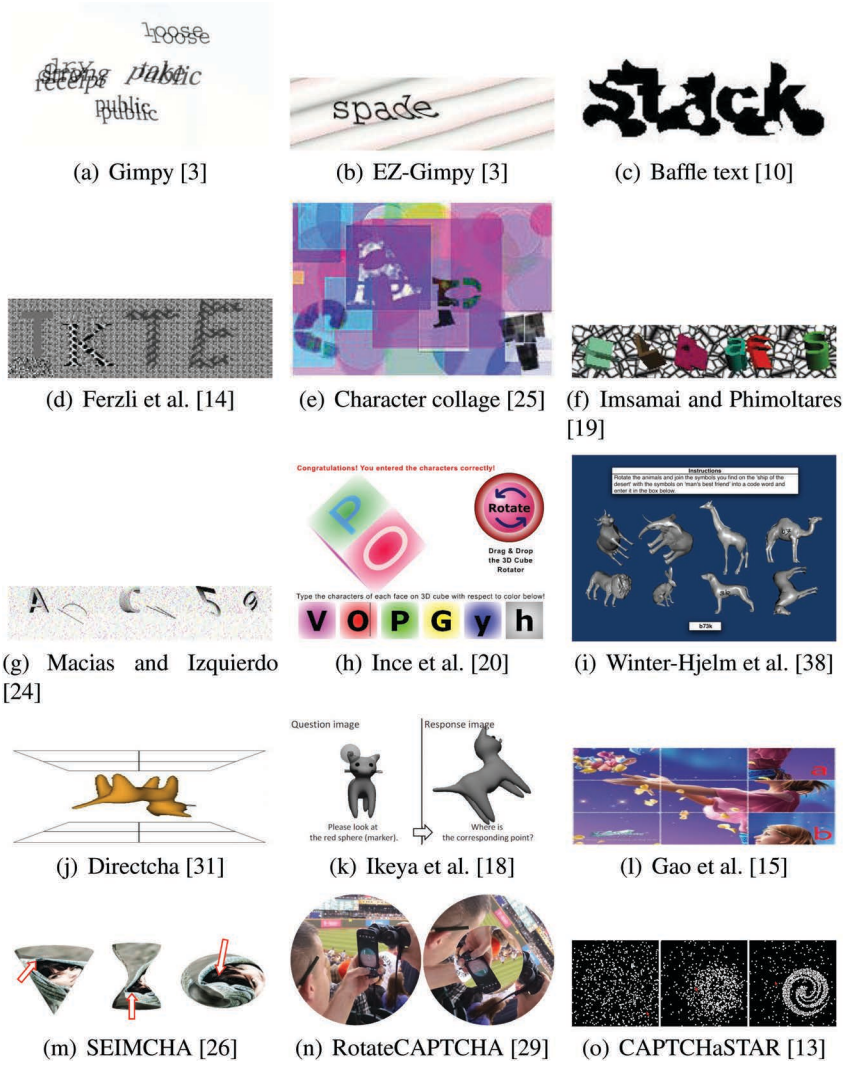


Figure 1 Examples of previous 2D text-based, 3D model-based, and interactive CAPTCHAs.

2.1 2D Text-based CAPTCHAs

2D text-based CAPTCHAs are the most widely used form, due to their ease of use and simple structure. A sequence of alphabets and numbers are presented to a user, and the user should identify the correct text to pass the test. Usually, noise and distortion appear in the letters to make the test robust to automated attacks.

Gimpy and EZ-Gimpy [3] are based on the human ability to read heavily distorted and corrupted text. Gimpy picks up 10 random words from the dictionary and arranges them to be overlapped with each other. Users have to identify at least three words to pass the test. EZ-Gimpy uses only one random word from the dictionary, but instead increases its security by deformation, blurring, noise, and distortion of letters. Mori and Malik [28] break Gimpy and EZ-Gimpy using object recognition algorithms and dictionary attacks.

Baffle text [10] minimizes the instances of dictionary attack by using pseudorandom but pronounceable words. The users are asked to infer the correct answer from words that have missing parts of letters. MSN Passport CAPTCHA [11] has eight characters, including alphabets and numbers, which are highly warped to distort the characters.

The prime advantages of 2D text-based CAPTCHAs are that they are easy to generate and identify. However, it is also easy to be recognized through OCR attacks [34, 39]. More advanced 2D text-based CAPTCHAs have been introduced [14, 25]; however, they have been easily broken by machine learning attacks [7, 9, 16].

2.2 3D Model-based CAPTCHAs

3D CAPTCHAs are designed to defeat bots, which use machine learning to identify 2D text-based CAPTCHAs easily, by minimizing the legibility. Most of the 3D model-based CAPTCHAs are based on the rotation of 3D models [18, 31, 38]. They take advantage of the cognitive ability of humans, called mental rotation [32, 33], which is an inherent characteristic of human nature. Mental rotation enables humans to compare two models in different orientations. However, users feel difficult to deal with an unfamiliar model, so they spend a long time to solve the problem.

3D text-based CAPTCHAs are more familiar to users because they originated from text-based CAPTCHAs that users have been accustomed to. Ince et al. [20] introduce a cubic-style 3D CAPTCHA, which contains six alphabets on each side of a cube. However, it is vulnerable to segmentation attacks because the letters are simply attached to each side of the cube without

interference. There have been several CAPTCHAs using a sequence of 3D letter models created with extrusion and warping [19, 24]. Ye et al. [42] demonstrate that CAPTCHAs with such a simple distorted form could easily be broken with a high degree of accuracy.

2.3 Interactive CAPTCHAs

Interactive CAPTCHAs rely on user interaction to mitigate automated attacks. They require users to solve the problem by cognitive abilities and human actions. The 3D model-based CAPTCHAs [20, 38], which require rotation of the 3D model to get the answers, also belong to the interactive CAPTCHAs.

2D images are the most commonly used sources in interactive CAPTCHAs. Gossweiler et al. [17] introduce a 2D-image based interactive CAPTCHA that requires users to rotate a randomly oriented image to its upright orientation. SEIMCHA [26] applies geometric transformations to 2D images, and users are required to identify the upright orientation of the image. Gao et al. [15] introduce a CAPTCHA that asks users to solve a jigsaw puzzle of a 2D image divided into small pieces and shuffled. In rotateCAPTCHA [29], which combines the orientation and puzzle-solving techniques, the users have to recreate the original image by rotating the segmented image.

CAPTCHaStar [13] requires the users to change the position of a set of randomly scattered small squares by moving the cursor to recognize the correct shape. Our DotCHA adopts the movement of small scattered objects from CAPTCHaStar to ensure resistance to random guessing attacks [23]. The main problem with interactive CAPTCHAs, which focus more on security than usability, is that it is difficult and time-consuming to solve the problem. This is because users are not accustomed to solving visual problems. Our DotCHA combines the strengths of text-based and interactive CAPTCHAs to satisfy both security and usability.

3 Generation of DotCHA

3.1 Requirements

DotCHA is designed to satisfy both security and usability by improving the security of 2D text-based CAPTCHAs and the usability of 3D model-based CAPTCHAs. DotCHA is a kind of 3D typography because it is a text-based 3D model. Kim and Choi [21] introduced an automatic generation of 3D typography, which makes a single model appear to be different letters

depending on the view direction. Figure 2 shows *DotCHA-C* which adopts Kim and Choi's 3D typography. It has a form in which up to three letters are engraved on each side of a cube. Since one cube represents three letters, it has the advantage of excellent embedding ability. However, as Kim and Choi mentioned, it is not possible to embed all the combinations of letters. Also, there is a lack in usability because users need to identify letters by rotating the model in three Degrees of Freedom (3-DoF).

To increase the usability, *DotCHA-L* is developed by engraving only one letter in a cube and arranging several cubes in a row. Figure 3 shows the shape of each cube rotated along its local axis to achieve a similar effect to the existing 2D text-based CAPTHCA. Although the rendered form is similar to the existing 2D text-based CAPTCHA, it has the advantage of being able to check the letter interactively. However, as mentioned above, excessive autonomy of 3-DoF leads to lack of usability in finding the correct direction of each letter.

Therefore, to reduce DoF, we generate a DotCHA that has a common axis but another axis is unique to each letter. All letters of the 2D text-based CAPTCHAs are visible at once, and the letters are legible because

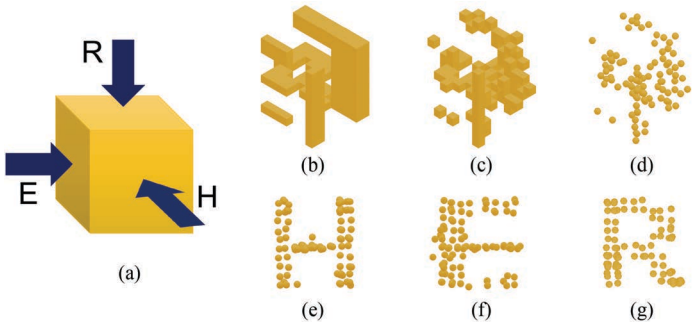
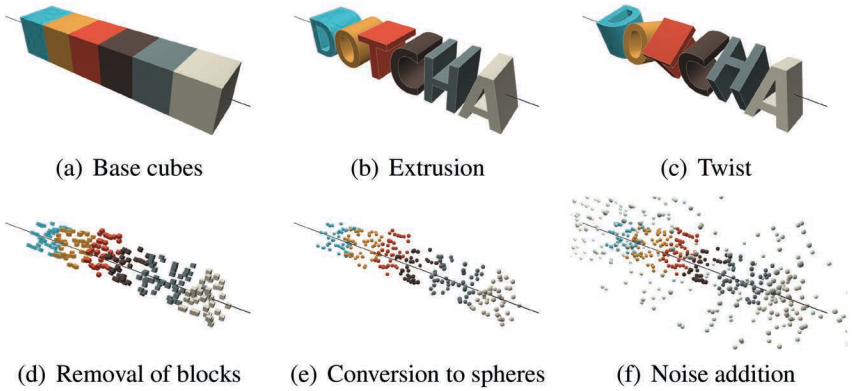


Figure 2 DotCHA-C is an adoption of the 3D typography from Kim and Choi [21]. (a) For a given cube, (b) we engrave letters on each side of the cube, (c) remove unnecessary blocks, and (d) convert blocks to spheres. The projection results from each side represent answer letters: (e) H, (f) E, and (g) R.

Figure 3 DotCHA-L is generated by engraving only one letter in a cube and arranging several cubes in a row. Each cube is rotated along its local axis.



(a) Base cubes (b) Extrusion (c) Twist

(d) Removal of blocks (e) Conversion to spheres (f) Noise addition

Figure 4 Sample results of the DotCHA generation.

of the clear form of lettering; therefore, it is vulnerable to OCR attacks. Each letter of DotCHA is only legible at a unique rotation angle, which makes this technique robust to OCR attacks, by twisting 3D extruded models around a center axis, as shown in Figure 4(c). In addition, we remove visual unnecessary parts of the models so that each letter is not read in any direction other than in its unique direction, as shown in Figure 4(d).

In order to improve usability, the contents of DotCHA include just 3D letter models instead of other visual representations, such as images or object models. The rotation axis of DotCHA is fixed to a single axis, shown as a black bar in Figure 4, to reduce the burden and confusion for users, who may wander around the 3D space. We replace remaining cubes with spheres to prevent direction attacks, which guess the unique orientation of each letter by aligning the edges of small cubes, and more details will be given in Section 3.4. Figure 5 shows the pipeline to generate a DotCHA from given target alphabets.

3.2 Extrusion and Twist of 3D Model

We use the molecular construction method [21, 27] to engrave the given letters on a solid rectangular parallelepiped model. Molecular construction [27] is a technique in which a model is divided into smaller units forming the larger model. The basic idea of generating a DotCHA involves cutting a solid cube model into small unit blocks and then deleting unnecessary blocks or adding missing blocks to represent the given letters. Figure 4(b) and 4(c) show the results of extrusion and twist, respectively. One letter has a size of $k \times k \times k$, and it retains the same shape as an extrusion of a 2D letter pattern.

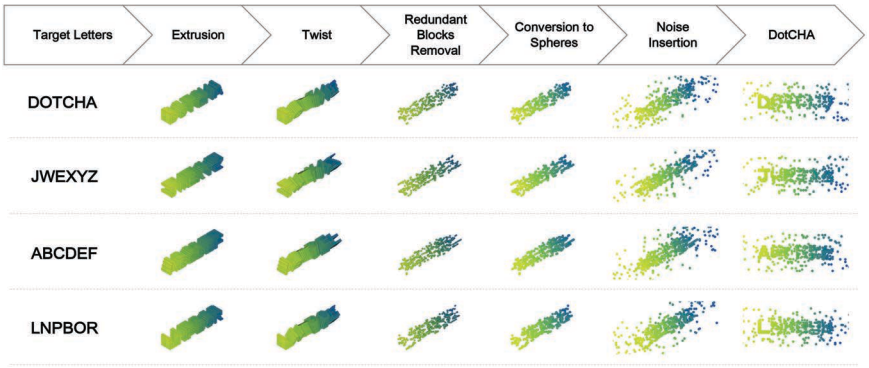


Figure 5 System Pipeline to generate DotCHA. Target letters are extruded and twisted to the 3D model, and split to small unit blocks. We remove redundant blocks which do not affect the perception of letters. The remaining blocks are converted to spheres, and noise spheres are added around the model.

The 3D letter models are then rotated around the center axis of the rectangular parallelepiped model at unique angles to ensure that the correct answers are not recognized from a single direction. If the number of letters used in a CAPTCHA is n, the correct answer of the DotCHA can be identified only if the machine finds all n directions.

3.3 Removal of Redundant Blocks

Although not all letters are visible at once in defense of segmentation attacks, a twisted model is still vulnerable to OCR attack. We remove a set of unit blocks from the models so that each letter is recognized only in one particular direction, not in any direction. The blocks are removed based on two conditions. Firstly, we remove unnecessary blocks that do not affect the shape of the letters. If two blocks are placed side by side along one axis, even if one block is discarded, it is still recognized as a letter in a certain direction. Secondly, the blocks are evenly removed while preserving the balance between directions, because the letters can be easily identified in an arbitrary direction if the blocks are gathered.

We use a multigraph G, which has multiple edges between a pair of vertices. Each unit block is represented as a vertex of the graph G. Two unit blocks are connected by two types of edges depending on whether the blocks are located on the same coordinates along the y or z axes, as shown in Figure 6. Since the rotation axis is fixed to the x-axis and the overlapping

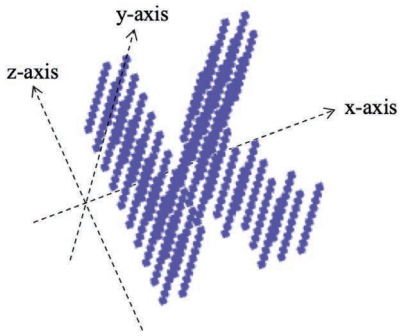


Figure 6 xyz axes of multigraph.

in x coordinates does not affect the recognition of the letter, we ignore the x-axis in G. We score all the vertices according to the scoring function S of vertex v as follows:

$$S(v) = \alpha \cdot |N_R(v)| + |N_G(v)|\,, \tag{1}$$

where $N_G(v)$ is a set of adjacent vertices of v in graph G and $N_R(v)$ is the set of neighboring vertices. For a given vertex v, we define a neighboring vertex as the one whose Euclidean distance from vertex v is at most k. The first term is related to the dispersion of blocks in the cube. It indicates the number of blocks that exist around the block v. The second term is for calculating the number of blocks that are placed along the y and z axes. α is a constant for balancing between the two terms. We used $\alpha = 0.3$ in our experiments.

We iterate the vertices in the descending order of the scores to decide whether to remove the block or not. At each iteration, unless the block is the only block placed along the y or z axes, it can be discarded from the DotCHA model. To avoid the deterministic removal for a given μ, α, and k, the target vertex v' of each iteration is removed with the following probability $\Psi(v')$:

$$\Psi(v') = rank(v')/\,|V(G)|\,, \tag{2}$$

where $|V(G)|$ is the number of vertices in graph G and $rank(v')$ is the rank of v' among vertices in G. The higher the $S(v)$, the more likely it is to be deleted. However, since it may not be deleted according to the probability, the randomness of the removal is guaranteed. The iterations stop when the number of iterations exceeds μnk^3, where μ represents the removal ratio and nk^3 is the volume of the bounding box. Large μ makes it take a long time for

the user to find the correct answer, while small μ makes the security weaker; therefore, an appropriate balance of μ is important. We used $\mu = 0.8$ in our experiments.

3.4 Prototype Implementation

Since the direction of cube blocks can be inferred from the edges of the cube, the orientation of letters can be easily identified. To hide the orientation of the model, we convert the unit blocks into spheres, and it has a similar result to the scatter-type method [6]. The post-processing involves three parameters to maximize usability and security:

- *Sphere radius (ρ)*: the radius of the sphere converted from the unit block. The edge length of a unit cube is 1, and $\rho = 1$ means that a sphere fits exactly into the cube without any cutoff or margins.
- *Sphere offset (σ)*: the location offset of the center of the sphere from the center of the unit block. $\sigma = 0$ means that the centers of the sphere and unit cube are identical, and $\sigma = 0.5$ means that the center of the sphere exists on the surface of the unit cube.
- *Noise (δ)*: the number of noise spheres.

After the redundant blocks are removed, δ noise spheres are added to the model to prevent recognition by automated machines. The region of noise spheres is three times bigger than nk^3, and excludes the bounding box of DotCHA. This is based on the concept of motion parallax, which gives users the perception of depth from the relative motion between models [12]. The noise spheres appear to move faster or slower as compared to the alphabet spheres, and the user can distinguish them by the human visual system of depth perception. We set ρ to be smaller than the half-edge of the unit block to avoid edge detection attacks. Each sphere is randomly translated within the range of $(0, \sigma)$ to avoid pixel-counting attacks.

The rotation axis of DotCHA is fixed to the x-axis in order to reduce the burden and confusion for users. Also, we support both automatic and interactive rotation to improve usability. DotCHA is implemented using Three.js library, which is a lightweight 3D engine, on HTML5 Canvas, so that it is supported by a majority of the browsers. We use $k = 10$ alphabet pattern with Consolas font. To defend against segmentation attack, random clutter spheres are added to the background, which makes it difficult to separate the foreground and background to identify the letters.

4 Experiments

We analyzed the security of DotCHA by considering several different attack scenarios. The goal of all the attacks is twofold: (1) to find the correct view directions to identify the letters and (2) to read the letters in the selected view directions. For the first goal, we tested whether a particular view direction can be characterized by the attacks. We tested the second goal by applying OCR to read the sampled images. We used $n = 6$ letters of DotCHA, and a combination of random alphabet letters was used to avoid dictionary attack.

4.1 Validation of Each Phase

Our pipeline consists of five steps: extrusion, twist, redundant blocks removal, conversion to spheres, and noise insertion, as shown in Figure 5. To validate the effect of each stage on the security of DotCHA, we conducted OCR tests on the interim results of each step. We fixed the rotation angle around the x-axis with the correct direction of the first letter. We used Google Tesseract [36] and ABBYY FineReader 14 [2] for OCR engines and Table 1 shows the test results. For incomplete models that do not go through all the steps, some letters are easily noticeable by OCR engines. The results show that every step is helping to increase security while achieving its goals.

Table 1 Validation results of each phase

	'DOTCHA'	Tesseract	ABBYY	'ZFEKLB'	Tesseract	ABBYY
Extrusion		*DOTCHA*	*DOTCHA*		*ZFEKLB*	*ZFEKLB*
Twist		*DOvany*	*OVUM*		*ZEMEB*	ZPINPI
Redundant blocks removal		*oeteny*	N/A		*ZPRRIR*	N/A
Conversion to spheres		*DETENY*	*Drow*		*ZPRRER*	*Z&mm*
Sphere radius adjustment		*Daheny*	*YIJy*		*Pe*	N/A
Random sphere offset		*Daren*	*CIW*		*2PRREA*	*mm*
Noise insertion		N/A	N/A		N/A	N/A

4.2 Finding the Correct View Directions

As mentioned in Section 3.2, an automated attack should find the $n = 6$ correct view directions to identify the correct answer. We sampled 30 different views including six ground truth views and scored them through pixel counting and edge detection.

4.2.1 Pixel-counting attack

Pixel-counting attack is based on two assumptions: first, the wider the overlap between the spheres, the clearer the shape of letters; second, the narrower the overlap between the spheres, the more information that can be represented. We counted the number of non-background pixels and checked whether the correct views can be distinguished from incorrect views by the number of pixels. We confirmed that the correct view directions cannot be identified by pixel counting, as described in Table 2, which shows the results of the pixel-counting attack when $\rho = 0.5$, $\sigma = 0.2$, and $\delta = 0.001$. It shows that the correct answers are ranked arbitrarily regardless of the pixel counts. In the process of converting the unit blocks into spheres, we added a random offset to the position of the sphere, and this makes DotCHA robust to pixel counting.

DotCHA requires several segmentations, as many as the number of letters, and this becomes an overhead to repeat. Moreover, segmentation can be avoided by increasing the range of the random offset of spheres. Even if the segmentation works well, DotCHA is still resistant to pixel-counting attack,

Table 2 Results of scoring with pixel counting. Thirty views have been ranked through pixel counting, and the views of the largest and smallest pixel counts are shown in order. In addition, the pixel counting results of the correct views are shown in the right column with correct letters. The pixel-counting attacks failed to find the correct view, and it shows that the correct view cannot be distinguished by pixel counting

Pixel Counting Result				Ground Truth	
Largest		Smallest			
	241,608		238,191	A	 240,544
	241,027		238,435	T	 240,497
	240,956		238,518	C	240,276
	240,949		238,710	O	 239,316
	240,806		238,855	H	239,180
	240,544		238,896	D	 239,119

because the number of pixels varies depending on the view direction and clutter, as depicted in Table 3. Furthermore, even if some segmentations succeed, it is impossible to guess the entire word from only a few letters obtained through segmentation, because DotCHA does not use dictionary words.

4.2.2 Edge detection attack

This criterion aims to find the correct view directions by detecting edges from the original images. We ran Canny edge detection [8] on every sampled image. A DotCHA model was projected onto a 2D text form after removing unnecessary pixel information via edge detection. We counted the number of pixels in the edge-detected images to distinguish the correct view directions from the irrelevant view directions.

There was no correlation between the correct view directions and the number of edges, as shown in Table 4. While converting the unit blocks into spheres, we made the sphere smaller than the unit block. As a result, the spheres were separated from each other, and edge detection showed the edges of spheres, which lowered the prominence of the edges of letters.

4.2.3 Geometric attack

Because DotCHA is rendered in orthographic projection instead of perspective projection, all dot seems to have the same size regardless of the depth. It makes users feel like the dot is moving in 2D coordinates rather than 3D. Therefore, an attacker may separate dots and calculate the distances between the dots to infer the correct view direction. Of course, it is another tricky issue to distinguish the grouped dots into individual dots. We do not set lighting for the rendering, and it makes the border of dots invisible.

However, we assume that the attacked separates all dots somehow to show the robustness of DotCHA against geometric attack. We use two measurements: average distance and scattering. The average distance is calculated

Table 3 Pixel counts from different views of letter 'D', when $\rho = 0.4$, $\sigma = 0.3$, and $\delta = 0.001$. The pixel counts vary depending on the rotation angle, and it is resistant to pixel-counting attacks

					
Pixel Counts	475	451	422	416	379

Table 4 Results of scoring with edge counting. Thirty views have been ranked through edge counting, and the views of the largest and smallest edge counts are shown in order. In addition, the edge counting results of the correct views are shown in the right column with correct letters. The edge counting attacks failed to find the correct view, and it shows that the correct view cannot be distinguished by edge counting. $\rho = 0.5$, $\sigma = 0.2$, and $\delta = 0.001$ were used for edge counting

Edge Detection Result				Ground Truth	
Largest		Smallest			
	12,668		11,856	H	 12,445
	12,564		11,788	O	 12,335
	12,507		11,784	D	 12,318
	12,459		11,742	C	 11,981
	12,445		11,693	T	 11,924
	12,404		11,685	A	 11,784

from the distance between each dot. To calculate the scattering, we measure the distance between each dot and center coordinate. Table 5 is the results of the average of distances between dots. It shows that the distances between dots do not correlate with the correct view.

4.3 Reading the Letters from the Correct View Directions

In the previous subsection, we showed that it is difficult to find the correct view direction automatically. For the experiment described in this subsection, we tested the possibility of reading letters from the given correct view directions when we assumed that the machine somehow found the correct view direction.

4.3.1 Pixel-counting attack

A pixel-counting attack [39, 41] counts the number of pixels of each segmented letter by the vertical histogram of a CAPTCHA image. The number of pixels is then mapped to the lookup table, which contains precomputed numbers of every alphabet.

The most important part of segmentation is the removal of background or clutters. However, it is difficult to remove them from DotCHA, because clutter spheres look similar to spheres that form the letter model. As a result, it disturbs the segmentation through vertical histogram, as shown in Figure 7.

Table 5 Results of geometric attack according to the average distance (Avg) and scattering (Sct). 0° makes a view of the correct answer: 'D'. It shows that there is no correlation between the correct view directions and distances of dots

		0°	72°	154°	231°	288°
$\sigma = 0$ $\delta = 0.0003$						
	Avg	8.81	**8.85**	**8.55**	8.77	8.69
	Sct	6.18	**6.20**	**5.93**	6.00	5.98
$\sigma = 0.2$ $\delta = 0.0005$						
	Avg	9.98	**9.70**	**9.99**	9.71	9.83
	Sct	6.89	**6.69**	6.82	**6.66**	**6.69**
$\sigma = 0.4$ $\delta = 0.0005$						
	Avg	10.70	10.74	**10.25**	**10.93**	10.38
	Sct	7.35	7.24	**6.98**	**7.48**	7.09

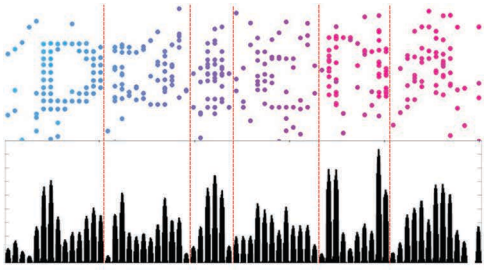
4.3.2 OCR attack

We conducted recognition tests using two well-known OCR engines: Google Tesseract [36] and ABBYY FineReader 14 [2]. We performed two types of attacks: entering whole words into the OCR engines for automated recognition and entering segmented individual letters into the OCR engines.

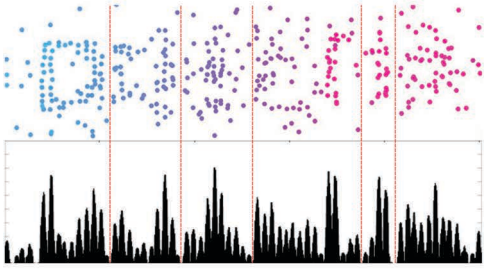
In both attacks, OCR engines could not completely recognize any of the words, even from the correct view images. Since the size of the spheres were small, they were not connected to each other. As a result, it was difficult to identify the letters by OCR, just as with the scatter-type CAPTCHA, which is resistant to OCR [6]. We shrank the image so that the shape of the spheres seemed to disappear and become downsampled. Then, the letters are partially recognized with a success rate of just 3.3%, which shows that DotCHA provides reasonable resistance to OCR attack.

4.4 User Study

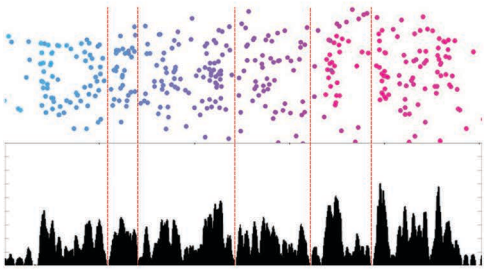
We ran a user study to compare the response time and success rate required to solve a 2D text-based CAPTCHA and our DotCHA according to the usability criteria [13]. A total of 50 participants recruited online took part in our web-based survey, and all the participants underwent eight unsupervised tests



(a) σ=0



(b) σ=0.25



(c) σ=0.5

Figure 7 Vertical histograms of DotCHA, when $\rho = 0.35$ and $\delta = 0.0015$. Noise remains in the image and it disturbs the segmentation by affecting the histogram. As the range of random offset increases, the segmentation tends to fail.

using their own devices: six DotCHA challenges (named T1 to T6) and two 2D text-based CAPTCHA challenges (T7 and T8).

Before the tests, we introduced the concept of DotCHA to the participants and explained how to identify the letters. To allow participants to become familiar to DotCHA, we showed an example of the correct answer to the participants without time limits. To ensure a fair time measurement, users were asked to load the model by clicking the button. Since there was no delay

in the model's loading, we recorded the time from loading to entering the answer.

All the DotCHA challenges had $n = 6$ letters and were randomly generated using the value of the parameters in Table 6. Six DotCHA challenges used the specified letters, but different models were generated according to the random values of the parameters whenever the user loaded the model into the browser. T7 and T8 were generated from reCAPTCHA with a single word, as shown in Figure 8. To familiarize the participants with the challenges, one practice DotCHA demo was shown to the participants at the beginning of the survey without revealing the correct answer. Through the practice demo, participants were taught how to rotate the DotCHA and how to enter the answer so that overhead due to inexperienced operation was not generated during the time measurements. The users were not told whether they had passed or failed each challenge.

Although participants sometimes gave partially correct answers, we only assigned credit when all letters were entered correctly. Table 7 shows the average time and standard deviation of success and failure cases and each test's success rate and Figure 9 plots the tests' success and failure response times. The graph indicates the domain of the response time according to the percentage of participants. For example, in the case of T2, 81.6% of participants successfully answered within 61.2 seconds and 50% of participants answered the correct answer within 31.3 seconds. Symbols are plotted on the graph to show the constant intervals (20%) in the participants' domain for each test.

While the overall success rate of DotCHA was lower than reCAPTCHA, some tests (T2, T4, and T6) showed a similar success rate as reCAPTCHA, which exceeded 80%. Some participants required a relatively long time to

Table 6 Values of parameters μ, ρ, σ, and δ for the survey

	T1	T2	T3	T4	T5	T6
μ	0.85	0.7	0.8	0.8	0.8	0.8
ρ	0.5	0.5	0.3	0.5	0.3	0.3
σ	0	0.2	0.2	0.3	0.3	0.3
δ	0.0005	0.0005	0.0005	0.0005	0.0005	0.001

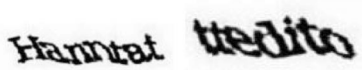


Figure 8 reCAPTCHAs for T7 and T8.

Table 7 Survey results for DotCHA and reCAPTCHA

		DotCHA						reCAPTCHA	
		T1	T2	T3	T4	T5	T6	T7	T8
Success Rate (%)		76.9	82.1	48.7	82.1	61.5	87.2	84.6	94.9
Success	Avg Time (s)	48.8	29.4	41.2	11.3	11.6	35.5	7.2	24.0
	Std	28.3	12.2	19.3	4.5	10.1	21.4	4.0	15.6
Failure	Avg Time (s)	34.3	26.2	28.6	8.3	6.6	19.2	5.7	18.9
	Std	24.2	8.7	14.8	4.5	2.2	16.8	2.3	6.3

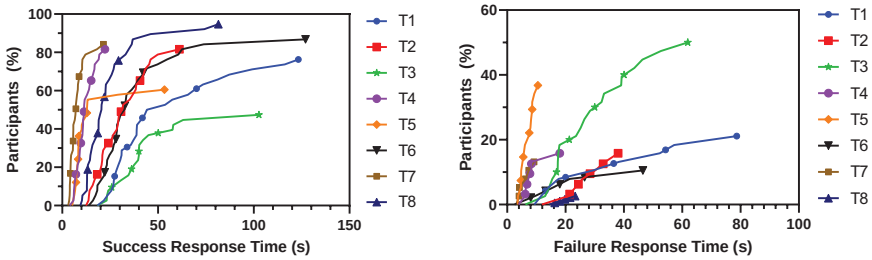


Figure 9 Success and failure response times.

solve T1 and T6. We suppose that this is because these participants were not yet familiar with DotCHA (T1), or when one of the most difficult combinations of parameters (high μ, σ, and δ and low ρ value) was used (T6). In the case of reCAPTCHA (T7 and T8), most users answered with a high success rate within a short response time but some participants spent more than a minute acquiring the correct answer. For T4 and T6, neither the success rate nor the average time for success differed significantly compared to reCAPTCHA.

The low success rate of T3 and T5 is due to the confusion of letters with similar forms, such as Y–V and D–O. We believe that this issue can be solved by using a font that has clear differences between the shapes of letters instead of the current Consolas font. Comparing T1, T2, and T3, it seems that the removal ratio affects the response time. However, participants gradually became accustomed to the large removal ratio and showed a stable response time from T4.

T4 and T5 compare the effect of the sphere radius and 80% of participants seem to have a similar response time. The difference setting between T3 and T5 is the sphere offset; surprisingly, T5 recorded faster success response times. We suppose that this is possible because the participants had become

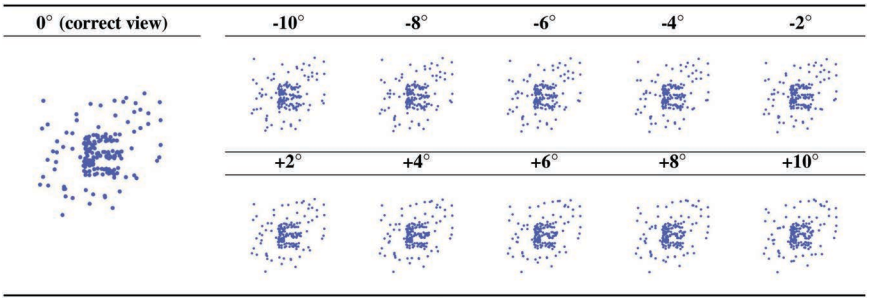
fully accustomed to the parameter's influence. T5 and T6 compare the effect of noise. When the noise increased, the success response time was slightly delayed but had a high success rate. We believe that this user study confirms the possibility that delicately tuning parameters can create a practical DotCHA that can balance security with usability.

5 Discussion

DotCHA is a newly introduced 3D text-based CAPTCHA to ensure both usability and security. Our work has several limitations and possibilities for improvements. The major issue is whether the letter can be read without finding the correct view perfectly. As Table 8 shows, this issue is influenced by compromises to usability. DotCHA is robust in security because it is expressed as a set of individual dots rather than connected contours. However, if small values of ρ and σ are employed for usability, there is a possibility that letters can be read without ensuring the exactly correct view. Machine learning-based OCRs, which were made for general purposes such as the Google Tesseract [36] and ABBYY FineReader 14 [2], have been found in previous experiments to be unable to read the current scatter type CAPTCHA. However, to prevent machine learning technology specially designed to attack DotCHA, additional improvements are required through geometric deformation utilizing the advantages of the 3D model.

The simplest solution is to add a 3D background that continues to move in the background behind the DotCHA as shown in Figure 10. The purpose of the 3D background is to make it difficult to separate itself from the foreground by moving regardless of the DotCHA's movements. Although a

Table 8 Results of view generation according to the change of angle near the correct angle (from $-10°$ to $10°$)



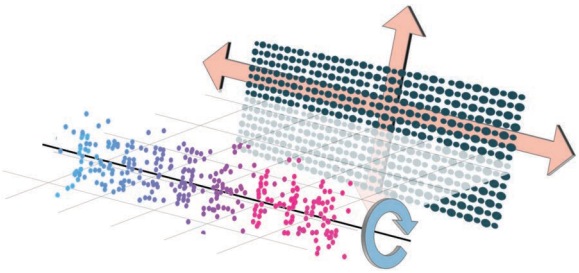


Figure 10 We can add a 3D background to disturb the machine in the separation of foreground (letters) background (noise). The background model has different transformations (red arrow) with letter model (blue arrow), such as scaling, translation, and rotation.

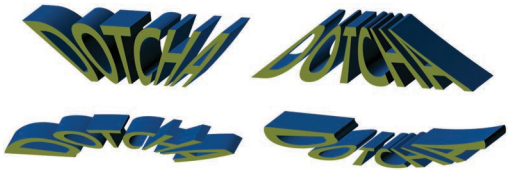
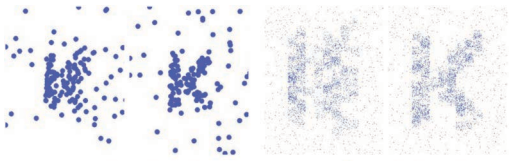


Figure 11 Adding 3D deformations in the extrusion stage to defend against segmentation and machine learning attack.

human can recognize the background from the foreground through motion parallax, machines generally judge captured images without motion parallax; therefore, the machine would be disturbed in the separation of the foreground and the background. However, with the development of machine learning technology, motion parallax will become vulnerable someday so defense through geometric deformation of the 3D model itself is required.

Deformation of the letters is a common technique used in 2D text-based CAPTCHA [3] and 3D text-based CAPTCHA [19, 24]. Deformation processes used in the existing 3D text-based CAPTCHA use simple extrusion and warping, and they cannot avoid the segmentation attack. More complex types of nonlinear deformations, such as bending, squashing, and stretching, can be used in the extrusion or twist stages of DotCHA generation, thereby making changes in the shape of letters according to the more diverse view angle. The nonlinear deformations also defend against machine learning attacks as the model changes into many unpredictable shapes, as shown in Figure 11. However, since usability also decreases due to the deformation, it is necessary to determine the appropriate degree of deformation through additional experiments.



(a) Sphere-based (b) Particle-based

Figure 12 Conversion of each dot into a set of small particles to reduce the placement tendency of spheres: (left) views from an arbitrary angle and (right) correct angle.

The placement tendency of spheres plays a major role in making letters recognizable in a view that is not necessarily the correct view. To reduce this tendency, converting each dot into a set of small particles is the best way, as shown in Figure 12. A set of particles increases the scatter ratio and these particles are expected to bring a big shape change in a small view difference.

DotCHA can be attacked not only by 2D image-based machine learning but also by 3D model-based machine learning. In the case of 2D learning, there is no other way but to bring disturbance to the learning dataset by adding random factors, such as backgrounds, deformation, and particles. Recently, 3D model-based learning has emerged, but it is still in its infancy stage as most focus has been placed on 3D model retrieval or reconstruction. 3D model-based learning should perform segmentation on every sphere of the given DotCHA model. Model segmentation is one of the most elaborate techniques in computer graphics, and sphere labeling to each letter will not be easy even if random factors are added to the model. However, machine learning has improved rapidly so we cannot be optimistic about defending against the new type of CAPTCHA. It is necessary to constantly study various attack and defense methods through variations of machine learning techniques for various purposes. There is no doubt that DotCHA's improvement and attacking technology will continue to constantly redevelop.

6 Conclusion

In this paper, we have proposed a new type of 3D text-based CAPTCHA, called DotCHA, which attempts to overcome the limitations of existing 2D and 3D approaches. It is a scatter-type CAPTCHA, which shows different letters according to the rotation angle, and the user should rotate the 3D model to identify the letters. We demonstrated that DotCHA is robust against several types of attacks.

There is a general consensus that it is hard to design a CAPTCHA that simultaneously combines good usability and security [30]. To improve the usability of DotCHA while preserving security, we combined the automated rotation and interactive systems. As we demonstrated, DotCHA defeats several types of attacks even when the correct view direction is given. In the same way as image-based CAPTCHA handling over images, the browser receives a 3D model, so security issues in communication are not problematic.

To improve the security, three additional strategies are possible: adding a background, using a set of small particles instead of a single sphere, and distorting the 3D model. Adding a complicated background protects the CAPTCHA against machines due to the difficulties in separating the foreground from the background to identify the letters. A set of small particles also increases the scatter ratio and enhances the defense against pixel-counting attack or edge detection attack. In addition, with the same principle as the 2D text-based CAPTCHA, distortion can be applied to the 3D letter model to make the DotCHA robust to segmentation attack.

Acknowledgement

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2019-0-01158, Development of a Framework for 3D Geometric Model Processing).

References

[1] Nucaptcha. http://www.nucaptcha.com. Accessed: 2019-12-10.

[2] Abbyy finereader 14. https://www.abbyy.com/en-apac/finereader/, 2014. Accessed: 2019-12-10.

[3] Luis von Ahn, Manuel Blum, Nicholas Hopper, John Langford, and Udi Manber. The captcha project. http://www.captcha.net/captchas/gimpy/, 2000. Accessed: 2019-12-10.

[4] Gerhard Bachfischer and Toni Robertson. From movable type to moving type-evolution in technological mediated typography. In *AUC Academic and Developers Conference*, 2005.

[5] Gerhard Bachfischer, Toni Robertson, and Agnieszka Zmijewska. A moving type framework. In *Proceedings of the 10th WSEAS international conference on Communications*, pages 607–612. World Scientific and Engineering Academy and Society (WSEAS), 2006.

[6] Henry S. Baird and Terry P. Riopka. Scattertype: a reading CAPTCHA resistant to segmentation attack. In *Document Recognition and Retrieval XII, San Jose, California, USA, January 16–20, 2005, Proceedings*, pages 197–207, 2005.

[7] Elie Bursztein, Jonathan Aigrain, Angelika Moscicki, and John C. Mitchell. The end is nigh: Generic solving of text-based captchas. In *8th USENIX Workshop on Offensive Technologies, WOOT '14, San Diego, CA, USA, August 19, 2014.*, 2014.

[8] John F. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986.

[9] Kumar Chellapilla and Patrice Y. Simard. Using machine learning to break visual human interaction proofs (hips). In *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13–18, 2004, Vancouver, British Columbia, Canada]*, pages 265–272, 2004.

[10] Monica Chew and Henry S. Baird. Baffletext: a human interactive proof. In *Document Recognition and Retrieval X, Santa Clara, California, USA, January 22–23, 2003, Proceedings*, pages 305–316, 2003.

[11] Sarika Choudhary, Ritika Saroha, Yatan Dahiya, and Sachin Choudhary. Understanding captcha: Text and audio based captcha with its applications. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(6), 2013.

[12] Yang-Wai Chow and Willy Susilo. Anicap: An animated 3D CAPTCHA scheme based on motion parallax. In *Cryptology and Network Security – 10th International Conference, CANS 2011, Sanya, China, December 10–12, 2011. Proceedings*, pages 255–271, 2011.

[13] Mauro Conti, Claudio Guarisco, and Riccardo Spolaor. Captchastar! A novel CAPTCHA based on interactive shape discovery. In *Applied Cryptography and Network Security – 14th International Conference, ACNS 2016, Guildford, UK, June 19–22, 2016. Proceedings*, pages 611–628, 2016.

[14] Rony Ferzli, Rida A. Bazzi, and Lina J. Karam. A captcha based on the human visual systems masking characteristics. In *Proceedings of the 2006 IEEE International Conference on Multimedia and Expo, ICME 2006, July 9–12 2006, Toronto, Ontario, Canada*, pages 517–520, 2006.

[15] Haichang Gao, Dan Yao, Honggang Liu, Xiyang Liu, and Liming Wang. A novel image based CAPTCHA using jigsaw puzzle. In *13th IEEE International Conference on Computational Science and Engineering,*

CSE 2010, Hong Kong, China, December 11–13, 2010, pages 351–356, 2010.

[16] Philippe Golle. Machine learning attacks against the asirra CAPTCHA. In *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008*, pages 535–542, 2008.

[17] Rich Gossweiler, Maryam Kamvar, and Shumeet Baluja. What's up captcha?: a CAPTCHA based on image orientation. In *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20–24, 2009*, pages 841–850, 2009.

[18] Yuki Ikeya, Masahiro Fujita, Junya Kani, Yuta Yoneyama, and Masakatsu Nishigaki. An image-based CAPTCHA using sophisticated mental rotation. In *Human Aspects of Information Security, Privacy, and Trust – Second International Conference, HAS 2014, Held as Part of HCI International 2014, Heraklion, Crete, Greece, June 22–27, 2014. Proceedings*, pages 57–68, 2014.

[19] Montree Imsamai and Suphakant Phimoltares. 3D captcha: A next generation of the captcha. In *Information Science and Applications (ICISA), 2010 International Conference on*, pages 1–8. IEEE, 2010.

[20] Ibrahim Furkan Ince, Yucel Batu Salman, Mustafa Eren Yildirim, and Tae-Cheon Yang. Execution time prediction for 3D interactive captcha by keystroke level model. In *Computer Sciences and Convergence Information Technology, 2009. ICCIT '09. Fourth International Conference on*, pages 1057–1061. IEEE, 2009.

[21] Suzi Kim and Sunghee Choi. Automatic generation of 3D typography. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference, SIGGRAPH '16, Anaheim, CA, USA, July 24–28, 2016, Posters Proceedings*, pages 21:1–21:2, 2016.

[22] Suzi Kim and Sunghee Choi. Dotcha: A 3D text-based scatter-type CAPTCHA. In *Web Engineering – 19th International Conference, ICWE 2019, Daejeon, South Korea, June 11–14, 2019, Proceedings*, pages 238–252, 2019.

[23] Punam Kumari and Mansi Kapoor. Effect of random guessing attack on image based captchas: Analysis and survey. *International Journal of Innovations & Advancement in Computer Science*, 4, 2015.

[24] Cristina Romero Macias and Ebroul Izquierdo. Visual word-based captcha using 3D characters. pages 1–5, 2009.

[25] Goran Martinovic, Andrew Attard, and Zdravko Krpic. Proposing a new type of CAPTCHA: character collage. In *MIPRO, 2011 Proceedings of*

the 34th International Convention, Opatija, Croatia, 23–27 May, 2011, pages 1447–1451, 2011.

[26] Maryam Mehrnejad, Abbas Ghaemi Bafghi, Ahad Harati, and Ehsan Toreini. Seimcha: a new semantic image captcha using geometric transformations. *The ISC International Journal of Information Security*, 4(1):63–76, 2012.

[27] J. Abbott Miller. *Dimensional Typography:: Words in Space: Kiosk Report# 1*. Number 1. Princeton Architectural Press, 1996.

[28] Greg Mori and Jitendra Malik. Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003), 16–22 June 2003, Madison, WI, USA*, pages 134–144, 2003.

[29] S. A. N. Samarasinghe N. R. W. W. M. R. D. Wickramasingha, H. B. R. A. K. R. A. M. Keerawella and R. G. Ragel. Rotatecaptcha a novel interactive captcha design targeting mobile devices. In *Industrial and Information Systems (ICIIS), 2015 IEEE 10th International Conference on*, pages 49–54. IEEE, 2015.

[30] Margarita Osadchy, Julio Hernandez-Castro, Stuart J. Gibson, Orr Dunkelman, and Daniel Pérez-Cabo. No bot expects the deepcaptcha! introducing immutable adversarial examples with applications to CAPTCHA. *IACR Cryptology ePrint Archive*, 2016:336, 2016.

[31] Ayane Sano, Masahiro Fujita, and Masakatsu Nishigaki. Directcha: A proposal of spatiometric mental rotation CAPTCHA. In *14th Annual Conference on Privacy, Security and Trust, PST 2016, Auckland, New Zealand, December 12–14, 2016*, pages 585–592, 2016.

[32] Roger N. Shepard and Jacqueline Metzler. Mental rotation of three-dimensional objects. *Science*, 171(3972):701–703, 1971.

[33] Shenna Shepard and Douglas Metzler. Mental rotation: effects of dimensionality of objects and type of task. *Journal of Experimental Psychology: Human Perception and Performance*, 14(1):3, 1988.

[34] Patrice Y. Simard, David Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *7th International Conference on Document Analysis and Recognition (ICDAR 2003), 2-Volume Set, 3–6 August 2003, Edinburgh, Scotland, UK*, pages 958–962, 2003.

[35] Suphannee Sivakorn, Iasonas Polakis, and Angelos D. Keromytis. I am robot: (deep) learning to break semantic image captchas. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21–24, 2016*, pages 388–403, 2016.

[36] R. Smith. An overview of the tesseract OCR engine. In *9th International Conference on Document Analysis and Recognition (ICDAR 2007), 23–26 September, Curitiba, Paraná, Brazil*, pages 629–633, 2007.

[37] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. CAPTCHA: using hard AI problems for security. In *Advances in Cryptology – EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003, Proceedings*, pages 294–311, 2003.

[38] C. Winter-Hjelm, M. H. Kleming, and R. H. Bakken. An interactive 3D captcha with semantic information. In *Proc. Norwegian Artificial Intelligence Symp.*, pages 157–160, 2009.

[39] Jeff Yan and Ahmad Salah El Ahmad. Breaking visual captchas with naive pattern recognition algorithms. In *23rd Annual Computer Security Applications Conference (ACSAC 2007), December 10–14, 2007, Miami Beach, Florida, USA*, pages 279–291, 2007.

[40] Jeff Yan and Ahmad Salah El Ahmad. Usability of captchas or usability issues in CAPTCHA design. In *Proceedings of the 4th Symposium on Usable Privacy and Security, SOUPS 2008, Pittsburgh, Pennsylvania, USA, July 23–25, 2008*, pages 44–52, 2008.

[41] Jeff Yan and Ahmad Salah El Ahmad. CAPTCHA security: A case study. *IEEE Security & Privacy*, 7(4):22–28, 2009.

[42] Qi Ye, Youbin Chen, and Bin Zhu. The robustness of a new 3D CAPTCHA. In *11th IAPR International Workshop on Document Analysis Systems, DAS 2014, Tours, France, April 7–10, 2014*, pages 319–323, 2014.

Biographies

Suzi Kim received the B.S. and M.S. degree in Computer Science from Korea Advanced Institute of Science and Technology (KAIST) in 2012 and

2016. She worked for the Daewoo Shipbuilding Marine Engineering (DSME) from Jan. 2012 to Mar. 2013 and Prezi from Apr. 2013 to Aug. 2014. She is currently working toward the Ph.D. degree in School of Computing at KAIST. Her research interests include computer graphics such as procedural and inverse-procedural modeling and geometry processing.

Sunghee Choi received the B.S. degree in computer engineering from Seoul National University in 1995, and the M.S. and Ph.D. degrees in computer science from the University of Texas at Austin, in 1997 and 2003, respectively. She has been working as a professor in School of Computing at Korea Advanced Institute of Science and Technology (KAIST) Daejeon, Korea since 2004. Her research interests include computational geometry, computer graphics and geometric problems in wireless sensor networks.