PiSHi: Click the Images and I Tell if You Are a Human

Maryam Mehrnezhad *[†], Abbas Ghaemi Bafghi[††], Ahad Harati[††] and Ehsan Toreini[†]

[†]School of Computing Science, Newcastle University, Newcastle upon Tyne, UK
[††]Computer Department, Ferdowsi University of Mashhad, Mashhad, Iran

Abstract

This paper introduces PiSHi (pictorial intelligent system for human identification), an image-based captcha which uses three human cognitive abilities to distinguish humans from machines. The first is the human ability to easily recognize the image's upright orientation. The second is the human brain's ability in recognising a picture's content when it is only partially visible. And the third is the human ability in unconscious decision making when encountering pictorial challenges. This work models such complicated human patterns in problem solving for the first time. In order to extract these behavioural patterns and save them in a pattern database, we have implemented our own captcha and performed a series of experiments. PiSHi's interface presents the user with a set of distorted pictures, and asks her to click on the upright orientation of all the pictures in any preferred order. Next, it captures the user's interaction patterns, compares them with the ones saved in the pattern database, and grants her a corresponding credit. Based on this credit, the user either passes or fails the test, and participates in updating the picture database. Our experiments indicate that human users can solve our proposed captcha effectively- with an accuracy of 99.44%. Besides, our proposed system is secure against several types of attacks including random guessing and reverse image search engines. The results offer the possibility of utilizing the identified human behavioural models in practical captchas.

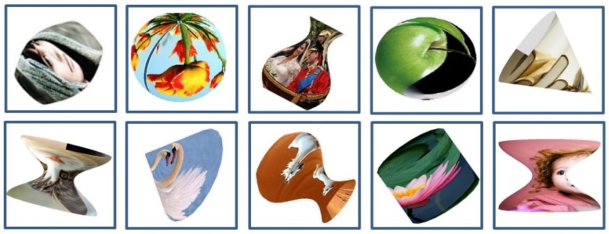


Figure 1: PiSHi asks the user to click on the upright orientation of 10 distorted pictures. Eight of these pictures are for evaluating the user, and have different hardness rates to be clicked correctly. The other two pictures are under observation of multiple users to be chosen in order to update the captcha database.

Keywords. Image CAPTCHA, User interaction patterns, Decision making, Security, Usability.

1 Introduction

Free web services are increasingly used by many users everyday. This attracts more attention to the problem of misuse through automated soft bots, and makes it essential to distinguish between a human user and a machine. Completely Automated Public Turing Test to Tell Computer and Humans Apart (CAPTCHA[1]) offers a way to make such a distinction. Captcha is an extended version of the Turing

*Email: m.mehrnezhad@ncl.ac.uk
Telephone: +44 191 20 85153, Fax: +44 191 20 88232

[1] We use captcha instead of its original form (CAPTCHA) for convenience.

Test, aka a secondary authentication mechanism.

The first captcha was a text-based one, designed for the Yahoo website [19] in 2000. A text-based captcha asks the user to distinguish, and type a set of characters presented in the context of a noisy image. Text-based captchas are popular since they are easy to design and implement. However, the distorted noisy images are sometimes too complex for human users [4]. Moreover, it has been shown that even the most difficult variants of distorted text can be solved with 99.8% accuracy by advanced methods [22].

Alternatively, other multimedia elements such as image, voice, and video have been proposed in the literature to be utilised in captcha solutions [2, 1, 17, 15, 14, 13, 10, 12]. Multimedia captchas are simple, attractive, and fun for human users. Besides, it is harder for machines to distinguish such elements compared to text. In particular, image-based captchas (image captchas) have been widely proposed by different researchers, and even commercialised by some companies. For example, Google upgraded its text-based reCAPTCHA ver 1.0 to an image-based one, reCAPTCHA ver 2.0 with a free API, in 2014 [22].

1.1 Image-based Captchas

There are many different image captchas, all of which present the users with a set of pictures from the captcha database. Many of the older image captchas were based on limited image databases which store a few number of pictures along with their labels (tags). This approach however, suffers from a few security issues. For example, the limited size of the database enables the attacker to learn the whole database after a while, and then access the system. Besides in order to grantee the security of such a design, the picture databases and the tags need to be kept private which is against the Kerckhoffs's principle, which states that a security system should remain secure even if all algorithms and databases are public [12].

Some other solutions suggest the inclusion of *semantics* in image captchas. Semantics are the human's abilities in recognising some particular subjects in the pictures. Examples include face recognition [18], and animal recognition [1]. Most of these solutions, however, have the same database problem. In addition, machine learning algorithms are pretty successful in the recognition of specific items such as human faces [16] or animal species [9].

More recent and advanced captchas suggest to the use of more complicated semantics such as image orientation. Almost all people learn basic concepts such as up and down, close and far, beauty and ugliness, etc. at a very young age. However, such concepts cannot be easily learned by automated programs. To the best of our knowledge, image orientation (to be more specifically, the top part of an image) is the only concept that has been applied in a few captchas [10, 17, 14, 13]. The image orientation concept is label-free i.e. the captcha does not need to save any matching labels for the pictures as the answers in the database. The reason is that almost all images are normally in the correct orientation. This unique property improves the captcha security significantly.

In this work, we utilize the concept of picture orientation differently. Instead of relying on the user's correct recognition of the top part of the picture, we focus on the interaction model that the user unconsciously follows to complete the challenge, as explained in the next subsections in detail.

1.2 Motivation

What discriminates a captcha from other security systems is the human's strong role in the process. In fact, captcha systems are supposed to be built based on human capabilities. Most image captchas that have been developed to date are based on human recognition abilities (such as face, animal, top part of an image). We believe that there are other human abilities which have not been explored by the researchers in the field. For instance, one of the most natural human properties is being error-proneness. It's said that 'to err is human'. As Anderson discusses it in his *Security Engineering* book: "error research confirms this: the predictable varieties of human error are rooted in the very nature of cognition" [24]. This property has not been noticed directly in captcha designs to date. Most captchas do not accept human errors in their systems i.e. a complete correct answer is required. However, a few have applied different policies such as accepting partial correct answers [2], [5], [1], [13]. In PiSHi, we neither rely on complete, nor partial correct answer strategies. We propose a new method which grants users associated credits based on their natural reactions - including errors- when solving a pictorial problem.

People have unique capabilities in problem solving. They have specific behavioural models for perceiving and answering image captchas. In this paper, we ask our users to solve an image captcha, and we analyse the user's entire engagement with the system. The problem presented to the users is the recognition of the upside part of multiple pictures, which are partially presented in the page. Hence, we also include another human brain's ability which is its strength in rebuilding the whole content of the picture by only seeing some parts of it. For example if a user sees an ear of an elephant, he can build the whole picture in his mind, and recognise the subject. For partial presentation of pictures, a captcha system can simply cut some parts of the picture randomly and present it to the user. However, a more usable and secure approach is to use some advanced transformations. Therefore, we distort original pictures by applying geometric transformations (see Fig. 3), as suggested in Multiple Seimcha [13].

The hypothesis of this paper is that when users are asked to click on the upside part of a set of distorted images (like in Fig. 1), they tend to click on the easier ones first and with higher accuracy compared to the harder ones. By easier pictures, we mean those that whose contents are more recognisable by the users. These sorts of patterned human interactions with pictorial systems have not been applied in captchas so far, and are the focus of this paper. Hence, the research questions are: a) to test if it is possible to model human users' behaviour in terms of interaction patterns with our image captcha, and b) to find out if it is feasible to use the extracted patterns as the core of an image captcha In order to observe our research questions we designed, implemented, and evaluated a prototype of the proposed system. The main contributions of this paper are:

1. We propose "PiSHi" as a new solution to distinguish between human and bots. PiSHi grants a user a credit based on his interaction with the system. This credit determines whether he passes or fails, and what impact he has on updating the database. Our captcha is the first that includes human behavioural interaction patterns in its design.

2. We present a proof-of-concept implementation of PiSHi. We conducted user studies to evaluate the usability of our system. Experiments confirmed that the extracted human interaction patterns effectively distinguish between the human users and the bots.

3. We present a security analysis of our proposed captcha including estimating the chance of random guessing attacks, and the results of reverse image search engines. The results show that the proposed system is effectively secure against such attacks.

Furthermore, we propose complexity metrics to be considered as a new category of captcha metrics. We evaluate PiSHi by presenting different metrics, and compare them with existing research.

1.3 Captcha Metrics

The following properties are commonly suggested in the literature for evaluating a captcha; automation, open algorithms and databases, usability, and security [12]. It is assumed that the first two properties are provided by most strong and commercial captchas. Based on this assumption, different works evaluate their proposed captchas by presenting *usability* studies as well as *security* analysis. However, there is no standard framework for captcha evaluation, and different approaches have been adapted in the literature. On the other hand, captchas are web-based systems, and different users are frequently connecting to the servers simultaneously. Hence, we believe that their scalability should be evaluated too. Accordingly, we suggest to add *complexity* factors as a new category of captcha metrics. Following this addition, captcha metrics can be categorized into three different groups; usability, security and complexity.

Usability. Usability pertains to those features which deal with user aspects of the system such as success rate, response time, and response mechanism. *Success rate* or response rate estimates the ratio of the cases that an average human user is successful in solving the captcha challenges. *Response time* is the time that an average human user spends on a captcha sample, regardless of the result, which could be either pass or fail. And *response mechanism* refers to the action that the user is required to take in order to interact with the captcha, e.g. clicking, typing, etc.

Security. Security metrics relate to possible attacks on captcha systems i.e. the probability of an automatic program succeeding in passing the challenges. In general, we can categorise the security attacks on image captchas into three different groups; random guessing, direct matching, and machine learning attacks. *Random guessing* is the situation when an attacker randomly passes the captcha challenges without any effort e.g., when a bot randomly clicks on the pictures correctly. *Direct matching* refers to the condition that an attacker solves the captcha by searching for the answer in a lookup table. It is possible to construct the lookup table by stealing the image database, or by mechanical Turk attack [17]. And finally, *machine learning attacks* try to learn the captcha images in the way that a human user does. These attacks generally use different types of advanced machine learning algorithms to break the captcha [20], [9]. It is not straightforward to analyse the strength of a picture captcha against machine learning attacks since there are different techniques for image learning, and also they should be customised depending on each captcha.

Alternative to customised machine learning attacks, many works present the results of *reverse image search engines* in matching their suggested distorted pictures with original ones available in the web [25]. There are many different reverse image search engines including Yanex[2], Karma Decay[3], Rev IMG[4], ImgSeek[5], Google image search[6], and Tineye[7]. Unfortunately, most of these search engines do not disclose their exact technical details such as feature selections, or learning algorithms. However, we roughly know that all reverse image search engines use advanced Content Based Image Retrieval (CBIR) methods as a part of their systems [23]. They produce image fingerprints and use different machine learning algorithms in order to find the possible matches in their databases. Hence, attacks based on reverse image search engines could be considered in the category of machine learning attacks.

Complexity. As mentioned earlier, since captchas are web applications, they are frequently used by different users simultaneously. Hence for a practical captcha, a few scalability constraints should be considered in the design. We suggest three important factors to be evaluated for a practical captcha; a) server complexity, b) client complexity, and c) communication complexity. *Server complexity* presents the complexity of the algorithms in the server side including producing a challenge and evaluating a response. *Client complexity* is based on the necessity of using any particular software or settings in the client side. And *communication complexity* is the amount of the data being communicated between the client and the server. These issues have been considered in few related works such as *Sketcha* [17], but have not been mentioned as captcha metrics. Thus, they are proposed here as a new category of captcha metrics and are referred to as complexity metrics.

2 PiSHi overview and initial configuration

Here, we review our proposed system, and we present the preprocessing steps and its initial settings.

2.1 Overview

We propose PiSHi (pictorial intelligent system for human identification) as a new image captcha. PiSHi

[2] www.yandex.com/images/

[3] karmadecay.com/

[4] www.revimg.com/

[5] sourceforge.net/projects/imgseek/

[6] www.google.com/imghp

[7] tineye.com/

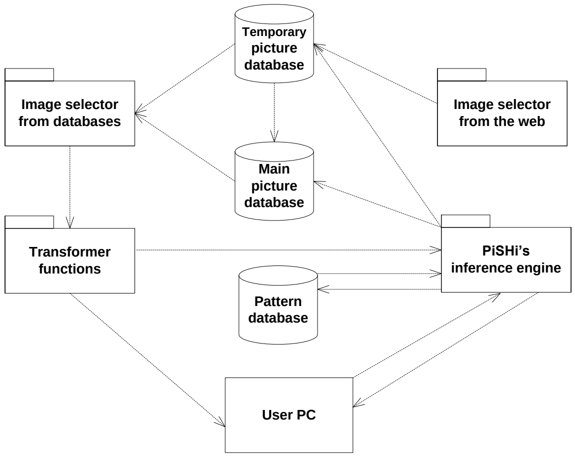


Figure 2: PiSHi's component diagram

mainly relies on modelling human abilities in decision making when answering image-based challenges. Human users have complicated patterns when they interact with computerized pictorial systems. However, before any pattern extraction and building the corresponding models, it is necessary to find appropriate challenges for the users to solve. In PiSHi, we utilize two challenges; a) the identification of the upright orientation of an image, and b) the recognition of the image content by seeing it partially. Both challenges are easy for human users, and hard for automated programs to solve. In the preprocessing phase, we use *geometric transformations* and random rotations to partially present the pictures to the users to click on the upright part of it. We also estimate the difficulty of user identification after distorting the original pictures and define it as *hardness rate*.

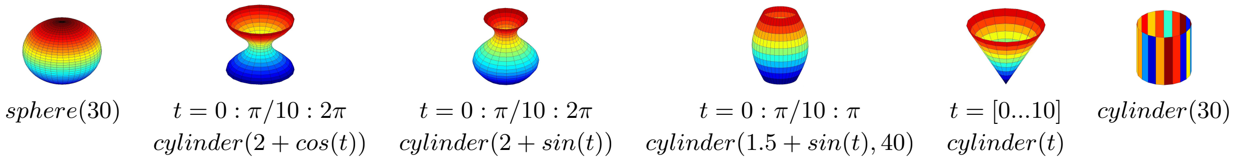
Combining multiple ideas as mentioned above gives us the opportunity to model the behaviour of human users. PiSHi distinguishes human users according to their particular patterned decisions which they make without even being aware of them. These decisions could be either right or wrong in terms of clicking the upright part of the pictures correctly. We extract our patterns in such a way that human users would benefit not only from their correct answers, but also from their natural wrong ones. We consider these decisions as interaction patterns, and extract and evaluate them by using experimental datasets. We have performed our experiments by asking volunteer users to participate in the study. We logged 155 different records for the training dataset, and 180 new records for the test dataset.

We implemented a prototype of PiSHi as a proof of concept and evaluated it with real world data from human users. This prototype was implemented as a web application using *ASP.Net*. Geometric transformations were implemented in *Matlab* and all images were saved in a file locally. Other information about images was saved in an *Access* database along with the details of the users' interactions with the system. All experiments were done locally on a laptop.

PiSHi shows 10 distorted images to the users; eight images for evaluating users, and two images for updating the database (Fig. 1). These pictures are randomly presented in the page. The user should click on the top of all images to complete the challenge. The system then calculates the user's credit by comparing this behaviour with the extracted patterns. Based on this credit, PiSHi either passes or fails the user, and also updates the picture database . Fig 2 shows the system's component diagram. We will present different aspects of PiSHi by building different parts of this component diagram gradually.

2.2 Original and distorted pictures

We downloaded 30 different pictures from the internet with different subjects *(Apple, Baby, Badminton ball, Bike, Birds, Books, Building, Camel, Car, Cat, Dolphins, Elephant, Flower, Fish, Frog, Hamster, Horses, Lighthouse, Oranges, Panda, Plane, Penguins, Peacock, Rose, Swans, Spiderman, Ship, Trees, Tom & Jerry, and Zebra)*, as shown in Fig. 5. Then, we applied a few distortions to obtain the output samples. The distortions include 2D rotations, geometric transformations, and 3D rotations. For the 2D rotations, we randomly chose an angle (0, 360) and rotated the pictures. In order to avoid white margins in the rotated pictures, we zoomed and cut some parts of them by using *ROIrotate* function in Matlab [3]. Next, we applied geometric transforma-



$sphere(30)$ $t = 0 : \pi/10 : 2\pi$ $cylinder(2 + cos(t))$ $t = 0 : \pi/10 : 2\pi$ $cylinder(2 + sin(t))$ $t = 0 : \pi/10 : \pi$ $cylinder(1.5 + sin(t), 40)$ $t = [0...10]$ $cylinder(t)$ $cylinder(30)$

Figure 3: Geometric functions and their corresponding 3D objects

tions and generated six 3D objects of each original picture. We used a few simple Matlab functions such as *sphere* and *cylinder* with different settings in order to produce the 3D objects, as demonstrated in Fig. 3. In the final step, we randomly chose an angle in 3D space to capture a 2D picture from each 3D object using *Azimuth* and *Elevation* settings in Matlab. Elevation was set in the range of [-50°,50°] to avoid producing unusable picture. We repeated this process four times and ended up with 24 different distorted pictures for each original picture and saved them into a file.

In order to study the effect of the suggested transformations, we asked 24 volunteer users (male and female, 18 to 40 years old, university and non-university people) to identify the *subject* of the distorted pictures from a list of 30 labels (Apple, Baby, Badminton ball, Bike, etc.). We randomly chose a distorted picture and showed it to the user in a web page along with a list of labels, and repeated it for 60 rounds. We programmed the system in a way that it showed all distorted versions distributively to different users; hence each 3D model was seen 240 times in total. In addition, 48 different distorted images were shown to the users per each original picture. Our users could identify the content of the pictures in 87% of the cases in about 7.5 seconds on average. We studied the pictures that the users failed to recognise, and found they were randomly distributed between the original images and the selected 3D models. These results show that the selected transformations are effective and usable for the users.

2.3 Hardness rate

In this phase, we studied the effect of the suggested transformations on the identification of the top part of the picture. Following *Multiple SEIMCHA*'s approach [13], a key image -coloured black in the top one third- was transformed with the original image simultaneously, as shown in Fig. 4. This key image was in charge of retaining the top of the picture, as the right answer area, in the transformed version. We asked 20 volunteer users (male and female, 18 to 30 years old, university students and staff) to click on the top part of the pictures to find the associated average identification rates. We presented them with a simple web application showing a single distorted image in the page in each step. Then, we asked them to click next to see another picture. We showed 60 different distorted images (two distorted pictures per each original one, evenly chosen from our set of 3D objects) to all users. Therefore, 40 distorted images were presented to the users per each original picture. We calculated the identification rates of the original pictures based on the average identification rates of their corresponding distorted versions. We used these average identification rates of the original pictures in order to define the hardness rates.

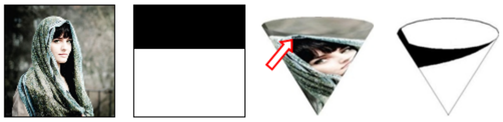


Figure 4: Examples of the original and key pictures, and their distorted versions

Following the same rules in [13], if the identification rate was less than 60%, the original image was rejected. Otherwise, we assigned the same identification rate as the picture's hardness rate (HR). Next, we put them in four different HR categories; *Simple* $(90\% - 100\%)$, *Medium* $(80\% - 90\%)$, *Hard* $(70\% - 80\%)$, and *Very Hard* $(60\% - 70\%)$. *S, M, H,* and *V* will represent these four groups, respectively,

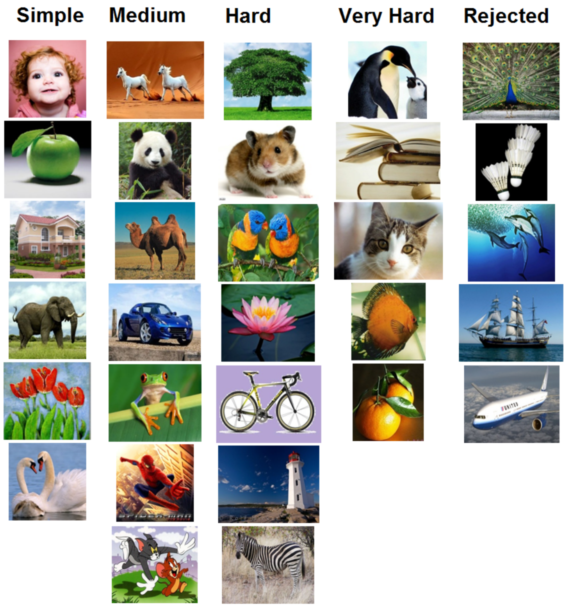


Figure 5: Original pictures categorised according to their hardness rates

in the rest of this paper. Fig. 5 shows the original pictures categorized according to their HRs. As it can be seen among the 30 initial pictures, five pictures were rejected due to their low recognition rates, and the rest are roughly distributed among all different difficulty groups. After analysing the rejected pictures and asking for users feedback, we found that these pictures had multiple top areas distributed in different parts of the picture. For example in one of the rejected pictures, there are four different badminton balls and their top parts are started from about the middle part of the picture to the very upper part of it. Hence, users had difficulties in recognising the upside orientation correctly.

To this end, some parts of PiSHi's component diagram have been built in the preprocessing phase. *Main image database* is the primary database which includes the original versions of the images along with their HRs. This database gets updated with new images, as we will present in Section 4. The *Image selector* algorithm picks pictures from each category of difficulty *(S, M, H and V)* from the main database to be distorted and shown to the user. This algorithm randomly includes non-redundant and equal number of pictures from each category (two of each category). *Transformer functions* include rotation algorithms and geometric transformations with multiple runtime random values for different variables. Now we are able to present the preprocessed pictures to the users and log their interaction with the system. Based on these interactions, we build the corresponding patterns, and use them in PiSHi's inference engine as explained in the next section.

3 Patterns and inference engine

In this section, first we present the data collection process for our training set, next we describe our suggested patterns, and finally we present our proposed inference engine to be used in PiSHi.

3.1 Data collection for training

We performed an experiment in order to collect experimental data for our training dataset. This dataset was used in order to extract users behaviour models, and to build the *pattern database* component. We asked 31 different users (male and female, 22 to 30 year old, university student and staff) to take part in this experiment. We presented each user with a webpage including eight distorted pictures from different categories of difficulties. We asked each user to click on the top part of each of the pictures in any order that they preferred. We repeated the experiment five times for all users, hence we had 155 different records at the end. Users interactions with the system were logged for further analysis.

Users were presented with the following instruction at the top of the page: *"Please click on the top part of the subject of each picture"*. A video guide was also prepared as a help document which users could click through. We also included an extra round for each user as a practice phase and did not log that. In order to improve the study, some of the users were asked to say aloud what they thought of the interaction with the system, or to answer the test by talking to a friend (pair work) instead of working individually

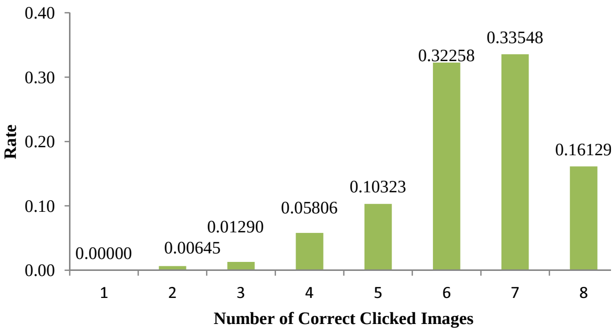


Figure 6: P_{No}: Success rate based on the number of correct answers

[7]. Almost all users found it easy to understand the system on their first try. There was however one user, who had a previous mental model of Sketcha [17], and he first thought he needed to click the images to rotate them.

3.2 Extracting interaction patterns

We studied the training dataset and extracted five different patterns based on the possibility of the occurrence of specific sequences. We analysed and modelled some particular patterns in the form of probability distribution functions. We extracted the five following patterns from our train set, and used them in PiSHi; success rate based on the number of correct answers, success rate based on hardness rate (HR), click order based on HR, success rate of each click based on HR, and response time. In the upcoming subsections we explain each pattern by including figures and examples. Note that some of the suggested patterns are complicated, and take time to follow.

3.2.1 P_{No}: Success rate based on the number of correct answers

One of the most important factors to decide to pass or fail the user is the number of the correct clicked images. Since a PiSHi's sample challenge includes images of different difficulty levels, it is expected that an average user clicks on some of the pictures wrongly. In other words, we do not expect the users to click all images correctly, instead we will give them credit if they err according to the extracted patterns. Fig. 6 shows the probability distribution of the number of users based on the correct clicked images.

We define P_{No} as the probability distribution function of this pattern. As can be seen in Fig. 6, if a user clicks of seven images correctly, she gets $P_{No}(7) = 0.335$ as her credit, which is the highest credit in this pattern. However, it is not fair to a user who has clicked eight correct images to gain less. Hence, we slightly modify this function to assign $P_{No}(7)$ to eight correct images too.

3.2.2 P_{HR-No}: Success rate based on HR

The second proposed pattern is the success rate based on hardness rate. The aim here is to discover how many of S, M, H, and V images are correctly identifiable by human users. We already know that simpler images will be answered more correctly in comparison to hard ones (that is where the definition of hardness rate comes from). Here, we want to model users' behaviour when they are presented with a set of pictures from different categories of difficulties.

Fig. 7 shows five different categories of possible sequences in this pattern. The labels on the x axis start with the number of right answers and continue with the type of wrongly clicked images. For example, the labels which start with 7 present the condition that seven images are correctly clicked. The incorrectly clicked image could be either S, M, H, or V, represented as 7-S, 7-M, 7-H, and 7-V, respectively. This pattern could be considered as a conditional probability function based on the first pattern (P1).

Categories of 7 and 6 present all their possible cases, while the others only show the more probable ones. The cases with the probability of zero in the experimental dataset are presented by the postfix of others in this figure. Categories 8, 3, 2 and 1 are not shown in Fig 7 since the first has the probability of 1, and the others rarely occurred in our experiment. If the cases that have not happened in the training experiment happen in the test experiment, PiSHi will grant them the minimum non-zero

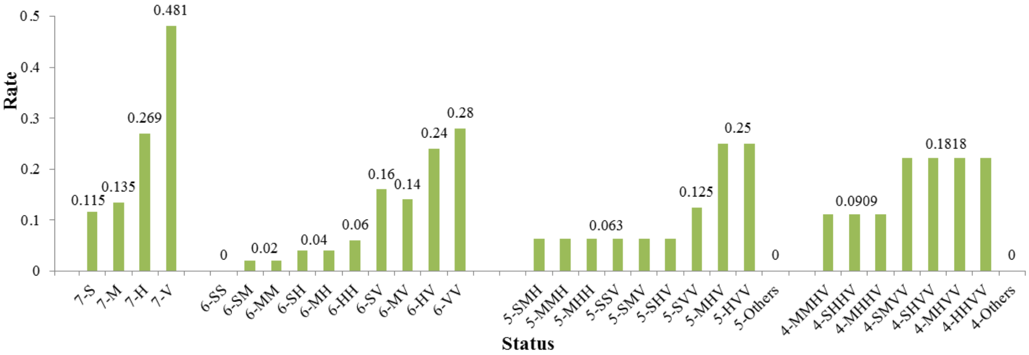


Figure 7: P_{HR-No}: Success rate based on HR

credit of this pattern. Let P_{HR-No} denote the probability distribution function of this pattern. As an example, if a user clicks 6 pictures correctly and two V images wrongly, PiSHi grants her the credit of $P_{HR-No}(6-VV) = 0.28$ in this pattern.

3.2.3 $P_{Cli-Order}$: Click order based on HR

The third suggested pattern presents the users' preferable orders of their clicks based on pictures HRs, regardless of the answer -correct or wrong. The idea here is to find out if the users select simpler images earlier in comparison to harder ones. For example, a possible click order is $SMHVSMHV$, i.e. the user chooses to click on a simple picture first (either correct or wrong), next he clicks on a medium picture, and then a hard one, and so on.

There are different possible click orders, some of which are very rare and did not appear in our training dataset. Therefore, we narrow them down to a subset of the more common ones by considering the first four clicked images, and disregarding the rest. We also do not consider in what order each picture has been opted by the user in these four clicks. In other words, we only observe if some particular click sequences has been seen in the first four clicks or not. Based on this suggestion, the total possible click orders are reduced to 19 as presented below:

- If there are no pair images from the same category of HR, there would be only one case (SMHV). As mentioned, it is not important if the user clicks the S image first, second, third, or last. Thus, SMHV is the representative of 23 other sequences too (substitutions of S, M, H and V inside the SMHV trace).

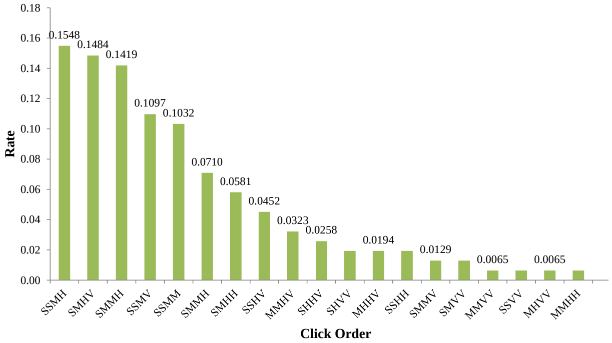


Figure 8: $P_{Cli-Order}$: Click order based on HR

- If there is only one pair from the same category of HR, there would be $4 \times 3 = 12$ cases (SSMH, SSMV, SSHV, MMSH, MMSV, MMHV, HHSM, HHSV, HHMV, VVSM, VVSH, and VVMH)
- And, if there are two pairs of the same category of HR, there would be six cases (SSMM, SSHH, SSVV, MMHH, MMVV, and HHVV)

Fig. 8 shows the probability of all these cases in our experimental dataset in descending order. The

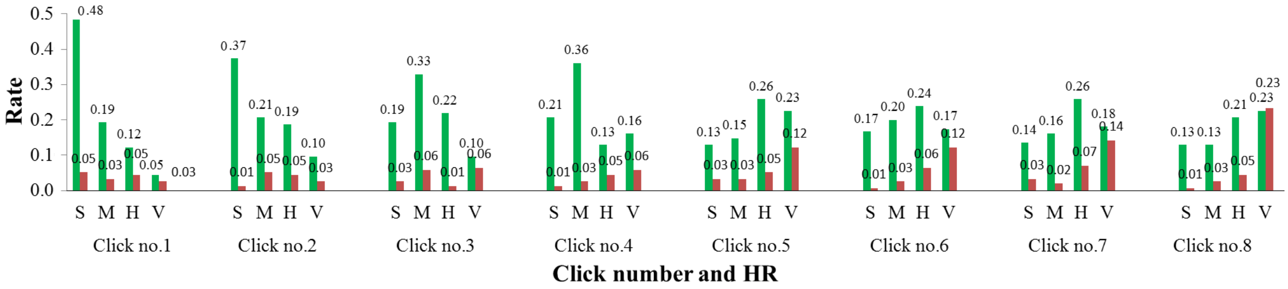


Figure 9: $P_{F-Pos-HR}$: Success rate of each click based on HR

x axis labels present the pictures which have been clicked in the first four clicks. Let $P_{Cli-Order}$ define the probability distribution of this pattern. As an example, if a user selects a medium image first, then a simple picture, and next two hard images, she gets $P_{Cli-Order}(SMHH) = 0.0581$ as her credit.

As it can be seen from Fig. 8, users tend to select simpler images in the earlier clicks, and subsequently harder ones in the later clicks. Besides, we already know that users click simpler pictures more correctly than hard ones. These two separate facts could be seen as a rational proof of our hypothesis that users would unconsciously respond to simpler images faster and more accurately. We also present the combination of these two facts in the form of a new pattern in the next subsection.

3.2.4 $P_{F-Pos-HR}$: Success rate of each click based on HR

The fourth pattern is the probability of each click's HR in combination with the probability of being clicked correctly. As mentioned before, this pattern be considered as a conditional probability function combined with the previous pattern. In order to present this pattern, eight different probability distribution functions are needed to show each click separately. Fig. 9 shows these distributions. The x axis represents two different levels of information. The first level shows the click number and each click contains four (S, M, H, and V) double beams. The green and red beams show the possibility of right and wrong clicks, respectively. We define eight different functions, $P_{Pos-HR1}$ to $P_{Pos-HR8}$ for click no. 1 to click no. 8. As an example, if the user clicks the images according to the following scenario, her credit is as below [8]:

- Click 1: M, right; $P_{Pos-HR1}(M - Pass) = 0.19$
- Click 2: V, wrong; $P_{Pos-HR2}(V - Fail) = 0.03$
- Click 3: S, right; $P_{Pos-HR3}(S - Pass) = 0.19$
- Click 4: S, right; $P_{Pos-HR4}(S - Pass) = 0.21$
- Click 5: H, right; $P_{Pos-HR5}(H - Pass) = 0.05$
- Click 6: M, wrong; $P_{Pos-HR6}(M - Fail) = 0.03$
- Click 7: V, right; $P_{Pos-HR7}(V - Pass) = 0.18$
- Click 8: H, wrong; $P_{Pos-HR8}(H - Fail) = 0.05$

Since these eight different credits are in the similar genre, we convert them to one credit. Different approaches such as addition, max or min selection, arithmetic or geometric mean could be applied. Rationally, maximum or minimum selection impair the impact of the rest. Thus, we suggest applying a geometric mean function to calculate the final credit in this pattern, $P_{F-Pos-HR}$, by equation 1:

$$P_{F-Pos-HR} = \sqrt[8]{\Pi P_{Pos-HRi}} \qquad (1)$$

where i is the click number, and varies from 1 to 8. Therefore, for the mentioned example

[8] Note that the numbers are rounded up for convenience, but were considered with five decimal-point accuracy in the calculations.

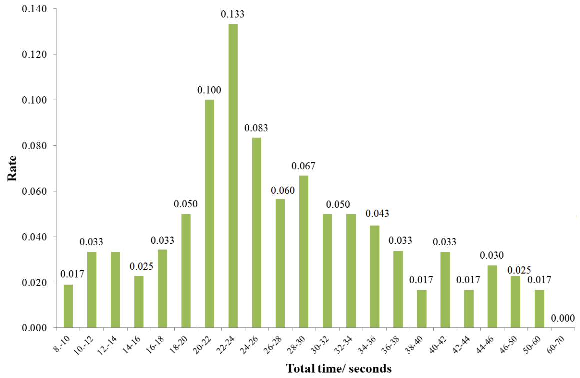


Figure 10: P_T: Success rate based on time

$P_{F-Pos-HR} = 0.086$. Fig. 9 illustrates that the possibility of correct answers decreases from the first click to the last one (the first four clicks are correct with the average possibility of around 85%, while it is less than 70% for the second four clicks). Moreover, it can be seen that the simple pictures have greater shares in the early clicks, and the harder ones appear more in the later clicks. This observation confirms our hypothesis.

3.2.5 P_T: Response time

The final pattern is the distribution of the total response time. The proposed captcha takes different times for different users; the probability distribution is shown in Fig. 10. We use the average of these values as the average response time (see Section 5.2). P_T is defined according to the probability distribution up to 60 seconds. For example, if it takes 19 seconds for a user to complete a test, the system would grant him the credit of 0.05 in this pattern.

3.3 Inference engine

The suggested patterns lead to five different probability distribution functions. Note that the eight functions presented for the forth factor (P4) were converted to one function by geometric mean function. We apply the same approach to calculate the user's final credit (C_{Final}) in equation 2:

P_x	Value	Max value of pattern
P_{No}	0.322	0.335
P_{HR-No}	0.481	0.481
$P_{Cli-order}$	0.148	0.155
$P_{F-Pos-HR}$	0.086	0.316 (ave. of maxs of 8 clicks)
P_T	0.100	0.133
$C_{Final} = \sqrt[5]{\frac{\Pi P x}{C}} = 0.716$		$C = \Pi max(P_x) = 0.001$

Table 1: Final credit estimation for a sample human user

$$C_{Final} = \sqrt[5]{\frac{P_{No} \times P_{HR-No} \times P_{Cli-order} \times P_{F-Pos-HR} \times P_T}{C}} \quad (2)$$

where C is a constant which is calculated by the multiplications of the highest credits of our users in each pattern in our experiments ($C = \Pi max(P_x)$, where P_x is the credit in the respective pattern).

As an example, consider a human user with the probability values as presented in the second column of Table 1. The maximum values of our probability functions from Fig. 6 to Fig. 10 are presented in the third column. Note that for pattern no. 4, we use the average of the maximum values of eight clicks to calculate C. As presented in the table, the final credit of this user is equal to 0.716.

In order to observe the functionality of our suggested credit system, we performed a manual test. We intentionally provided random answers to a few PiSHi tests. Then we compared the given credits in each pattern, as well as the final credits with our rational expectations. We found out that the output of the proposed functions matched the expected results. Please see Appendix A for the details of our experiments for simulating random guessing tests.

4 Updating the database

Updating a captcha's picture database improves both the security and usability aspects of the system. It diminishes several kinds of attacks by replacing the old images by new ones. We propose a new method for updating the database based on user's credits, which we call the *credit-based* approach. In this approach,

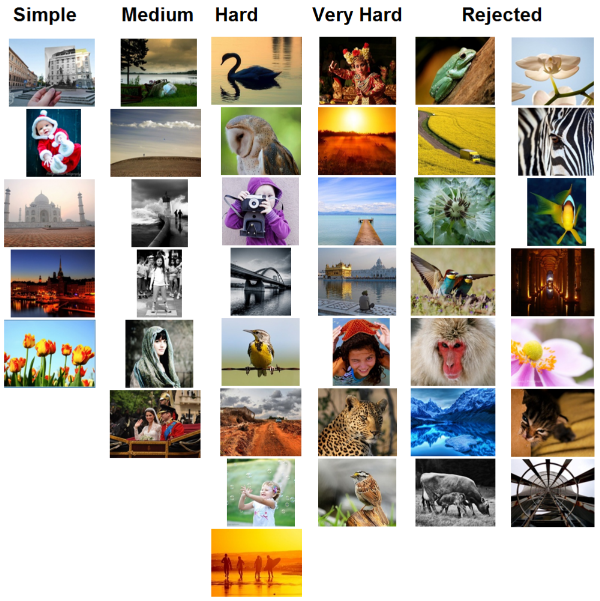


Figure 11: Pictures in the temporary database categorised based on the credit-based approach

the more credit the user gets, the more impact he has on updating the database. In our system, we have two picture databases; the temporary database, and the main database (Fig. 2). The *temporary database* contains the pictures which are under the observation of the users to be either added to the main database or removed from the system. We propose PiSHi's interface contain 10 images; eight for evaluating the user, and two for updating the database. Our credit-based method multiplies the user's final credit by his answer for the new picture, where it assigns 1 for a correct answer and -1 for a wrong one. Then, PiSHi's inference engine adds all the users' answers to each other to find out whether the new picture is accepted (and to which HR category it belongs to), or rejected.

In order to evaluate our credit-based idea, we downloaded 40 new images from featured photos of Picasa web albums[9]. These pictures were completely random and we did not know anything about their contents. We inserted them into our *temporary image database* in order to evaluate them during our final experiments (see Section 5.1 for details) through 180 different tests. We estimated hardness rates of all new images based on two different approaches; simple counting (accepting or rejecting the pictures based on counting the number of correct answers) and credit-based. Fig. 11 shows the categorized pictures based the credit-based approach.

We analysed the results in two ways; manually, and experimentally. The findings of our manual analysis show that the credit-based approach sorts the new images better. For example, the *dandelion* picture which was rejected by the credit-based algorithm, was categorised as M in the simple counting method. However, it is an ambiguous round-shape subject which makes it challenging for users to recognise its top part. For experimental evaluation, we performed another experiment in which we showed the distorted versions of all 40 pictures in five rounds to 10 new users (each original picture was seen 50 times). In order to evaluate the performance of our approach, we only showed one picture per page in each round. The pictures' identification rates in this experiment were very close to the results of the credit-based method, while they were different from the results of the simple counting approach.

Apart from the good performance of the credit-based method, it also categorises the pictures faster i.e. it needs less users to make a decision about a new picture since it increases the impact of the good voters. Another advantage of this approach is that it alleviates the impact of random guessing on new images since a random guessing attacker will not get enough credit to vote.

As it can be seen in Fig. 11, since we did not apply any filter when selecting the new pictures for the temporary database, the H, V, and rejected pictures are more than the S and M ones. In order to set a proper frequency for updating the captcha database, we need to consider the category with slower updating speed. This speed would change based on the web source that we use for new pictures. New images can be included from different sources by using *Image selector from the web* component as presented in Fig. 2. While social network websites such as *Flickr.com*

[9] picaweb.google.com/lh/explore

might raise copyright issues, free websites for public domain images such as *wikimedia.org* will not. Such free services are appropriate for a practical captcha.

5 Evaluation

In this section, we evaluate and analyse different aspects of our system. First, we evaluate the usability, security, and complexity of PiSHi based on real world data which we collected in our final experiments. Next, we discuss the automation of the system, and our access control policies about of the algorithms and databases.

5.1 Data collection for testing

In order to evaluate our final suggested captcha system with the updating feature, we performed another experiment for building our test dataset. We collected data from 30 new users (male and female, 18 to 50 year old, university student and staff). We presented them with PiSHi's final interface, as shown in Fig.1. As it can be seen, this interface includes 10 images in the scale of 200×200 pixels. These 10 images include eight images for user evaluation (two S, two M, two H, and two V), and two images for updating the database. These pictures are shown in random places on the webpage. The users could click on the top of each of them in any preferable order. The border colour of the clicked images changes in order to help the users with identifying the remained ones.

We asked each user to complete the presented captcha challenge and repeat it for six times. We ended up with 180 different records at the end of the experiment. Similar to the training data collection, we presented each user with a brief description of the experiment. We also included a practice round which we did not log. Another video file was clickable as the system's help in the case of any difficulty. As expected, all users could easily understand the system and follow the experiment.

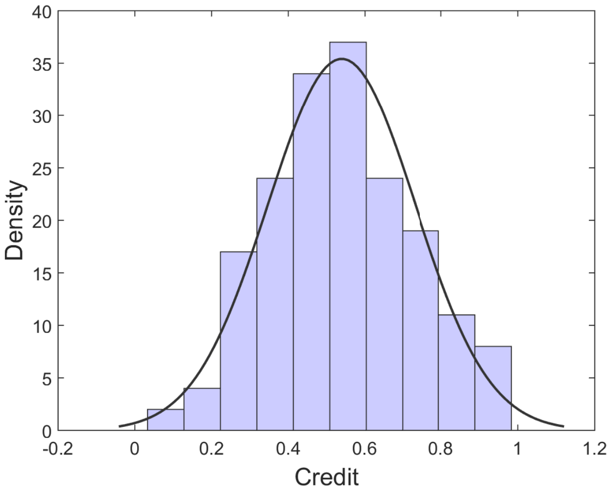


Figure 12: Final credits distribution of human users

5.2 Usability metrics

Before reporting PiSHi's response time and rate, we are required to determine a minimum credit as the system's threshold for passing PiSHi's challenges. For this we need to find a balance between the usability of the system for human users, and its security against random guessing attacks. Fig. 12 shows the histogram of the users' credits of our final experiment, calculated from equation 2. As it can be seen, the average credit of a human user is around 0.55 in our system. On the other hand according to our analysis in Appendix A, a random guessing attacker's credit would be less than 0.1, which is far away from an average human user. Hence, we consider 0.1 as the minimum credit required by the system. Based on this minimum required credit, the success rate of PiSHi is 99.44%. In addition, users on average respond to a sample challenge in 28.10 seconds, with an expiration time of 60 seconds. And as the final usability metric, PiSHi requires the users to click once on each picture, which is an easy response mechanism for a captcha.

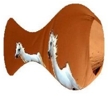
Picture							
Effect & setting	Original None	Colour Tint	Cut Circular cut	Rotation Left rotation	Light Full light	Texture Pencil Sketch	Geo func No rotation
Tineye	11	12	11	10	10	9	0
Google	293	293	279	234	216	46	0

Figure 13: The results of reverse image search engines for different types of picture distortions

5.3 Security metrics

Random guessing. According to our analysis in Appendix A, the random guessing attacker's final credit would be less than 0.1. Hence, an attack program would not be able to pass the proposed captcha by random guessing, since it does not get enough credit. We have designed the system in a way that human users would benefit from their natural error production approach when solving our suggested pictorial system. This unique design reduces the chance of random guessing attacks close to zero since random responses to the system are distinctively different from human patterns.

Direct matching attacks. The geometric transformation algorithms, in combination with multiple rotation rounds produce a large search space by assigning different values to different variables in runtime [13]. Hence, building a look-up table, and updating it according to the captcha database is a complex and expensive procedure for the attacker. Note that although we only used six geometric transformations in this paper, they actually could be chosen from a much wider set. Choosing a random 3D object with random settings in the runtime would even further improve the security of the system. Therefore, we offer little support for an attacker to recover the original pictures by searching the distorted versions in a look-up table.

Reverse image search engines. As mentioned in Section 1.3, evaluating the strength of image captchas against reverse image search engines is a common approach to assessing the security of a system. Here, we observe the effectiveness of our transformations against reverse matching by using reverse image search engines including Google image search, and Tineye.com. These selected reverse image search engines are fast, web-based, and multi-purpose. We used Picasa[10] and altered a few original pictures. After each modification, we submitted the pictures to both Tineye and Google to see if they can find any samples of the submitted pictures. For example, we chose a horse picture and submitted it to the mentioned engines and found out the number of findings. Then, we distorted it in several ways; changing the colour, light, angle, texture, size/ cut, etc. Fig. 13 shows the results of Tineye and Google for each effect. The effects were set to 50% of the maximum possible amount allowed in Picasa. The only exception is the circular cut which was done by MS Paint.

As it can be seen in the picture, all changes - except the geometric transformations- are recognisable by these engines. We observed that these reverse search engines are able to identify some of these effects such as resizing/ cutting, and colour better than others including light, rotation, and texture change. However, the tested reverse image search engines are absolutely incapable of finding any samples of our distorted pictures produced by geometric transformations. As a result, the suggested geometric functions in PiSHi are resilient against reverse image recognition methods, and prevent the attacker from mapping the distorted images to their original ones.

Customised machine learning attacks. As explained before, upright orientation is a hard problem to learn for a machine, especially when the content of the picture is partially visible. In addition, machines need to solve another hard problem which is finding

[10] picasa.google.com/

the easier images to click earlier and better. Moreover, continuously updating the database makes security attacks even harder since the attacker needs to construct a new look up table, and/or to retrain the system with the new images added to the database. Moreover, we showed that available machine learning algorithms applied in commercial image search engines are incapable of learning the pictures used in PiSHi. Therefore, undertaking a machine learning attack should prove difficult since PiSHi is based on multiple hard problems, updates its database, and is robust against the reverse search engines. However, a customised machine learning attack which is designed for this captcha is not implemented in this paper. These types of advanced attacks are challenging and out of the scope of this work.

5.4 Complexity metrics

As mentioned in section 1.3, we suggest to evaluate complexity metrics for a captcha system. These metrics are server, client, and communication complexities. In PiSHi, the server side components use some pre-implemented functions in Matlab which have linear computational complexity depending on the size of the original picture. PiSHi sends 10 images (200×200 pixs) to the clients, with an average size of 7.8 kB of each. The communication complexity would be even less if the image size was decreased by an image processing algorithm. The client returns eight clicked points to the server as the captcha answer. Also, the clients do not need any extra software, which makes PiSHi's client complexity almost zero.

5.5 Automation and availability of the system components

All our suggested algorithms are automated. In addition, all our algorithms and databases are public. Here we discuss how giving open access to different parts of our system will not affect its security.

Algorithms. We described our algorithms to be non-deterministic i.e. they produce different outputs for each run. Selecting images from the web and from the image databases is done randomly. The transformation algorithms use random values for multiple runtime variables. Also, the inference algorithm's decision to pass or fail the user is based on the user behaviour. Obviously, the values of the runtime variables including the produced key image should be kept private during the test. As the result of the non-determinism of the algorithms, making them public should not impose any security risks to the system.

Databases. Both the picture databases (main and temporary), and also the pattern database can be made public without risking security. Exposing the original pictures to attackers would not give them any advantages since these pictures will be randomly distorted before being presented to the users. The distortion algorithms are one way functions. Therefore, the attacker will not be able to relate the distorted picture to its original one. Knowing the patterns without being able to identify the original picture of a distorted one would not give the attacker any advantage either. One might argue that mapping distorted images to the original ones could be feasible by using direct matching attacks or machine learning techniques. However as we discussed in the previous subsections, performing such attacks are not straightforward on our proposed captcha.

6 Comparison

In this section, we compare PiSHi with previous image-based captchas as well as text-based captchas in terms of usability, security and complexity.

6.1 Image-based CAPTCHAs

We compare PiSHi with four image-based captchas; Microsoft Asirra [1], What's UP CAPTCHA [10], Sketcha [17], and Multiple Seimcha [13]. Microsoft Assira is interesting for us since it is the first image captcha which allows human mistakes in its design. The other three works are based on upright orientation. Table 2 reports different metrics for PiSHi as well as these image captchas.

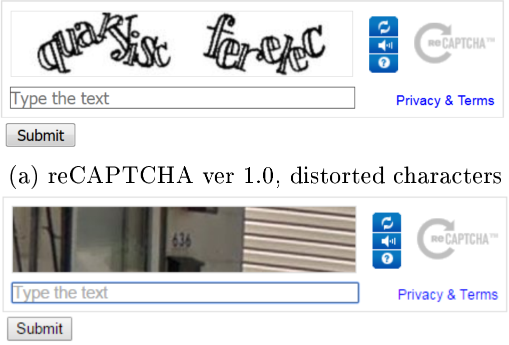
Usability. As mentioned before, PiSHi has a success rate of 99.44% which is the best result compared to others. Besides, in terms of timing, it roughly counterbalances the rest. Asirra [1] has a success rate of 99.96% when it presents the test two more rounds to the user if he fails in the first round, though this increases the response time to 45 seconds.

Security. PiSHi reduces the chance of an average random guessing attack to zero which outperforms all others. It also has some other security features including being based on multiple hard problems, updating the databases, and being resistant against image reverse search engines. The mentioned security features not only diminish all types of security attacks, but also improve the usability of the system. In contrast, none of the previous works possess all the mentioned security features together. They either don't update the databases or are weak against reverse image search engines. Most of them either use original images (Asirra), or some weak distortions such as cut and rotation (What's up) which are recognizable by search engines. Sketcha uses pen drawings which can be still recognisable by search engines to some extent. Though Sketcha pictures are based on 3D models which makes it hard for the attacker to recover the original 3D models. By comparison, our distorted images produced by geometric functions are not recognisable by search engines at all.

Complexity. In terms of server complexity, most of the existing works and also PiSHi have transformation functions in the server side. Also, the communication complexity is roughly similar for all, however the pixels of the pictures in PiSHi are less than most of other works. PiSHi's response mechanism is extremely easy, requiring a single click per image. It does not need any extra software on the client side which is an advantage compared to Sketcha which needs a program to rotate images on the client side.

6.2 Text-based CAPTCHAs

Many different text-based captchas have been designed and implemented over the years. Among them, reCAPTCHA [2] is the most widely used captcha provider in the world being used by more than half a million live websites[11] including companies such as Google and Facebook. reCAPTCHA has two versions at the moment; ver 1.0 (text-based) and ver 2.0 (image-based). At the time of these experiments (October 2015) when testing reCAPTCHA ver 1.0 through Google demo[12], it demonstrates two different types of text-based captchas. We categorise these two variations based on the type of the challenges and name them as *distorted characters* (Fig. 14, a), and *characters in pictures* (Fig. 14, b). In this section, we compare our work with both variations.



(a) reCAPTCHA ver 1.0, distorted characters

(b) reCAPTCHA ver 1.0, characters in picture

Figure 14: reCAPTCHA ver1.0 variations

Usability. For evaluating the usability metrics, we asked 30 users (male and female, aged 22 to 38 university student and staff) to complete six reCAPTCHA tests by using Google demo. The users were asked not to skip any of the given challenges and provide answers to all of them. We recorded the response time and the success rate per test. This Demo provides random tests including both variations of ver 1.0. We calculated the response times and success rates separately for these two variations. The results are presented in Table 3. As it can be seen, PiSHi's success rate is higher than both reCAPTCHA variations. However, its response time is slightly longer than the *distorted characters* variation, and much longer than the *characters in pictures* variation. The reason is that the second variation normally presents users with very short numbers, e.g. three digits, which in return risks the security of the system.

In terms of the response mechanism, PiSHi requires users to click on pictures, while they have to type multiple characters and numbers to pass re-

[11] trends.builtwith.com/widgets/reCAPTCHA

[12] www.google.com/recaptcha/demo/

Criteria	**PiSHi**	Asirra [1]	What's up[10]	Sketcha [17]	Multiple seimcha[13]
Usability					
Success rate	**99.44%**	83.4%	84%	88%	92%
Response time	**28.10 s**	15 s	NA	32.5 s	26 s
Security					
Hard problems	**img. upside** **partial img.** **decision making**	animal rec.	img. upside	img. upside	img. upside partial img.
Ave. random guessing attack	**0%- min credit of 0.1 (8 imgs)**	0.39% (8 imgs) 0.024% (12 imgs)	0.009% (3 imgs)	0.001% (8 imgs)	0.009% (8 imgs)
Reverse search engine resistance	**yes**	no	no	yes	yes
Database update	**yes (credit-based)**	yes	no	yes (counting)	no
Complexity					
Server side	**geometric functions**	none	none	3D models to drawings	geometric functions
Server sending pics (pixel)	**10 (200*200)**	12 (350*166)	3 (180*180)	10 (240*240)	8 (280*210)
Client sending (per img)	**single click**	single click	rotated angle	1.5 clicks	single click
Extra software in client side	**no**	no	yes	yes	no

Table 2: Comparison between PiSHi and previous image captchas

Works Usability Metrics	PiSHi	reCAPTCHA (distorted characters)	reCAPTCHA (characters in pictures)
Success rate	99.44 %	88.67 %	91.33 %
Response time	28.10 s	24.47 s	7.20 s
Response mechanism	Clicking pictures	Typing characters (may include numbers)	Typing numbers (may include alphabetic characters)

Table 3: Usability comparison between PiSHi and reCAPTCHA

CAPTCHA. It has been agreed by reCAPTCHA developers that typing a line of distorted text is tediously harder than clicking pictures, specially for mobile users [22].

Security. In terms of direct matching attacks, strong text-based captchas such as reCAPTCHA *distorted characters* version are resilient, since they generally produce the challenge in real time by randomly choosing characters and distorting them. To estimate the chance of random guessing, *distorted characters* variation consists of two words; a verification word (reCAPTCHA server knows the answer), and a read word (comes from an old book, which can be entered incorrectly)[13]. On the verification word, reCAPTCHA allows an *off by one* error. Depending on the length of the verification word, the random guessing chance changes. For example, with a random word consisting of five alphabetical characters, the probability of the attack would be around

[13] code.google.com/p/recaptcha/wiki/FAQ

$\frac{1}{26^4} = 0.0002\%$. On the other hand, the strength of the *characters in the pictures* variation depends on the number of characters that the user is presented with. As mentioned earlier, the presented pictures mainly include number sequences e.g. 3-digit numbers; which yields to a probability of $\frac{1}{10^3} = 0.1\%$ for a random guessing attack. As a result, PiSHi outperforms both variations of text-based reCAPTCHA when encounters a random guessing attack.

Furthermore, several researchers have tried to break text-based captchas using machine learning techniques. There exist different works such as [11], [8], [6], [21] in which demonstrate that overcoming text-based captchas by automated programs is indeed practical. For example, Starostenko et al. in [21] show that their method can successfully attack text-based reCAPTCHA in only 0.49 seconds. Their method can segment the shown characters in 75.9% cases, and recognise them with 95% accuracy. As a matter of fact, Google acknowledged these sorts of attacks in December 2014, stating that the most difficult variants of distorted texts can be solved with 99.8% accuracy by advanced methods [22]. Most of these methods adopt advanced image processing algorithms for, first, segmenting the noisy characters in the picture, and then identifying them in each segment [21]. In terms of comparison, these methods are not applicable to an image captcha such as PiSHi.

Complexity. Almost all text-based captchas have some sort of server side transformations in order to produce the final pictures. The pictures sent to the clients could be either black and white, or coloured. And the client returns the typed characters to the server. By comparison, PiSHi counterbalances a typical text-based captcha since it uses transformations in the server side, sends coloured pictures, and return click points to the server.

Overall, the good performance of our system, in combination to its security features, and its scalability offer the ideas in this paper to be considered in practical captcha systems.

7 Discussions

Applications. In this work, we proposed the use of the users' interaction models with pictorial systems as a new approach to be considered when designing security systems. As a proof-of-concept of the idea, we implemented and evaluated PiSHi which is an image captcha. We believe analysing human interaction patterns when encountering pictorial systems could be used for other purposes such as identification of age, gender, profession, etc. For example an online age recognition system could be really helpful in multiple ways such as preventing children to have access to certain online contents when their parents are not monitoring them. Moreover, we propose our system to be used in mobile devices. The simple response mechanism of PiSHi makes it a mobile-friendly service. Tapping on the touch screen is much easier than other methods used in other captchas e.g. typing characters, or choosing labels from a list.

Patterns. We suggested five different factors to include in our model, and we showed their effectiveness in Section 5. However, some of them might be improvable. For example, response time may not contribute to the system security, since it is easy for the attacker to identify and fabricate the best response time. However, we estimated that even if the attacker gets the highest credit in this factor, his final credit would be 0.1273 (see Appendix A), which is still far from the user average credit (0.5492). Moreover, our suggested patterns have been built by presenting eight pictures to the users, while the final system includes two extra pictures for updating the database. The new presentation of the system might affect the patterns. While PiSHi is performing at a high accuracy using the current patterns, updating them according to the final presentation of the system might further improve the results. Also, other strategies such as adding weighting based on the importance of some particular factors may improve the system performance even more.

Users. We tried to cover a wide range of ages and backgrounds of users when performing our user studies in different phases of this project. Yet, a larger scale experiments in a longer time would observe the suggested ideas more accurately. By performing the experiments in a longer time and with a wider set, we might find new patterns due to possible changes in user behaviour after using the system for a while. While such considerations are valid, we believe that they would not invalidate the general results. Furthermore, this paper presents a proof-of-concept of PiSHi in order to demonstrate the feasibility of including human behavioural analysis in a captcha design. For a commercial version of the proposed system, more comprehensive studies are required.

8 Conclusion

In this paper, we presented PiSHi (pictorial intelligent system for human identification) as a new image captcha. Overall, our captcha is the first which models human interaction patterns by considering human users errors when working with a pictorial system. We designed our system based on these behavioural patterns, and granted the users credits based on comparing their entire engagement with the system with these patterns. These credits were used in order to passing or failing the users, as well as updating the captcha database. PiSHi has a high success rate (99.44%), is easy for human users to interact, and is secure against random guessing attacks and reverse image search engines.

As future work, we would like to improve our extracted patterns by applying more advanced pattern recognition methods. We also aim to extend PiSHi's ideas to other contexts such as age recognition. We are also interested in discovering and modelling other hard concepts for machines such as *beauty.* And finally, as a reverse approach, we are interested in finding problems which are easy for machines and hard for humans, such as optical illusions, to use in a captcha.

9 Acknowledgement

We would like to sincerely thank our volunteer users who contributed to the user studies of this work. We thank Prof. Brian Randell, Dr. Siamak F. Shahandashti, and Mr. Sami Alajrami from Newcastle University, UK, Dr Mohesn Kahani, Dr. Hamidreza Pourreza, and Mr. Hamid Mansouri from Ferdowsi University of Mashhad, Iran, for their invaluable feedback during different phases of this project. We also would like to thank anonymous reviewers of this paper for their constructive comments. The last author of the paper is supported by ERC Starting Grant No. 306994.

References

[1] Asirra: A captcha that exploits interest-aligned manual image categorization. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 366–374, New York, NY, USA, 2007. ACM.

[2] L. V. Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. RECAPTCHA: Human-based character recognition via web security measures. *Science*, 321:1465–1468, 2008.

[3] V. Atanasiu. Roirotate; fill corners of rotated image. Matlab Function, 2002. http://www.mathworks.com/matlabcentral/fileexchange/1825-fill-corners-of-rotated-image.

[4] K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski. Designing human friendly human interaction proofs (hips). In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 711–720, New York, NY, USA, 2005. ACM.

[5] M. Chew and J. Tygar. Collaborative filtering captchas. In H. Baird and D. Lopresti, editors, *Human Interactive Proofs*, volume 3517 of *Lecture Notes in Computer Science*, pages 66–81. Springer Berlin Heidelberg, 2005.

[6] A. S. El Ahmad, J. Yan, and L. Marshall. The robustness of a new captcha. In *Proceedings of the Third European Workshop on System Security*, EUROSEC '10, pages 36–41, New York, NY, USA, 2010. ACM.

[7] F. F. K. M. G. Hackman, Thomas Jr. An experiential approach to teaching students about usability and hci. In *Newsletter ACM SIGCHI Bulletin*, volume 26, Jan 19994.

[8] H. Gao, W. Wang, J. Qi, X. Wang, X. Liu, and J. Yan. The robustness of hollow captchas. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer &#38; Communications Security*, CCS '13, pages 1075–1086, New York, NY, USA, 2013. ACM.

[9] P. Golle. Machine learning attacks against the asirra captcha. In *15th ACM Conference on Computer and Communications Security (CCS)*, CCS2008, 2008.

[10] R. Gossweiler, M. Kamvar, and S. Baluja. What's up captcha?: A captcha based on image orientation. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 841–850, New York, NY, USA, 2009. ACM.

[11] A. Hindle, M. Godfrey, and R. Holt. Reverse engineering captchas. In *Reverse Engineering, 2008. WCRE '08. 15th Working Conference on*, pages 59–68, Oct 2008.

[12] K. A. Kluever and R. Zanibbi. Balancing usability and security in a video captcha. In *Proceedings of the 5th Symposium on Usable Privacy and Security*, SOUPS '09, pages 14:1–14:11, New York, NY, USA, 2009. ACM.

[13] M. Mehrnejad, A. Bafghi, A. Harati, and E. Toreini. Multiple seimcha: Multiple semantic image captcha. In *Internet Technology and Secured Transactions (ICITST), 2011 International Conference for*, pages 196–201, Dec 2011.

[14] M. Mehrnejad, A. Bafghi, A. Harati, and E. Toreini. SEIMCHA: A New Semantic Image CAPTCHA Using Geometric Transformations. In *ISecure Journal*, Vol. 4 No.1, September 2012.

[15] M. Mehrnejad, A. Bafghi, A. Harati, and E. Toreini. A new image based CAPTCHA based on geometric transformations. In *8th International ISC Conference on Information Security and Cryotpgraphy (ISCISC)*, 2011.

[16] I. Polakis, M. Lancini, G. Kontaxis, F. Maggi, S. Ioannidis, A. D. Keromytis, and S. Zanero. All your face are belong to us: Breaking facebook's social authentication. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, pages 399–408, New York, NY, USA, 2012. ACM.

[17] S. A. Ross, J. A. Halderman, and A. Finkelstein. Sketcha: A captcha based on line drawings of 3d models. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 821–830, New York, NY, USA, 2010. ACM.

[18] Y. Rui and Z. Liu. Artifacial: Automated reverse turing test using facial features. *Multimedia Systems*, 9(6):493–502, 2004.

[19] L. von Ahn, M. Blum, N. Hopper, and J. Langford. Captcha: Using hard ai problems for security. In E. Biham, editor, *Advances in Cryptology, EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 294–311. Springer Berlin Heidelberg, 2003.

[20] B. B. Zhu, J. Yan, Q. Li, C. Yang, J. Liu, N. Xu, M. Yi, and K. Cai. Attacks and design of image recognition captchas. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 187–200, New York, NY, USA, 2010. ACM.

[21] O. Starostenko, C. Cruz-Perez, F. Ponga, and V. Aquino. Breaking text-based CAPTCHAs with variable word and character orientation. In *Pattern Recognition*, 48(4):1101–1112, 2015. ACM.

[22] V. Shet. Are you a robot? Introducing "No CAPTCHA reCAPTCHA". blog post at https://googleonlinesecurity.blogspot.com.au /2014/12/are-you-robot-introducing-no-captcha.html. 2014.

[23] P. Anantharatnasamy, K. Sriskandaraja, V. Nandakumar, and S. Deegalla. Fusion of Colour, Shape and Texture Features for Content Based Image Retrieval. In *The 8th International Conference on Computer Science and Education* , ICCSE 2013, Sri Lanka, 2013.

[24] R. Anderson. Security Engineering: A Guide to Building Dependable Distributed Systems, second edition, chapter two. By *Wiley Publishing*, 2008.

[25] D. Lorenz, et al. Generating Secure Images for CAPTCHAs through Noise Addition. In *The ACM Symposium on Access Control Models and Technologies*, SACMAT'15, Austria, 2015.

A Random guessing attack analysis

In this section, first we calculate the credit of an average random guessing attack according to the extracted patterns presented in Section 3.2. Next, we present the details of an experimental test to simulate multiple random guessing attacks, and present the credit distribution gained by these attacks.

A.1 Average attacker's credit

P_{No}**:** Based on our analysis, after distorting the original pictures, on average, 17.5% of the output pictures is the answer area visible to the users. Hence, a random click would be correct in 17.5% of the cases; equal to 1.4 images out of eight. As it can be seen in Fig. 6, the probabilities of the credit of clicking on one and two correct pictures are equal to 0 and 0.00645, respectively. A linear average, which is equal to 0.0033 (rounded up), would give us an approximate estimation of the attacker's credit in this pattern.

P_{HR-No}**:** As we explained before, the probability of some of the less probable cases are not shown in Fig. 7 since there were either no, or few samples in our dataset. For example, in the categories of 1 and 2 correct answers, most of the cases did not happen in our experiments. Therefore, as explained in Section 3.2.2, we assign the lowest non-zero value of the corresponding function, which is 0.0200, to the attacker.

$P_{Cli-order}$**:** In order to calculate the output of this function for the attacker, we should divide the sum of the function amounts, which is 1, into the number of inputs, which is 19. Hence, an attacker would get $\frac{1}{19} = 0.0526$ as its credit in this pattern.

$P_{F-Pos-HR}$**:** For an attacker, the chance of clicking on different categories of images is equal to 0.25. As mentioned earlier, an attacker would click a picture from whatever category (S, M, H, V) correctly with a possibility of 17.5%. Accordingly, he clicks the picture incorrectly with a chance of 82.5%. Hence, we are able to estimate the credit for each click in this pattern by applying a weighted sum equation:

$P_{Pos-HRn}(ran) = 0.175 \times Ave(P_{Pos-HRn}(X - Pass)) + 0.825 \times Ave(P_{Pos-HRn}(X - Fail))$

where n is the click number, X is the representative of all category of images (S, M, H, and V), and ave is a simple mean function. Table 4 calculates the attacker's final credit obtained from this pattern according to equation 1. As it can be seen, an average attacker would score 0.0799 in this pattern.

P_T**:** In terms of response time, a machine is much faster than a human user. Therefore, we consider the amount of the first category -8 to 10 seconds and less- which is 0.017 as the credit of this pattern to the attacker.

Final Credit (C_{Final})**:** As presented in Table 5, and according to equation 2, the attacker will get around 0.0839 as its final credit by a random guess attack. Distinctively, the attackers' final credit is too low comparing to an average human user (0.55).

A.2 Attack simulation

In this experiment we wrote a Matlab code in order to simulate 180 different random guessing attacks on our system. As it can be seen, the code randomly selects an unseen picture from different categories of difficulties. Then it randomly calculates a response for this picture according to its chance; 17.5% correctly clicked, 82.5% wrongly clicked. Next, the code saves the sequences of all eight pictures, and if they have been clicked correctly or wrongly in the answer array. Finally another module is called to calculate the final credits of all randomly produced tests according to our patterns.

A histogram of these final credits is presented in Fig. 15. As it can be seen, the credit associated with the peak

Click no.	$ave(Pass)$	$ave(Fail)$	$P_{Pos-HRn}$
1	0.2113	0.0387	0.0689
2	0.2161	0.0339	0.0657
3	0.2097	0.0403	0.0699
4	0.2145	0.0355	0.0668
5	0.1903	0.0597	0.0825
6	0.1952	0.0548	0.07937
7	0.1839	0.0661	0.08671
8	0.1726	0.0774	0.09406
Chance	0.175	0.825	
$P_{F-Pos-HR}(ran) = \sqrt[8]{\Pi P_{Pos-HRn}} = 0.0761$			

Table 4: Random attacker's credit from $P_{F-Pos-HR}$

P_x	Value
P_{No}	0.0033
P_{HR-No}	0.0200
$P_{Cli-order}$	0.0526
$P_{F-Pos-HR}$	0.0761
P_T	0.0166
$C_{Final}(ran) = \sqrt[5]{\frac{\Pi Px}{C}} = 0.0839$	

Table 5: Final credit estimation for an average random guessing attack

of this histogram is slightly less than the credit of our estimation of an average attacker. The reason is that in the case of clicking none of the images correctly, the system will give a final credit of 0 to the attacker. This case happened in this simulated experiment several times, while we did not have such data in our patterns for our human users. If we compare this fit with the one from Fig. 12 for human users credits, we discover that the average credit for an average human user (0.55) is far away from the machine's credit (simulated average result: 0.025, estimated average result: 0.084). This helps us to define a proper threshold for our system as explained in Section 5.

```
for j= 1: 180
t= zeros(8); tc= 1; flagg= 0;
for i= 1:8
flagg= 0;
while(flagg == 0)
b= floor(1+8*rand(1,1));
result = find(t == b);
if result> 0 flagg= 0;
else flagg= 1;
end
end
t(tc)= b; tc= tc+1;
c= (100*rand(1,1));
if c< 17.5 anss= 1;
else anss= 0;
end
Ansswer(j,i,1)= mod(b,4)+1;
Ansswer(j,i,2)= anss;
end
end
Credit = FinalCredit(Ansswer)
histfit(Credit)
```

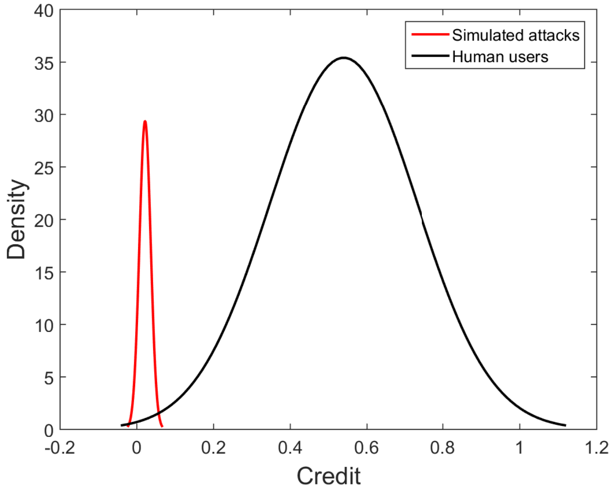


Figure 15: Histograms of final credits of simulated random guessing attacks vs. human users