Detecting human attacks on text-based CAPTCHAs using the keystroke dynamic approach

Suliman A. Alsuhibany | Latifah A. Alreshoodi

Department of Computer Science, College of Computer, Qassim University, Buraydah, Saudi Arabia

Abstract
A Completely Automated Public Turing Test to Tell Computers and Humans Apart (CAPTCHA) is a simple test that is used on websites to differentiate between human users and automated attacks that indulge in spamming and other fraudulent activities. A text-based CAPTCHA is the most popular security technique used by many websites on the Internet, such as Microsoft, Google and eBay, to secure their sites from automated attacks. By design, however, a CAPTCHA is unable to differentiate between a legitimate human user and a human-based attacker. This may make websites vulnerable to human-based attacks while using CAPTCHAs. Hence this article proposes a novel defence system using the keystroke dynamic approach. To evaluate our system, a laboratory experiment was conducted and the results showed that the proposed system is able to detect human-based attacks on text-based CAPTCHAs effectively with a 100% detection rate.

1 | INTRODUCTION

Distinguishing between humans and computers is one of the main security problems for online services and websites. To address this, a Completely Automated Public Turing Test to Tell Computers and Humans Apart (CAPTCHA) was introduced as a challenge–response test to distinguish between a human user and a malicious bot (i.e. computer automated programme, sometimes called script or bot) [1]. It is used mainly to protect the online service provider's systems, resources and web applications from malicious attacks and misuse [2].

There are many categories of CAPTCHA, such as audio-based, image-based and text-based. Text-based CAPTCHAs are the most common security technique used by many online services such as ticket reservations as well as several websites on the Internet (e.g. Microsoft, eBay, Google). This is because they are simple to implement [3] and are easy for users worldwide to solve without much instruction because they require users only to recognise characters and/or digits [4]. A text-based CAPTCHA works by automatically generating a test on an image in the registration form or login form. This image includes some numbers and upper- and lower-case letters that the user is asked to solve. The test is designed to be easily solved by users, but it is difficult for computers. If the user solves the test correctly, it is confirmed that he or she is human and the registration is completed successfully [5]. In this article, the term CAPTCHA refers to text-based CAPTCHA only.

The use of CAPTCHA has grown in recent years; over 100 million CAPTCHAs are solved daily [6]. There are generally two types of attacks against CAPTCHAs: automated and human-based. Automated attacks are based on image processing and machine-learning algorithms to solve CAPTCHAs [7], whereas human-based attacks are based on human intelligence that involves hiring a human solver who is paid for a specific number of successful attempts [8]. This can be done by exploiting cheap human labour located in developing countries (sweatshops) [8].

Previously, the dominant type of attack to break the security provided by CAPTCHAs was automated [9]. However, with the improvement in CAPTCHAs, this attack has become complex and costly, as in Zi et al. [34] and Wang et al. [10]. A human-based attack is simpler and less costly because a human solver earns $0.75 per 1000 CAPTCHAs solved in an hour [8]. Also, it is a guaranteed way to bypass security provided by CAPTCHAs because the solvers are human, not automated codes [11]. Most existing studies concentrate on developing or preventing automated attacks; only a few focus on preventing human-based attacks. Thus, malicious entities continue to defeat the security provided by CAPTCHAs through human-based attacks [8]. Furthermore, most if not all existing
text-based CAPTCHAs are vulnerable to human-based attacks because they do not use a reliable technique to detect human-based attacks by differentiating between a legitimate human user and a human attacker [7]. For these reasons, there has been an increase in the use of human-based attacks; it has become an effective and economical attack option for malicious entities [8,11]. Accordingly, it is necessary to improve the security of the text-based CAPTCHA to protect it not only from automated attacks but also from human-based attacks.

Keystroke dynamics has become an active research field owing to its low cost and the ease with which it can be integrated into the targeted systems [12]. Keystroke dynamics is a behavioural biometric authentication method that uses a person’s typing patterns (typing rhythm) on digital devices such as a computer keyboard to validate her or his identity [13]. As mentioned in Monrose and Rubin [14], the keystroke dynamics is ‘not what you type, but how you type’. In this type of authentication, it can be done without the knowledge of a user and there is no need for additional work by the user to authenticate; he or she types in text as usual [13]. A unique keystroke profile or template can be created from different typing features [12] detected when a user presses or releases keys on a keyboard, such as the duration of a key or key hold time (i.e. the time a certain key is held down), the latency (i.e. the time between two consecutive keys such as releasing the first key and pressing the second key) [15], the typing speed, the pressure applied on the keys, finger positions on the keys [12], typing error and shift key use [16] and so on. Thus, the user’s typing patterns have neurophysiological factors that make them unique [12]. Therefore, this technique can be used for distinguishing among individuals.

This article proposes a CAPTCHA system against human-based attacks by exploiting the keystroke dynamics authentication system. This system is designed, implemented and evaluated through an experimental study.

The contributions of this article are as follows: first, we developed a new methodology for detecting human-based attacks on text-based CAPTCHAs using the keystroke dynamics approach. We then applied three time features to identify the typing patterns of the user. Next, we used the Euclidean distance as a classification method to measure similarities between users’ attempts to detect human attackers. These three time features have not been used before to detect human-based attacks. Finally, we conducted an experimental study to demonstrate the effectiveness of the proposed system.

The rest of the article is organised as follows: Section 2 reviews the related works. Section 3 describes the proposed system through a general abstract model. The methodology is explained in Section 4. Section 5 explains the experimental studies. Section 6 presents the results, whereas Section 7 discusses them. Finally, Section 8 concludes the article.

2 RELATED WORKS

This section reviews related studies for protecting CAPTCHAs in which the CAPTCHAs are redirected to third-party users to help attackers [11,17–22].

The Drag-n-Drop Interactive Masking CAPTCHA designed in Bin Ye et al. [17] applied drag and drop, interaction, and masking techniques to text-based CAPTCHA to differentiate legitimate users from computer bots and third-party human users. These techniques made their CAPTCHA system able to resist traditional attacks with a success rate of 64% for a CAPTCHA-length 5, 9% for a CAPTCHA-length 6 and 1% for a CAPTCHA-length 7. However, they did not test their CAPTCHA against third-party human attacks.

In Wei et al. [18], a GeoCAPTCHA is proposed to defend against third-party human attack by combining personalised information with an image-based CAPTCHA. The answer is known only by the user, so it can successfully prevent human attacks. A user has to remember a geolocation street view scene image that is preselected by him and preregistered with the server. To pass the test, the user has to rotate a given street view challenge to match the preregistered scene image. This will determine whether the user is a third-party solver.

Payal and Challa [19] proposed a new CAPTCHA scheme, AJIGJAX CAPTCHA, for websites that need high security and the requirement to authenticate a user. It also defends against third-party human attack. AJIGJAX CAPTCHA is a drag and drop–based CAPTCHA in the form of a linear jigsaw puzzle. In this type of CAPTCHA, the user needs to follow two phases: registration and login. In registration, the user uploads an image and then creates a pass-point on that image and types a label string using a virtual keyboard. In login, the user has to click on the right pass-point of the uploaded image at the registration phase (limited with three chances). If the user clicks on the right pass-point, the pictures for each character in the labelled string are formed and presented in a linear jigsaw puzzle manner in a random fashion that forms a dynamic CAPTCHA test for the user. The user is required to remember the label and solve the puzzle by drag and drop functionality using a mouse. The CAPTCHA is implemented using jQuery, Asynchronous JavaScript and XML (AJAX) and Java Server Pages. The proposed AJIGJAX CAPTCHA is expected to prevent third-party human attack because an illegal human user cannot solve this type of CAPTCHA even if he or she clicks on the right pass-point. Only a legitimate human user can solve the CAPTCHA, which is a kind of password that only the authorised user knows.

In Nanglae and Bhattacharjy [20], the authors proposed a method to identify an authorised user from an unauthorised user using a combination of the attributes of text-based CAPTCHA and human capabilities, called a biodetection function (BDF). They conducted an experiment to prove that the BDF for each person using a CAPTCHA is different. They developed a CAPTCHA system running on Adobe Flash CS5.5, written in PHP. First, the user is asked to fill in his or her demographic information (age, sex, occupation, eye problems, eyesight and any colour blindness). This information is used to indicate the user characteristics under the CAPTCHA tests. Then, the user has to solve a simple text-based CAPTCHA that contains four random digits and randomly selected colours of font and background. The
authors used the input time for the user to type the CAPTCHA solution as the key point index for BDF. They had 100 participants in their experiment. Their method of analysis was the chi-square test with a 95% confidence interval ($\alpha = 0.05$). The result showed that there is some limitation in human BDF under the text-based CAPTCHA mechanisms depending on the colours, position and shape of the number used. Thus, CAPTCHA can be used to indicate a personal characteristic of the user by considering all combinations of colours, shapes, positions and typing time. This proposed method can also be used to protect a text-based CAPTCHA from third-party human attack.

The study by Truong et al. [11] proposed an interactive CAPTCHA (iCAPTCHA), which is a text-based CAPTCHA to defend against third-party human attacks. First, they developed a streamlined human-based CAPTCHA attack that uses Instant Messenger (IM) infrastructure to clarify the threat of the human solver attacks. This attack allows iCAPTCHA challenges to be delivered to third-party human solvers by IM technology at speeds that defy detection by CAPTCHA timeout values (30 s) that are used as a defence against third-party human attacks in many CAPTCHAs. Finally, they proposed a defence system called iCAPTCHA, which involves a series of user interactions that require a user to enter each character individually with mouse clicks on the displayed characters' buttons until all characters are recognised. The multistep back-and-forth traffic between client and server amplifies the statistical timing difference between a legitimate user and a human solver, which enables them to detect third-party human attacks.

Finally, two research papers use behavioural biometrics integrated with CAPTCHA systems to be more secure. The first, in Souza [21], used an image-based CAPTCHA integrated with Mouse Dynamics, which is one of the behavioural biometrics. Mouse Dynamics monitors the mouse interaction of a human user. To solve the challenge, the user must identify and select a certain class of images. While the user tries to solve the CAPTCHA, the way each user interacts with the mouse (i.e. mouse clicks, mouse movements, mouse cursor screen coordinates) is recorded. These recorded mouse movements form the mouse dynamics signature (MDS) of the user. This MDS is an extra secure technique to differentiate humans from bots. The authors tested the security of the CAPTCHA by having an adversary execute a mouse bot attempting to solve the CAPTCHA challenges. They observed that their linear support vector machine classifier performed well in detecting the bot with 100% accuracy, whereas it had an accuracy of close to 86% in detecting humans attempting to solve the CAPTCHA samples.

The second study, in Al-naymat et al. [22], proposed merging keystroke dynamics with text-based CAPTCHA for the identification process of differentiating humans from computers (bots) to enhance the security of the CAPTCHA and protect it from automated or bot attack. They conducted an experiment to examine three cases of employing keystroke dynamics in a text-based CAPTCHA. In the first case, they assumed that the bots did not know about using keystroke dynamics in the text-based CAPTCHA. Thus, the classification result between humans and bots would be 100% correct. The software can distinguish between human and bot by checking whether the time features exist. If they exist, the CAPTCHA solver is human; if they do not exist, the CAPTCHA solver is the bot. That is, if a human solves the CAPTCHA, it will send the time features with the text answer to the server; on the other hand, if a bot solves the CAPTCHA, it will send only the text without the time features. Because of this, the authors did not conduct an experiment for this case. In the second case, they assumed that the bot knew about the time features sent with the CAPTCHA text and the number of features by attacking the code of the website. Thus, the bots would try to generate random numbers for time features from any keystroke data set and send them with the text answer to be similar to the human answer. To imitate this case, they created 20 fake users and generated random numbers for their time features; they also randomly chose 20 users from their data set, which was generated from their other experiment in which they studied the use of keystroke dynamics with a short password containing 56 users, nine records for each person and seven time features to represent the typing style of a user. They then divided the data set into a training set and testing set. In the second case, they assumed that the bot was smarter than in the second case and knew everything about the time features and their calculation methods. To imitate this case, they selected the same 20 users as in the second case, created 20 fake users, generated random numbers for only two main time features (hold time and flight time) and calculated the other time features based on the two main features. They then divided the data set into a training set and testing set. They used the WEKA tool to run their experiment and used three classifiers: Multi-Layer Perceptron (MLP), Random Forest and J48. They used accuracy to evaluate the data. For the case two data set, the accuracy results achieved for the Random Forest classifier and J48 classifier were 98.13%, which was higher than the 95% achieved by the MLP classifier. For the case three data set, the accuracy result achieved for the Random Forest classifier was 93.13%, which was the best accuracy result in case three, and the result achieved by the MLP classifier was 90.63%, which was higher than the 88.75% achieved by the J48. Their accuracy result was great, proving the effectiveness of their method to secure text-based CAPTCHA and protect it from bot attacks. This research in Al-naymat et al. [22] used keystroke dynamics with text-based CAPTCHA to detect an automated or bot attack, whereas our proposal is to detect a third-party human attack on text-based CAPTCHA to enhance its security.

Therefore, a CAPTCHA system still needs considerable research and enhancement to defend against human-based attacks to be more secure and usable. Moreover, to the best of our knowledge, there has not been a study on differentiating between a legitimate user and a human attacker of text-based CAPTCHA using keystroke dynamics (behavioural biometrics). Therefore, this article investigates the application of this approach by developing a text-based CAPTCHA that includes keystroke dynamics as one system.
3 | ABSTRACT SYSTEM MODEL

To define the proposed system under consideration precisely, we start by providing an abstract system model of the attack steps, involving a text-based CAPTCHA to be redirected to third-party users to help attackers, as shown in Figure 1. This model provides the basis of the experimental study that we present in Section 5.

In this system model, (1) a text-based CAPTCHA is shown on the website; (2) then, an attacker targets this website using human-based attackers. For this, this CAPTCHA is redirected by the attacker to human workers (3) to solve it. (4) Once the redirected CAPTCHA is solved by the workers, the solution as well as the keystroke timing data will be submitted to the targeted website. Accordingly, this attack fulfils the definition of CAPTCHAs, as introduced in von Ahn et al. [5], that it is a programme that generates tests that humans can pass easily, whereas computers cannot. For this, there is a need to detect not only the automated attack, but also the human-based attacks. Thus, the proposed system in this article takes advantage of keystroke timing data to detect human-based attacks. The following section details the detection methodology.

4 | METHODOLOGY

In this section, we describe our proposed system and how it can detect human-based attacks. As mentioned earlier, the system includes two subsystems: a text-based CAPTCHA generator and keystroke dynamics. Each will be explained in detail in the following subsections.

4.1 | CAPTCHA generator

We have developed a CAPTCHA generator that can generate a word containing English characters and/or Arabic digits (A–Z, a–z, and 1–9) randomly selected, as suggested in Alsuhibany [23]. According to the study by Killourhy and Maxion [24], which investigated research on anomaly detection using keystroke dynamics authentication, using 10 characters as the text length for keystroke dynamics authentication is typical. Based on this, we have selected this length for the keystroke dynamics authentication applied in our system. Finally, the CAPTCHA will be displayed to the user on a web page. For each request from the user, the generator generates a new CAPTCHA. Figure 2 shows a generated sample.

We have not applied anti-segmentation techniques in the developed generator, such as crowding characters together; instead, we added space between characters. This is only to demonstrate the effectiveness of the proposed system and approve the concept of the proposed idea. However, the antisegmentation features will be investigated and applied in our future research.

Character confusion is one of the common usability problems in a CAPTCHA [25]. We observed some character confusion when we conducted the primary experiment. Therefore, we removed the following characters from the character set of the generator: I (capital i) and l (small L), because they may confuse the user and affect CAPTCHA usability; the number 0 was also removed because the user may confuse it with the character O.

4.2 | Keystroke dynamics system for detecting human-based CAPTCHA attacks

This section highlights the keystroke dynamics system and the proposed detection methodology.

4.2.1 | Keystroke dynamics

We have exploited the keystroke dynamics authentication system to detect human-based CAPTCHA attacks for the following advantages [26]. First, a user's typing patterns are unique, and it is hard to reproduce because the keystroke dynamics system can calculate the keystroke action up to milliseconds precision. Also, the typing patterns cannot be lost or shared. Furthermore, it is easy to integrate with other existing systems and no extra hardware is required. In addition, it is considered the least expensive biometric authentication method because it is a software-based technique and requires only a keyboard, which is a necessary part of every computer [22]. Finally, it is transparent and nonintrusive, because the user's typing pattern on a keyboard can be collected without the knowledge of the user. Therefore, there is no need for users to change their behaviour and no need for them to do additional work to be authenticated [13].
The applied keystroke dynamics authentication system consists of two phases: enrolment and verification. For the enrolment phase, the user is asked to solve the CAPTCHA test several times. Specifically, the user types the CAPTCHA word that appears in the image. Meanwhile, the system collects the raw keystroke timing data (i.e. the press and release timestamp of each key typed to solve the CAPTCHA) and extracts the timing features that represent the typing patterns of each user. For the verification phase, the keystroke dynamics system tries to detect the human attacker by verifying whether the previous CAPTCHA tests were solved by the same user who has the same IP. This phase starts when the user reaches attempt number 100 to solve the CAPTCHA. We have decided that the verification phase relies on 100 attempts, which is inspired by the detection method of spam email for the email server of this study [27]. The authors of this study [27] specified the spam threshold to be 100 in their spam filter. Furthermore, they were able to detect spam email messages when a message was transferred more than 100 times. In addition to this study, the work of Motoyama et al. [8] stated that a real human attacker solves 1000 CAPTCHA tests in an hour. However, because the participants in our experiment were not real attackers, they were a little bit slower and hence were expected to solve 100 tests in approximately 1 hour.

Furthermore, there are four stages in the applied keystroke dynamics authentication system: data collection, feature extraction and file creation, classification method, and evaluation. In the following, each component is discussed in detail (except for evaluation, which will be discussed in Section 7). For data collection, which is the first step of our keystroke dynamics detection system, we have decided to develop a web application that generates a CAPTCHA and asks the user to solve and type it in a text box. The system collects the raw keystroke data while the user is typing the generated CAPTCHA. We used the generated CAPTCHAs as input text for our keystroke dynamics system. It is a free text that is not predefined and is randomly generated at the user's request. The collected keystroke timing data will be used to extract features. For the feature extraction and file creation, the collected keystroke times are used, and the time differences among them measured to generate time features that would be used to authenticate the user. These features are calculated from the key press timestamp and key release timestamp in milliseconds for each key typed. Furthermore, we have used three di-graph timing features, as suggested in Alsuhibany et al. [28]: the first is the Hold time or dwell time, which is the time period from a key press until it is released: for example, hold time for key 1 = release time of key 1 − press time of key 1. The second and third are represented by the flight time (or latencies), which has two types: Down–Down (DD) (or Press–Press) and Up–Down (UD) (or Release–Press). The former is the time difference between a key press and the press of the next key: for example, DD time = press time of key 2 − press time of key 1. The latter is the time difference between a key release and the press of the next key: for example UD time = press time of key 2 − release time of key 1. Figure 3 shows the pseudocode of acquiring these timing features.

Figure 4 shows an example of extracting these time features for two keys. Based on this example, the hold time for key K = 450 − 000 = 450 ms (milliseconds), DD time (for K and M keys) = 600 − 000 = 600 ms and UD (or Release–Press) time = 600 − 450 = 150 ms.

Afterward, the system computes the average for each time feature, that is the DD, UD and Hold times, to form a timing vector that will be stored in the database as the user's profile to be used for classification, as suggested in Alsuhibany et al. [28]. Figure 5 representing one of the user's profiles.

### 4.2.2 Proposed detection method

In our proposed detection method, we need to compare user data only with the data of the user himself or herself to verify that the user is the same as the one who solved the CAPTCHA tests more than 99 times to attack the system. Therefore, we proposed using the IP address as an identifier for the user data to differentiate between users, and using keystroke dynamics verification to detect human attackers.

---

**Figure 3** The pseudocode of acquiring timing features
We were inspired by Bursztein and Bethard [29], because they used the IP address to reduce attacks by limiting the number of attempts to solve the CAPTCHA for each IP address. In our research, we assumed that the user does not change his or her IP address.

To find the distance and detection method, the Euclidean distance, which is a commonly used distance metric, is used as a classification method for user profiles of typing patterns, owing to its simplicity and effective performance, as demonstrated in several studies [28,30]. Moreover, we computed the Euclidean distance to find how close the user's test data is to the user's profile for detecting a human-based CAPTCHA attack. In particular, we computed the distance between two time vectors: one reference time vector obtained previously (i.e. the user's profile stored in the database) and a new test time vector, using the Euclidean distance equation in three-dimensional space [31]:

$$d(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + (a_3 - b_3)^2}$$ (1)

where the two time vectors are $a$ and $b$. In our research, the user's test vector is represented by the letter "a" and the user's profile vector is represented by the letter "b". Furthermore, $d(a,b)$ is equal to or greater than 0 (i.e. $d(a,b) \geq 0$) [31].

To find a threshold for the Euclidean distance to classify the data and identify the users, the standard deviation (SD) of a user's profile vector is selected, as used in Alsuhibany et al. [28]. Also, the SD is commonly used to measure the variability or diversity of a set of values [32]:

$$SD = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N-1}}$$ (2)

where $x_i = \{x_1, x_2, \ldots\}$ represents the values of the time features, which are \{mean of hold time, mean of UD time and
mean of DD time) and \( \bar{x} \) represents the mean value of these three features. Also, \( N \) represents the number of items, which is three time features [32]. Table 1 shows the determined thresholds in our system derived in the primary experiment, as detailed in Section 5.1.

The proposed system computes the Euclidean distance of a new test vector with all stored time vectors (the user profiles) that have the same IP address. It then compares the computed distance value with the specified threshold, which is the SD of the user’s profile that is stored in the database. If it is less than the threshold, the test vector belongs to the same user of the compared profile (i.e. the test vector is similar to the user’s profile). If matching occurs 86 or more times, which is an empirical similarity threshold, it will classify the user as an attacker and the system will block him or her. Otherwise, it will consider him or her to be a legitimate user and grant access to the system. To illustrate this visibly, Figure 6 shows a flowchart of the proposed system.

This empirical similarity threshold (i.e. 86) may be increased when the experiment is released in real life in which real attackers are involved, and this might need more investigation, which is the subject of our future research.

5 | EXPERIMENTAL STUDIES

This section explains the conducted experiments to evaluate our proposed system.

5.1 | Primary experiment

We ran a controlled laboratory experiment with six participants to test the implemented system initially, as well as to determine the similarity threshold for detecting the attacker to be used in the real experiment. All the participants were students from Qassim University; their ages ranged from 19 to 22 years.

The participants acted as human attackers. They were asked to solve the CAPTCHA test by recognising the characters shown in the image, writing them inside the text box, and pressing the submit button once they were finished. They were asked to solve the CAPTCHA tests more than 100 times for 1 hour, until they were detected as attackers. In this experiment, we initially set the similarity threshold at 70, which was randomly selected. The system assigned a fake IP address to each participant as we conducted the experiment on one laptop. Table 2 illustrates the details of this experiment for each participant. The table contains the IP address of each participant, the number of solved CAPTCHA tests and the number of similar profiles for each participant when they were detected as attackers and blocked.

After conducting this experiment, the system was working properly and was able to detect all attackers. Interestingly, we observed that the minimum number of similar profiles when attackers were detected was 86. Hence, we modified the similarity threshold to be 86 for the real experiment.

![Figure 6](image-url)

**Figure 6** Flowchart of the proposed system

| TABLE 1 | Determined thresholds in our system |
|----------|--------------------------------------|
| Threshold name | Threshold value | Description |
| Verification starting | 100 | Verification phase starts at attempt number 100 |
| Standard deviation | (Computed for extracted time features) | Threshold for classification of user profiles |
| Similarity threshold | 86 | Number of similar user profiles must be 86 or greater to classify user as attacker |
Each targeted system had 30 participants. The participants in the attackers' group were instructed to solve the CAPTCHA by typing the characters that appeared in the image in the text box using normal typing for any text on the computer, and then to press the submit button when they were finished. Participants were also given the following guidelines: type the CAPTCHA characters without the space shown in the image, and the case of the characters did not matter, so they could solve it with either an uppercase or lowercase letter.

Participants in the attackers' group were instructed to solve the CAPTCHA test more than 100 times in an hour until the system blocked them and displayed the block page, whereas participants in the legitimate users' group were instructed to solve the CAPTCHA test once or more but without exceeding 99 times in an hour. When the attacker is detected, the system records the number of CAPTCHA tests solved by him or her and the number of similar profiles for each attacker.

Considering the modifications given by the preliminary experiment, a controlled laboratory experiment was conducted that aimed to investigate whether our proposed system could detect human-based attacks. The experiment setup and procedure are described in the following sections.

### 5.2 Real experiment

This explains the way in which we conducted the experiment, including instructions for the participants and the procedure. The participants were instructed that the goal of the experiment is to test whether our proposed system is able to detect human attackers by differentiating between a legitimate user and an attacker through keystroke dynamics. The participants were instructed to solve the CAPTCHA by typing the characters that appeared in the image in the text box using normal typing for any text on the computer, and then to press the submit button when they were finished. Participants were also given the following guidelines: type the CAPTCHA characters without the space shown in the image, and the case of the characters did not matter, so they could solve it with either an uppercase or lowercase letter.

Participants in the attackers' group were instructed to solve the CAPTCHA test more than 100 times in an hour until the system blocked them and displayed the block page, whereas participants in the legitimate users' group were instructed to solve the CAPTCHA test once or more but without exceeding 99 times in an hour. When the attacker is detected, the system records the number of CAPTCHA tests solved by him or her and the number of similar profiles for each attacker.

### 5.2.1 Experiment setup

The experimental design, participants, system and collected data are as follows. For the experiment design, we used a between-subjects design in which participants are divided into two groups: legitimate users and attackers, with each group solving a different number of CAPTCHA tests. This type of design ensures that the exact samples of CAPTCHAs are used in each experiment condition, and there is no unnecessary confounding factor biasing the results (at the cost of recruiting a relatively large number of participants). The legitimate users' group was asked to solve the CAPTCHA test once or more without exceeding 99 times in an hour. The attackers' group was asked to solve the CAPTCHA test 100 times or more until the system detected them as attackers. Moreover, the experiment was conducted in a controlled environment in which we set an hour for each attacker to solve the 100 CAPTCHA tests. We assumed that malicious entities redirect the entire webpage to the human attacker to solve a CAPTCHA test several times to attack the targeted system.

For the participants, we recruited 60 users who participated in this experiment over 5 weeks. Participants were randomly divided into two groups: legitimate users and attackers. Each group had 30 participants. The participants in the attackers' group solved the CAPTCHAs as human attackers and the participants in the legitimate users' group solved the CAPTCHAs as normal users. For both groups, the ages ranged from 18 to 35 years. Most participants had a technical background and varying levels of typing skill.

We developed our proposed system as a Java web application by using the Java servlet, HTML, CSS, JavaScript and jQuery, AJAX and MySQL database. We tested the system on a Windows laptop. The application is hosted by a local server, Apache Tomcat. The system assigned a fake IP address to each participant because we conducted the experiment on one laptop. The IP addresses of the attackers ranged from 192.168.0.1 to 192.168.0.30, whereas the IP addresses of the legitimate users ranged from 127.0.0.2 to 127.0.0.31, randomly selected. However, the real IP address for each user will be detected when we test our proposed system in an online environment.

For the collected data, each successful attempt by the participant is stored by the system and includes the keystroke times in milliseconds, the time features, the time vector and the SD with the IP address in the database. The system also stores the blocked IPs in the database. Furthermore, it records the time taken for each participating attacker to solve the CAPTCHA tests until the system detects him or her. When the attacker is detected, the system records the number of CAPTCHA tests solved by him or her and the number of similar profiles for each attacker.

### 5.2.2 Experiment procedure

This explains the way in which we conducted the experiment, including instructions for the participants and the procedure. The participants were instructed that the goal of the experiment is to test whether our proposed system is able to detect human attackers by differentiating between a legitimate user and an attacker through keystroke dynamics. The participants were instructed to solve the CAPTCHA by typing the characters that appeared in the image in the text box using normal typing for any text on the computer, and then to press the submit button when they were finished. Participants were also given the following guidelines: type the CAPTCHA characters without the space shown in the image, and the case of the characters did not matter, so they could solve it with either an uppercase or lowercase letter.

Participants in the attackers' group were instructed to solve the CAPTCHA test more than 100 times in an hour until the system blocked them and displayed the block page, whereas participants in the legitimate users' group were instructed to solve the CAPTCHA test once or more but without exceeding 99 times in an hour. Both groups were instructed that we did not collect personal information, we used a fake IP address to identify each participant, and we collected the keystroke times only when they typed the CAPTCHA answer.

For the procedure, the experiment was conducted in a controlled laboratory environment to make sure that each participant did not exceed 1 hour when solving the CAPTCHA tests. Each participant was asked to solve the CAPTCHA test by typing what was displayed in the image in the text box, and to press the submit button.

| IP address   | Number of solved CAPTCHAs | Number of similar profiles |
|--------------|---------------------------|----------------------------|
| 192.168.0.1  | 106                       | 94                         |
| 192.168.0.2  | 101                       | 96                         |
| 192.168.0.3  | 100                       | 91                         |
| 192.168.0.4  | 100                       | 94                         |
| 192.168.0.5  | 100                       | 86                         |
| 192.168.0.6  | 101                       | 93                         |

### Table 2: Details of the primary experiment
6 | RESULTS

All participants successfully completed their task. The CAPTCHA was solved 3212 times. Based on this, we collected 3182 time vectors (i.e. user profiles) for 60 participants, 86 of which belonged to legitimate users and 30% of which belonged to the attackers. Although there were actually 3126 attacker profiles in total, only 30% were recorded, because when an attacker was detected, we stored only the blocked IP address (i.e. we did not store the typing data of the last attempt for each attacker). The details of these results are explained in the following sections.

6.1 | Number of Completely Automated Public Turing Tests to Tell Computers and Humans Apart solved by legitimate users

As shown in Figure 7, the minimum number of solved CAPTCHAs by legitimate users was one and the maximum was nine.

6.2 | Average of hold time, up–down time and down–down time for each legitimate user

The results of the averages of hold time, UD time and DD time for each legitimate user are shown in Figure 8.

![Figure 7](image_url)

**FIGURE 7** Number of Completely Automated Public Turing Tests to Tell Computers and Humans Apart solved by each legitimate user

![Figure 8](image_url)

**FIGURE 8** Averages of hold time, up–down time and down–down time for each legitimate user

6.3 | Number of Completely Automated Public Turing Tests to Tell Computers and Humans Apart solved by each attacker until the system detects him or her as an attacker

The results for the number of CAPTCHAs solved by each attacker until the system detects him or her as an attacker are shown in Figure 9. The red line indicates the attackers who were detected by the system when they solved 100 CAPTCHA tests.

6.4 | Time taken for each attacker to be detected by the system

The results for the time taken for each attacker to be detected by the system are shown in Figure 10. This time was recorded in minutes and should not exceed 1 hour for each attacker.

6.5 | Number of CAPTCHA tests solved by each attacker versus the number of similar profiles for each attacker

Figure 11 shows the results for the number of CAPTCHA tests solved by each attacker against the number of similar profiles for each attacker when the system detected him or her as an attacker.

6.6 | Average standard deviation for all successful attempts for each attacker

The results for the average SD for all the successful attempts for each attacker are shown in Figure 12.

6.7 | Averages of hold time, up–down time and down–down time for all successful attempts for each attacker

Figure 13 shows the results for the averages of hold time, UD time and DD time for all successful attempts for each attacker.
FIGURE 9  Number of Completely Automated Public Turing Tests to Tell Computers and Humans Apart solved by each attacker until the system detects him or her as an attacker.

FIGURE 10  Results of time taken for each attacker to be detected by the system.

FIGURE 11  Results of number of Completely Automated Public Turing Tests to Tell Computers and Humans Apart solved by each attacker versus the number of similar profiles for each attacker.
7 | DISCUSSION

The results demonstrate that the proposed system was able to detect all attackers when they solved the CAPTCHA test more than 99 times, as shown in Figures 9 and 11, without exceeding 1 hour, as shown in Figure 9. Also, Figure 7 shows that the number of CAPTCHA tests that were successfully solved by legitimate users within 1 hour did not exceed nine attempts. No legitimate user solved more than 99 CAPTCHA tests in 1 hour. Consequently, the detection rate of the proposed system is 100%. However, this detection rate might vary when the system is run in real life.

Moreover, as shown in Figures 9 and 11, the system detected 11 attackers at attempt number 100, whereas the others exceeded 100 attempts. This may be because they solved the CAPTCHA tests continuously without stopping. Also, they were concentrating while typing the solutions, with no distractions such as talking to a friend or using the phone.

Interestingly, Figure 11 shows that some attackers were detected when the number of attempts exceeded 100. The
system did not detect them when they were at attempt number 100 of solving the CAPTCHA. A possible explanation for this is that because the similarity of user profiles did not reach the determined threshold of more than 85 similar profiles, the system allowed them to solve more CAPTCHAs; once they reached the similarity threshold, the system detected them as attackers and blocked them.

Furthermore, there were some variations in attacker profiles in Figure 11 owing to distractions while the attacker was typing the CAPTCHA answer. That is, the typing patterns of the attacker were affected and hence changed. For example, attacker 21, who had the maximum number of solved CAPTCHAs (128), as shown in Figure 11, was listening to a podcast while typing the CAPTCHA answers. As for the other attackers who exceeded 100 attempts and were not detected, they were most likely talking to their friends while typing the CAPTCHA answers. However, all attackers were detected by our system with a different number of attempts (i.e., 100–128) to solve the CAPTCHAs within a time not exceeding 1 hour, as shown in Figure 10. However, when the system is run in real life, we can add a limit to the maximum number of attempts to solve the CAPTCHA tests for each IP address per hour, day or month, according to the security level required by the system.

In addition, the time taken for all attackers to complete their task did not exceed 1 hour, as shown in Figure 10. Specifically, the time taken to solve the CAPTCHA tests varied owing to the different typing skills. The minimum time was 20 min and the maximum time was 45 min. The time variation was according to users’ familiarity with typing in the English language. However, we conclude that the average time taken by all participating attackers is 33 min.

Figures 8 and 13 proved that the typing pattern is unique for each person. In addition, we observed no duplicate values in time features in these figures. This allows us to detect attackers according to their style of typing through these three time features (i.e., hold time, DD time and UD time) for all characters’ keys that were typed to solve the CAPTCHA test. Moreover, Figure 12 shows the average SD for all successful attempts to solve the CAPTCHA tests for each attacker; we can observe that the SD values for each person are different, which also helps distinguish users. This also proves that keystroke dynamics authentication with the three time features could be successfully used with CAPTCHA as an approach to detecting human-based attacks that has not been previously proven.

Furthermore, we measured the detection rate of attackers and the two error rates, which are the false acceptance rate (FAR) and false rejection rate (FRR) [33]. Therefore, we have achieved 0% FRR, which represents the percentage of attempts in which the system incorrectly rejects a legitimate user, and 3.58% FAR, which represents the percentage of attempts in which the system incorrectly accepts the attacker as a legitimate user. Our system accepted 112 of 3126 attackers’ attempts. This may be due to some distractions for the attackers while solving the CAPTCHAs, as discussed previously. We could have a lower FAR if we apply the experiment in a more controlled environment in which we restrict participants to avoid distractions. However, these results indicate that the performance of our system is outstanding and confirm that our proposed system is effective in terms of detecting human attackers.

We compared our proposed system with that of Truong [11], which proposed the iCAPTCHA system to defend against human-based attacks, as mentioned in Section 2. It requires a user to solve a text-based CAPTCHA by a series of user interactions. The author recorded the iCAPTCHA solving time on a per-character basis. The researcher then developed two detection algorithms to detect human attackers based on per-character response times. The first algorithm is the single slow response detection algorithm. It compares the per-character response times with a predetermined threshold D, which is the average of a user decode time plus the round trip time between the server and the user. If there are one or more per-character response times higher than D, it will reject that iCAPTCHA test response. The second algorithm is the two consecutive slow responses detection algorithm. It rejects an iCAPTCHA test that has only two consecutive per-character response times higher than the threshold D. There were 63 legitimate users and two human attackers participating in that study. The legitimate users solved the iCAPTCHA 226 times and the attackers also solved the iCAPTCHA 226 times. The two detection algorithms detected all human attackers, resulting in a 100% detection rate. However, the first algorithm achieved a 0% FAR error rate and a 10.17% FRR error rate; that is, the system rejected only 23 correct responses from 226 responses by legitimate users. Moreover, the second algorithm achieved a 0% FAR error rate and a 1.77% FRR error rate; that is, the system rejected only four correct responses from 226 responses from legitimate users.

However, that detection system is based on the statistical timing difference while solving the iCAPTCHA between a legitimate user and a third-party human attacker. Therefore, users whose first language is not English may take a long time to recognise the characters, so they will have a slow response time. Consequently, they will be detected as human attackers and rejected. Our system overcomes this issue by using keystroke dynamics authentication to detect human-based attacks. However, Table 3 compares our proposed system with the iCAPTCHA system to detect human-based attacks on text-based CAPTCHA according to the detection rate of the human attackers; two error rates, FAR and FRR; the number of participants; and the number of solved CAPTCHA tests in the experiment. Although the detection rate was the same, our FRR is better. Moreover, if we consider the number of attackers who have exceeded the threshold (i.e., 100), as shown in Figure 9, the FAR of our result is 3.58%. Nevertheless, it may be that these attackers have not been detected, which did not occur in our system because we detected all attack attempts.

Furthermore, Wei et al. [14] and Palay and Challa [19] proposed systems to defend against third-party human attacks that rely on information identified by the legitimate user during the registration. Therefore, the user should remember this information during the login to confirm that he or she is a legitimate user. Unfortunately, this is difficult for the human
TABLE 3 Comparison of our proposed system’s results and the results of Rudrapal et al. [15]

|                  | [15]* | [15]† | Current study |
|------------------|-------|-------|---------------|
| Detection rate   | 100%  | 100%  | 100%          |
| FRR              | 10.1% | 1.7%  | 0.0%          |
| FAR              | 0.0%  | 0.0%  | 3.58%         |
| No. of attackers | 2     | 2     | 30            |
| No. of legitimate users | 63   | 63   | 30            |
| No. of solved CAPTCHAs | 452 | 452   | 3212          |

Abbreviations: CAPTCHA, Completely Automated Public Turing Test to Tell Computers and Humans Apart; FAR, False acceptance rate; FRR, False rejection rate.
*Indicates the first detection algorithm of interactive CAPTCHA [15].
†Indicates the second detection algorithm of interactive CAPTCHA [15].

mind, which has a lot of information to remember, so it is likely the user may forget the information he or she specified during registration. Therefore, when the user forgets it, he or she might be classified as an attacker and be prevented from accessing the system. However, our system overcomes this weakness by not remembering any information; it requires the user only to type the CAPTCHA as usual with no additional task. Hence, our system can detect human-based attacks by detecting attackers using keystroke dynamics. Furthermore, studies [18,19] used an image-based CAPTCHA [18] and a puzzle-based CAPTCHA [19], whereas we have used the most common CAPTCHA type, the text-based CAPTCHA. Thus, our results cannot be comparable.

Moreover, we cannot compare our results with the proposed methods of Bin Ye et al. [13] and Nanglae and Bhattacharokosol [20] to defend against third-party human attackers, because neither study conducted experiments on human-based attacks.

We have designed a CAPTCHA that is breakable by automated attacks, because this design of text-based CAPTCHA is only for proving the proposed concept, which is detecting human-based attacks through using keystroke dynamics. Nevertheless, we plan to design a scheme that is resistant to automated attacks as well as human-based ones. This will be a subject of our future research.

8 CONCLUSION

A text-based CAPTCHA, by design, is unable to differentiate between a human-based attacker and a legitimate human user. Human-based attacks have been reported as a simpler, less costly and, more important, guaranteed way to bypass the security provided by CAPTCHAs, because the solvers are human, not automated codes. This makes CAPTCHAs vulnerable to human-based attacks.

This report thus proposed a novel methodology for detecting human-based attacks using keystroke dynamics. This methodology is based on timing features that can be extracted from the time lapses between two actions on the keyboard: are key press and key release. Specifically, we have used three time features, hold time, DD time and UP time, to identify user typing patterns to verify users, using the Euclidean distance. A laboratory experiment was conducted to evaluate the proposed methodology, and the results indicated that the proposed methodology can effectively detect human-based attacks.

In future work, we will conduct an online experiment with more participants to evaluate our proposed system in a real environment, to obtain more accurate results. Moreover, it would be interesting to investigate the possibility of including automated attack tools in the proposed system, to make it more secure and reliable for defending against automated attacks as well as human-based attacks.

ACKNOWLEDGMENTS

The authors would like to thank Qassim University for supporting this research.

ORCID

Suliman A. Alsuhibany © https://orcid.org/0000-0001-7735-9781
Latifah A. Alreshoodi © https://orcid.org/0000-0002-6473-915X

REFERENCES

1. Huang, S.Y., et al.: An efficient segmentation algorithm for CAPTCHAs with line cluttering and character warping. Multimed. Tools Appl. 48(2), 267–289 (2010).
2. Egele, M., et al.: CAPTCHA smuggling: hijacking web browsing sessions to create CAPTCHA farms. In: Proceedings of ACM Symposium on Applied Computing, 1865–1870 (2010).
3. Singh, V.P., Pal, P.: Survey of different types of CAPTCHA. Int. J. Comput. Sci. Inf. Technol. 5(2), 2242–2245 (2014).
4. Wang, X., et al.: Robustness of text-based completely automated public testing to tell computers and humans apart. IET Inf. Secur. 10(1), 45–52 (2016).
5. von Ahn, L., Blum, M., Langford, J.: Telling humans and computers apart automatically. Commun ACM. 47(2), 56–60 (2004).
6. De Marsico, M., et al.: FATCHA: biometrics lends tools for CAPTCHAs. Multimed. Tools Appl. 76(4), 5117–5140 (2017).
7. Mohamed, M., et al.: On the security and usability of dynamic cognitive game CAPTCHAs. J. Comput. Secur. 25(3), 205–230 (2017).
8. Motoyama, M., et al.: Re: Captchas – Understanding CAPTCHA-based Captchas and designing image-based Captchas. J. Comput. Sci. Inf. Technol. 5(2), 2242–2245 (2014).
9. Tang, M., et al.: Research on deep learning techniques in breaking text-based Captchas and designing image-based Captcha. IEEE Trans. Inf. Forensics Secur. 13(10), 2522–2537 (2018).
10. Wang, P., et al.: Simple and easy: transfer learning-based attacks to text CAPTCHA. IEEE Access. 8, 59044–59058 (2020).
11. Truong, H.D., et al.: The next generation of CAPTCHA designed to defend against 3rd party human attacks. In: IEEE International Conference on Communications (ICC), 1–6 (2011).
12. Ali, M.L., et al.: Keystone biometric systems for user authentication. J. Signal Process. Syst. 86(2–3), 175–190 (2017).
13. Rudrapal, D., Das, S., Saha, A.: Analysis of user keystroke dynamics and its application for preventing automated form filling programmes. In: International Conference on Computing Communication and Information Technology. 27, 176 (2012).
14. Alreshoodi, L., et al.: Authentication via keystroke dynamics. In: 4th ACM Conference on Computer & Communications Security, 48–56 (1997).
15. Rudrapal, D., Das, S., Saha, A.: Analysis of user keystroke dynamics and its application for preventing automated form filling programmes. In: International Conference on Computing Communication and Information Technology. 27, 176 (2012).
16. Alreshoodi, L., Warwick, K., Wei, H.: Non-conventional keystroke dynamics for user authentication. Pattern Recognit. Lett. 89, 53–59 (2017).
17. Bin Ye, Q., et al.: A novel drag-n-drop interactive masking CAPTCHA against the third party human attacks. In: Proceedings of 2013 Conference on Technologies and Applications of Artificial Intelligence, 158–163 (2013)
18. Wei, T.E., Jeng, A.B., Lee, H.M.: GeoCAPTCHA - a novel personalised CAPTCHA using geographic concept to defend against 3rd party human attack. In: 2012 IEEE 31st International Performance Computing and Communications Conference, 392–399 (2012)
19. Payal, N., Challa, R. K.: AJIGJAX: A Hybrid image based model for Captcha/CaRP. In: 2016 IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics Engineering (UPCON), 38–43 (2016)
20. Nangla, N., Bhattacharjee, P.: A study of human bio-detection function under text-based CAPTCHA system. In: Proceedings of 2012 IEEE/ACIS 11th International Conference on Computer and Information Science ICIS 2012, 139–144 (2012)
21. Souza, D.F.D.: Avatar captcha: telling computers and humans apart via face classification and mouse dynamics (2014)
22. Al-naymat, G., et al.: Dynamics-based approach for accurate user identification and authentication using machine learning techniques. In: International Conference on Information Technology and Applications (2016)
23. Alsuhibany, S.A.: A benchmark for designing usable and secure text-based Captchas. Int. J. Netw. Secur. Its Appl. 8(4), 41–54 (2016)
24. Killourhy, K.S., Maxion, R.A.: Comparing anomaly-detection algorithms for keystroke dynamics. In: Proceedings of International Conference on Dependable Systems & Networks, 125–134 (2009)
25. Alsuhibany, S.A.: Optimising CAPTCHA generation. In: Proceedings of 2014 6th International Conference on Availability, Reliability and Security ARES 2011, 740–745 (2011)
26. Teh, P.S., Troha, A.J., Yue, S.: A survey of keystroke dynamics biometrics. Sci. World J. 2013, 1–24 (2013)
27. Yoshida, K., et al.: Density-based spam detector. In: KDD-2004, Proceedings of Tenth ACM SIGKDD International Conference on Knowledge Discovery Data Mining, 486–493 (2004)
28. Alsuhibany, S.A., et al.: Analysis of free-Text keystroke dynamics for Arabic language using Euclidean distance. In: Proceedings of 2016 12th International Conference on Innovations in Information Technology IIT, 185–190 (2016)
29. Burshtein, E., Bethard, S.: Decaptcha: breaking 75% of eBay audio CAPTChas. In: 3rd USENIX workshop on Offensive Technologies WOOT, 1–7 (2009)
30. Singh, S., Arya, K.V.: Key classification: a new approach in free text keystroke authentication system. In: Proceedings of 2011 3rd Pacific-Asia Conference on Circuits, Communications and System, 1–5 (2011)
31. Waterman, G.: Euclidean space and vectors. In: Linear Algebra I, Oregon Institute of Technology, 43–59 (2007)
32. Rouaud, M.: Probability, statistics, and estimation: propagation of uncertainties in experimental measurement. In: Probability, statistics, and estimation, pp. 3–6 (2017)
33. Hassan, B. M., Fouad, K. M., Hassan, M. F.: Keystroke dynamics authentication in Cloud Computing: a Survey. Int. J. Enterp. Inf. Syst. 11(4), 99–120 (2015)

How to cite this article: Alsuhibany SA, Alreshoodi LA. Detecting human attacks on text-based CAPTChas using the keystroke dynamic approach. IET Inf. Secur. 2021;15:191–204. https://doi.org/10.1049/isc2.12018

APPENDICES
The following shows a screenshot of the developed web application of our system.

FIGURE 14  A screenshot of the developed website