Multimedia Systems (2025) 31:323
https://doi.org/10.1007/s00530-025-01897-0

ORIGINAL ARTICLE

CAPTCHA farm detection and user authentication via mouse-trajectory similarity measurement

Rui Jin[1] · **Yong Liao**[1]

Received: 12 November 2024 / Accepted: 3 June 2025 / Published online: 21 August 2025

Abstract
A well-known challenge Completely Automated Public Turing Test To Tell Computers and Humans Apart (CAPTCHA) faces is the CAPTCHA farm, where workers solve CAPTCHAs manually. In this work, we propose to tackle this challenge from a novel perspective. We convert CAPTCHA farm detection to identity inconsistency detection, which essentially becomes an authentication process. Specifically, we develop a novel embedding model, which measures the similarity between mouse trajectories collected during the session and when registering/solving CAPTCHA. Furthermore, we propose using diverse mouse movement data to implement enrollment sample selection and dynamic authentication, enhancing both security and flexibility during authentication. Experiment results validate the superiority of our method over the state-of-the-art time series classification methods and mouse-based authentication systems, achieving 96.9% and 99.1% of AUC in identity and authentication inconsistency detection, respectively. Moreover, unlike most existing works that employ a separate mouse movement classifier for each individual user, which brings in considerable costs when serving a large number of users, our model performs detection tasks using only one classifier for all users, significantly reducing the cost. These results indicate that our model, powered by a single classifier, performs exceptionally well and can detect inconsistencies in the identity of new users, making it a promising approach for detecting CAPTCHA farm attacks and authentication.

Keywords Mouse · CAPTCHA · Biometrics · Authentication

1 Introduction

Completely Automated Public Turing Test To Tell Computers and Humans Apart (CAPTCHA) aims to recognize and block automated programs by requiring the user to solve hard AI problems [44]. CAPTCHAs are widely used in registration to avoid fake accounts in authentication systems. However, CAPTCHAs have made solving CAPTCHAs a profitable business [34]. CAPTCHA farms are often used by programs that imitate human behavior to bypass CAPTCHAs. These programs may be used for activities such as booking tickets for scalpers or creating fake social media accounts. The attackers use them to lower their costs: employing workers in low-income areas to solve CAPTCHAs manually is much easier and cheaper compared to solving CAPTCHAs by the attackers themselves or by training new AIs. The cost of such attacks can be as low as 3$ per 1000 CAPTCHAs.[1] More importantly, existing CAPTCHA can not defend farming since CAPTCHAs are designed to be easy for humans. This issue has been largely overlooked so far. Although CAPTCHA providers have come up with utilizing browser fingerprinting such as the client's IP address, explorer information, hardware information, etc., since the CAPTCHA solvers can use virtual machines and proxies or remote control to provide identical software/hardware environment with the attacker, these mechanisms cannot stop CAPTCHA farms. Biometrics, such as mouse dynamics, keystroke behavior, and images of hands, have been used to construct new CAPTCHAs, but existing studies only attempted to detect automated bots [7, 11].

Communicated by Bin Xiao.

✉ Yong Liao
yliao@ustc.edu.cn

Rui Jin
jinrui1@mail.ustc.edu.cn

[1] School of Cyber Science and Technology, University of Science and Technology of China, Hefei 230026, Anhui, China

[1] https://2captcha.com/.

CAPTCHA farms will cause identity inconsistency in the session. Noticing that it is not feasible to detect CAPTCHA farms with existing CAPTCHAs and browser fingerprints, we propose to identify the users via their mouse trajectories and reckon the sessions in which CAPTCHA is solved by another person as CAPTCHA farm attacks. Despite some studies suggesting combining mouse- and keyboard-based biometrics, we utilize only mouse trajectories because most novel CAPTCHAs require the user to operate their mouse instead of keyboard [25, 29, 39]. Traditional biometrics such as fingerprint, iris, and face are collected only when needed for authentication or identification. On the other hand, mouse movements are captured whenever the user performs operations that involve the cursor, allowing for continuous verification of the user's identity throughout the session. Additionally, recording mouse behaviors does not require additional devices, making the cost of deployment relatively low.

In 2003, Everitt and McOwan made the first attempt to develop an authentication system using users' signatures drawn with a mouse [17]. Since then, researchers have been experimenting with mouse-based authentication using statistical methods, machine learning, and deep learning (DL). It's quite surprising that studies employing statistical analysis or manual feature engineering generally produce better results [26]. This could be because these methods are able to extract information within a time frame of typically 10 min to half an hour, which is too long for deep-learning approaches. Longer records tend to contain more information due to the diversity of mouse trajectories. However, a lengthy authentication process also poses security risks, as an attacker could complete their task before detection. To make it worse, training authentication classifiers is expensive. Many studies have transformed the authentication and identification problem into multiple One-vs-Rest classification tasks, training a separate classifier for each registered user. This approach increases processing power demands and results in an unbalanced dataset when scaling up. This hampers the application of mouse-based authentication on a larger scale. Although one-class SVM and multi-output deep learning have been introduced to tackle this issue, the results have been unsatisfactory. Furthermore, existing mouse-based authentication methods require training on the registered users before predicting whether the input comes from them. This means they can't be used to detect CAPTCHA farm attacks, as the users are unlikely to be included in the training dataset.

In this paper, we introduce a novel system that can be utilized for both CAPTCHA farm attack detection and user authentication. The proposed system comprises a preprocessing procedure, an embedding network, and two classifiers. The core idea is to accept pairs of mouse trajectories and assess their similarity in identity. This simplifies the problem into one binary classification task instead of multiple One-vs-Rest classification tasks. By using a single classifier, we can predict the similarity between two mouse trajectories, which significantly reduces both training and storage costs. The proposed system can process input from unseen users and be applied to both user authentication and CAPTCHA farm detection by converting them to identity inconsistency detection. We also propose two mechanisms to take advantage of the diversity of mouse trajectories, i.e., enrollment sample selection and dynamic authentication. Specifically speaking, we sample long input with a sliding window and evaluate the representativity of the samples to generate multiple different and representative sample pairs as input to improve the model's performance.

We notice that CAPTCHA farm detection and mouse-based authentication can share the same solution: measuring the identity similarity between two mouse trajectories. If the given mouse trajectories are matched by those recorded by the user it claims to be, it passes the authentication, and vice versa. Similarly, if the mouse trajectories recorded when solving the CAPTCHA have significant differences from those recorded later, we can predict that the CAPTCHA is solved by someone else. To build a model capable of calculating the similarity between two given mouse trajectories, we apply an embedding network to extract features from the input and train it using sample pairs. Furthermore, we propose to perform base sample selection and dynamic authentication to improve the model's performance. Our contributions include:

- We propose a novel system to measure the similarity of two mouse trajectories for both user authentication and CAPTCHA farm detection. To the best of our knowledge, this is the first study addressing the third-party CAPTCHA farm problem.
- The proposed system extracts features from mouse trajectories using the same model. This can lower the required number of classifiers and significantly decrease training and storage costs.
- We propose enrollment sample selection and dynamic authentication, which utilize the diversity of mouse trajectories and can significantly improve the model's performance.
- The proposed model is tested on a hybrid dataset of mouse movement records from 130 users in guided and unguided environments. The experiment results demonstrate our model's effectiveness, robustness, and advantages over the state-of-the-art time series classification approaches and mouse-based authentication methods.

The structure of this paper is as follows. Related studies will be reviewed in Sect. 2. The proposed system and the application details for authentication and CAPTCHA farm

detection will be introduced in Sect. 3. Section 4 presents the experiments and results.

2 Related works

2.1 Mouse-based authentication

Mouse-based biometrics are mainly studied for mouse-based authentication [26]. Since mouse behaviors are commonly generated continuously after the user passes the authentication, the researchers found mouse-based biometrics suitable for continuous authentication to prevent hijacking [21, 35, 41]. A data stream is assumed to be available, which allows for feature engineering over an extended period of time. Gamboa and Fred developed a game in which a user has to match a pair of tiles in a grid of tiles by clicking on them [20]. The authentication system records the user interaction during the game and obtains an equal error rate (EER) of 0.2%. Despite the high accuracy, the data collection requires 10–15 min. Ahmed and Issa analyzed 22 users' mouse behaviors during unguided sessions and found a false acceptance rate (FAR) of 2.4649% and a false rejection rate (FRR) of 2.4614% [2]. However, it takes 13.55 min on average to detect an identity mismatch. In another study, Zheng et al. used angle-based metrics and SVM for classification [46]. They achieved an EER of 1.3% with an average authentication time of 37.73 min. Additionally, Shen et al. proposed Mouse-Behavior Pattern Mining and obtained FAR of 0.09% and FRR of 1% on 5 min' frequent behavior segments [38].

An authentication process that takes 10 min to half an hour is considered too slow for traditional authentication tasks. A shorter authentication time also enables the use of deep learning. For example, Hu et al. utilized mouse dynamic behavior to achieve a FAR of 2.94%, a FRR of 2.28%, and an authentication time of 7 s when detecting insider threats using a Convolutional Neural Network (CNN) [22]. Additionally, the RUMBA-Mouse, a combined CNN-Recurrent Neural Network (RNN) model, produced a 3.16% EER with an average authentication delay of 6.11 s [18]. However, the methods mentioned above require training a classifier for each user. Shen et al. collected mouse movements from 37 participants in a tightly controlled environment. They achieved a FAR of 8.74% and an FRR of 7.69%, with an authentication time of 11.8 s [37]. It is worth noticing that these results were obtained by training a one-class Support Vector Machine (SVM) classifier to detect imposters instead of a separate classifier for each user. Similarly, 2D-CNN has been applied to avoid training multiple classifiers: the model has numerous binary outputs, each representing the prediction of the authentication results for a different user, resulting in an EER of 10% [10]. Overall, mouse-based authentication requires further improvement in the accuracy, authentication time, robustness, and cost of training to be practical.

Researchers have also tried to expand the data-collecting environments. The feasibility in unguided environments is most common and is usually achieved by capturing consecutive mouse events through segmentation [3, 4, 24]. Siddiqui et al. recorded mouse dynamics while 10 users played the video game Minecraft and achieved an average accuracy of 92% [40, 41]. Antal et al. developed a clicking game based on JavaScript to collect mouse dynamics [6]. Fu et al. proposed introducing angle offset to the standard mouse to combine human learning and machine learning [19].

2.2 Mouse-based CAPTCHAs

Most CAPTCHAs require the user to move their cursors. Chu et al. developed a client-side logger and a classifier based on the C4.5 algorithm to identify bots [12]. UNICAPTCHA, a CAPTCHA based on mouse, keyboard, and web behaviors, applied Hybrid biLSTM+Softmax to detect bots [42]. Both studies have achieved an accuracy of over 99%, proving that the analysis result of mouse behaviors can be an important reference for detecting bots. Acien et al. proposed to model the trajectories according to the Sigma-Lognormal model from the kinematic theory of rapid human movements and proved their BeCAPTCHA-Mouse can detect mouse trajectories generated by a GAN (Generative Adversarial Network) efficiently [1]. More importantly, mouse-based CAPTCHAs have been widely deployed as a part of commercial CAPTCHAs such as GeeTest[2]. However, existing methods can only distinguish bots from humans yet cannot detect CAPTCHA farm attacks.

It's important to highlight that some existing studies have attempted to detect CAPTCHA farms. Longe introduced Double CAPTCHA Challenge-Response Systems aimed at identifying man-in-the-middle attacks. However, the author incorrectly assumed that the solvers in CAPTCHA farms are bots, which is no longer the case today [28]. Additionally, Mohamed et al. developed a mechanism for detecting streaming-enabled game captcha farming by analyzing real-time game statistics [33]. Unfortunately, this mechanism has become ineffective due to changes in the verification protocol and the operation of CAPTCHA farms (see Sect. 3.2).

3 Methodology

In this section, we will first introduce the details of the proposed system, including a preprocessing procedure and the model consisting of a shared embedding network

[2] https://www.geetest.com/

and two classifiers. Then, we will present its application in CAPTCHA farm detection and user authentication.

3.1 Mouse movement embedding

3.1.1 Data preprocessing

Mouse trajectories are usually presented as a sequence of cursor coordinates and the corresponding timestamps $\{(x_1, y_1, t_1), (x_2, y_2, t_2), \ldots, (x_n, y_n, t_n)\}$. The primary challenge we encounter when constructing a reliable mouse movement embedding model is preprocessing mouse movement data gathered from various environments. Although solving CAPTCHAs can be seen as a controlled environment, user behaviors on websites are unpredictable and may include interruptions and accidental actions, reducing information density and introducing noise. There are two widely adopted preprocessing methods: one involves extracting a certain number of intentional mouse behaviors, such as clicks or movements, while the other entails segmenting the data using a fixed window. The former method is often combined with statistical analysis or feature engineering to process data collected in unguided environments due to the varying lengths and lack of continuity in mouse behaviors. Studies using this method typically require longer authentication times because the decision is not made until hundreds of mouse events are collected. On the other hand, the CAPTCHAs can typically be solved within 1 min [45], which may not meet the authentication time demands. The latter method, used in deep learning-based approaches, usually requires only a few seconds of mouse movement. However, it cannot be used for data collected in unguided environments since the mouse might be still for a long time, potentially disrupting the continuity of mouse movement. Thus, we need a more robust preprocessing procedure to preserve information in mouse trajectories collected in guided and unguided environments.

To tackle this problem, we propose to perform segmentation, trimming, and concatenation on the moving state of the mouse. As shown in Fig. 1, we initially segment mouse movements to ensure the continuity of the mouse movements (blue blocks), then discard meaningless records and concatenate the segments to increase the information density. We cut the data where there is a change in the key's state (pressed or released), or when the time gap between two successive timestamps exceeds 0.3 s. We consider segments with fewer than 5 data points, as well as those where the cursor movement in the x or y direction is under 5% of the screen (white blocks), to contain minimal mouse movement information and, therefore, discard them. We concatenate neighbor segments to provide samples with sufficient information for embedding and classification while maintaining continuity. Samples are generated using a sliding window with a specific maximum number of data points. All segments fully encompassed by the window are concatenated to create a sample. The stride of the window is set to one segment. To avoid creating discontinuous time series by directly concatenating (x, y) coordinate series, we opt to calculate the distance (dx and dy) and the speed of the mouse $\left(\frac{dx}{dt} \text{ and } \frac{dy}{dt}\right)$ between two data points along two directions. These features are chosen to describe the mouse movement in mouse dynamics and have been widely used in many existing studies [6, 10, 18]. Note that we rejected some alternative features, such as those related to scrolling or dragging, as the corresponding wheeling/dragging event might not be captured in real-world scenarios. We have set the sample length limit to 256. Therefore, the input for our model consists of two matrices with a shape of (256, 4).

3.1.2 Proposed model

Most existing studies train multiple classifiers, with each one performing a one-vs-rest binary classification. However, this

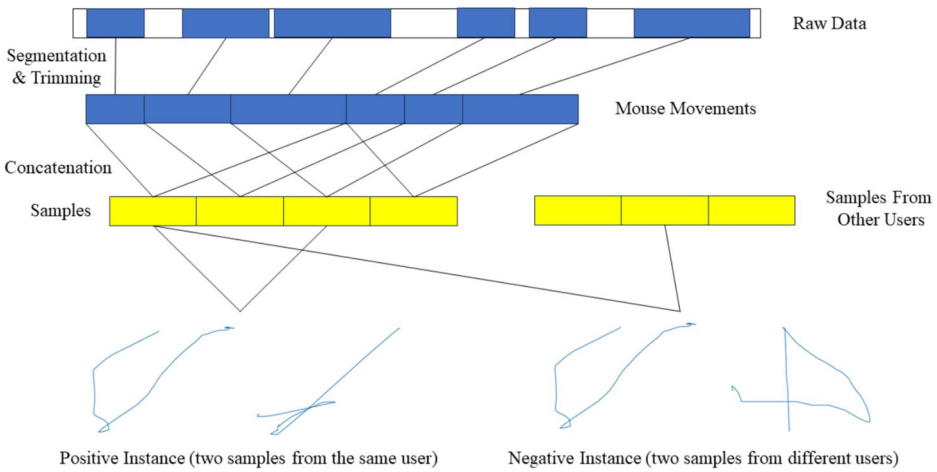


Fig. 1 The preprocessing procedure involves cutting meaningful mouse movements from the raw data and then combining them to create samples, which are essentially long enough time series that can be used as inputs. Two samples together form instances. Our model assesses the similarity between two samples within the instances and predicts whether they belong to the same user

approach results in heavy training and storage burdens as the number of users increases. Shen et al. trained a legitimate-vs-illegitimate classifier, disregarding the difference between legitimate users [37]. Chong et al. applied a 2D CNN with N outputs [10]. However, this "All-in-one" classifier relies on the personal information stored in the network. Both studies attempted to use only one classifier but failed to maintain performance comparable to other studies. We believe this is due to their lack of emphasis on the differences between input data, as individual characteristics significantly influence the trained classifiers.

The shared goal of authentication and CAPTCHA farm detection is to determine the similarity between the users. Widely deployed biometrics such as fingerprints and faces have similar structures. When a user claims to be someone, their input data is converted into a feature vector and then compared with the corresponding feature vectors stored in the database. The authentication system calculates the similarity and then either accepts or declines the user's claim. This framework allows us to avoid training individual classifiers for each user. As shown in Fig. 2, we input two samples at a time, which are then transformed into feature vectors using embedding networks with shared layers. These feature vectors are concatenated and passed to a classifier for similarity prediction. The embedding network consists of two parallel networks: a three-layer 1D Convolutional Neural Network (1D CNN) and a two-layer Long short-term memory (LSTM) network. It's worth noting that 2D CNN has been applied in numerous studies for image feature extraction. FaceNet, a famous model for face recognition, used 2D CNN for face embedding [36]. When working with images, it's common to use convolutions in both dimensions, but for time series, convolutions along the time axis are more appropriate. In our approach, we utilize three layers of 1D CNN with batch normalization and add a global average pooling layer at the end to decrease the dimensions. For sequence data, RNNs are effective due to their hidden states that function as "memory" to retain information about the processed part of the sequence. The LSTM, a type of RNN, is especially good at preserving long-term memory. In our case, we employ two layers of LSTM to capture information from longer mouse trajectories. Finally, the classifier is a three-layer feedforward neural network (FNN) with dropouts to prevent overfitting. The network is trained with an Adam optimizer at a learning rate of 0.00001 using binary cross-entropy loss. The model is trained for 200 epochs for both authentication and CAPTCHA farm detection.

We utilized a combination of traditional machine learning and deep learning since some studies that employed manual feature engineering had shown promising results. We extracted identical features from both inputs and obtained the difference between the two feature vectors. This difference was then passed to a Random Forest classifier. To perform feature engineering, we followed the approach outlined by Antal and Egyed-Zsigmond [4]. The final prediction was calculated by averaging the two similarities using decision-level fusion.

3.2 CAPTCHA farm detection

Today, most websites use CAPTCHAs from third-party providers, as the ongoing arms race between CAPTCHA providers and solvers has made it challenging to develop CAPTCHAs independently (Fig. 3a). Before implementing third-party CAPTCHAs, a website must obtain a public site key and a private secret key from the provider. Most websites will place the CAPTCHA and public site key in an iframe on the client side. As the first line of defense, users are typically required to click a checkbox to request and load the CAPTCHA content. The public site key is used to invoke the CAPTCHA. Once the CAPTCHA is solved, the client sends the collected data to the provider. If the provider

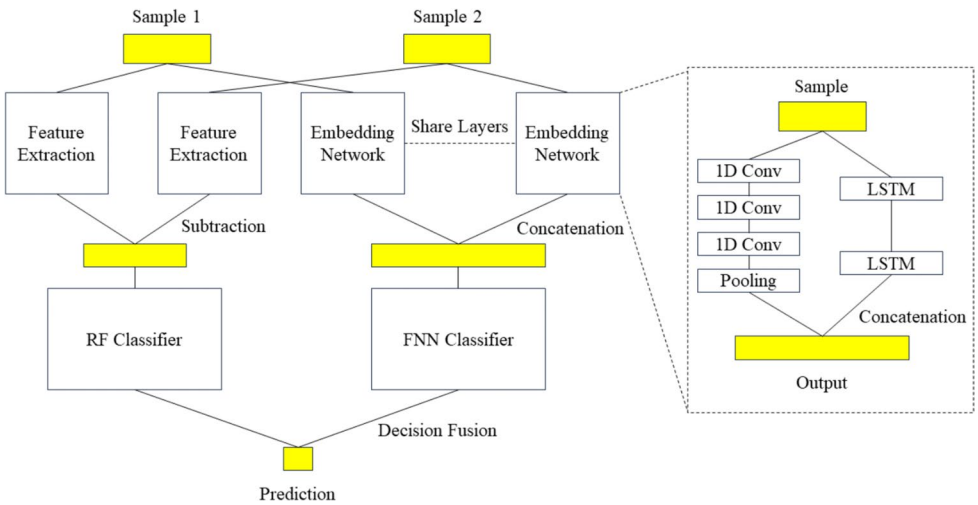
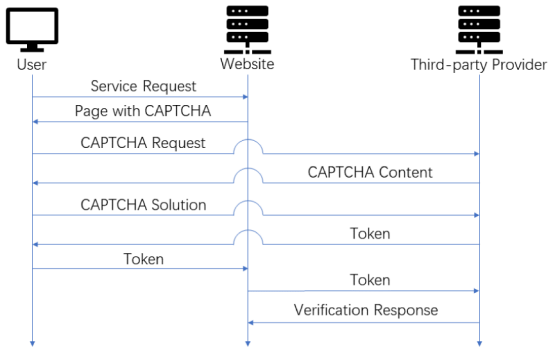
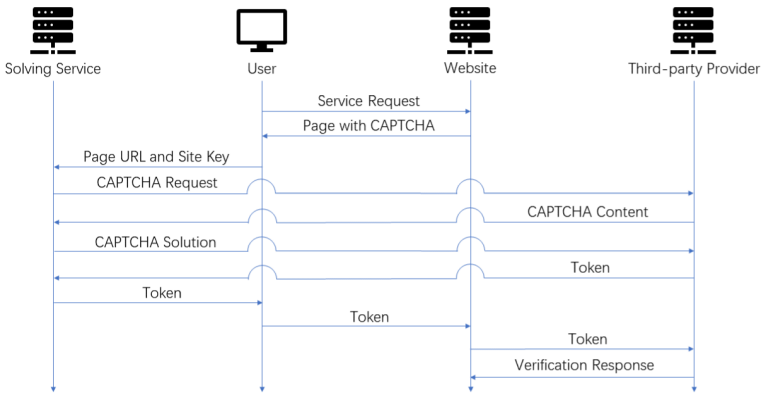


Fig. 2 Model Framework. We input two samples at a time, transform them into feature vectors using shared layers, and then feed them into an FNN classifier. Handcrafted features are also extracted from both inputs, and the difference between the two feature vectors is passed to a Random Forest (RF) classifier. The final prediction is calculated by averaging the two similarities using decision-level fusion

Fig. 3 The mechanism of third-party CAPCHAs and CAPTCHA farm attack



(a) The mechanism of third-party CAPCHAs. The page and the CAPTCHA are rendered asynchronously. A response token will be sent to the client if the CAPTCHA provider believes the client is a human, which is then sent to the website for verification.



(b) How attackers can breach CAPTCHAs using CAPTCHA farms. The attacker extracts information from the page that the solving services require to load the CAPTCHA. Then, the solving services will pretend to be the user and obtain the response token. The attackers can use these tokens to pass the websites' verification.

determines that the client is a human, a response token is generated and sent back to the client. The client then sends the response token to the server. The server can verify the response by checking with the CAPTCHA provider using the received response and the private secret key.

This verification procedure, however, has been breached by CAPTCHA-solving services, also known as CAPTCHA farms. These services are utilized by programs that mimic human behavior in order to bypass CAPTCHAs. They are often used for activities such as booking tickets for scalpers or creating fake social media accounts. Attackers use these services to reduce their costs; it is easier and cheaper for them to employ individuals in low-income areas to solve CAPTCHAs manually rather than doing it themselves or training new AIs. These attackers pretend to be regular users and misuse tokens to bypass the CAPTCHA. In Fig. 3b , it is illustrated that attackers who carry out harmful activities on websites do not directly engage with CAPTCHA providers. Instead, they relay the necessary information to CAPTCHA farms, where workers solve the CAPTCHA and send the response tokens they receive from the CAPTCHA provider back to the attackers. Upon receiving the response token, the user only needs to complete the remaining communication process after successfully passing the challenge, which typically involves entering the token in an element on the page and submitting the form or dispatching an event with it.

Threat model A website offers valuable content that is protected by a CAPTCHA, which is managed by a third-party provider. An attacker aims to access this content without manually solving the CAPTCHA, so they purchase a CAPTCHA-solving service from a CAPTCHA farm. The service is provided with the page URL and site key, allowing the assigned worker to solve the CAPTCHA and generate a token. The attacker will then use this token to bypass the CAPTCHA and perform further actions independently.

We've discovered that these attacks work well because the CAPTCHA provider and the website are unable to confirm the

consistency of the user's identity. The lack of identity verification means that anyone can solve the CAPTCHA. However, creating a new protocol to address this issue will be difficult mainly because of the collaboration between attackers and CAPTCHA farms. These parties share all the messages they send or receive from the CAPTCHA provider and websites. CAPTCHA solvers can use virtual machines, proxies, or remote controls to falsify hardware and software information. Therefore, a verification system that can ensure that the CAPTCHA solver and the user are the same person is necessary. By ensuring consistent identity verification, we can compel attackers to either hire solvers for further actions or expend more effort in mimicking them, thereby increasing the cost.

Problem definition Given two mouse trajectories from unseen users, CAPTCHA farm detection aims to predict whether they come from the same user.

Since CAPTCHA solvers can falsify hardware and software information, we suggest using mouse-based biometrics to create a system that can detect identity inconsistencies. This system measures the similarity between two mouse movements from unseen users using the framework mentioned earlier. This will help us develop a mechanism to detect CAPTCHA farms, as shown in Fig. 4. During each session, the mouse movement data used to solve the CAPTCHA will be stored until the session ends. The website can collect the user's mouse movements when the user takes subsequent actions, such as booking a ticket or voting. These records will then be preprocessed and fed into the model to measure their identity similarity. This will allow the website to determine whether the records are from different users. Since the subsequent actions are not performed by the CAPTCHA solvers, attackers using CAPTCHA farms can be detected.

When creating the training dataset, we preprocess all mouse trajectories in the given dataset and combine the resulting samples to generate positive and negative instances. Let's say that for a user with ID i, n_s^i samples are created. When generating positive instances, if we include all possible combinations, the number of positive instances for each user will be $\frac{n_s^i * (n_s^i - 1)}{2}$. However, this can become prohibitively large for users with a lot of data. Similarly, the number of negative instances increases significantly when scaling up. We have also observed that neighboring sample pairs exhibit substantial overlaps due to the sliding window. As a result, these sample pairs can be easily identified as positive, which is not practical in real-world scenarios. Thus, for a sample set $\left\{S_0^i, S_1^i, S_2^i, \ldots, S_{n_s^i-1}^i\right\}$, we generate positive instance set

$$\left\{\left(S_0^i, S_{\frac{n_s^i}{2}}^i\right), \left(S_1^i, S_{1+\frac{n_s^i}{2}}^i\right), \left(S_2^i, S_{2+\frac{n_s^i}{2}}^i\right), \ldots, \left(S_{\frac{n_s^i}{2}-1}^i, S_{n_s^i-1}^i\right)\right\}$$

to take full advantage of the data while generating a proper number of practical positive instances for training. This also helps generate sufficient and varied training and testing samples, thus improving generalization and easing overfitting. To create a balanced dataset, we combine each positive instance with one sample from it and another sample chosen randomly from other users to create a negative instance. The model is trained using an Adam optimizer with a learning rate of 0.00001 and binary cross-entropy loss. After training the model with the generated dataset, it can be used for identity consistency verification.

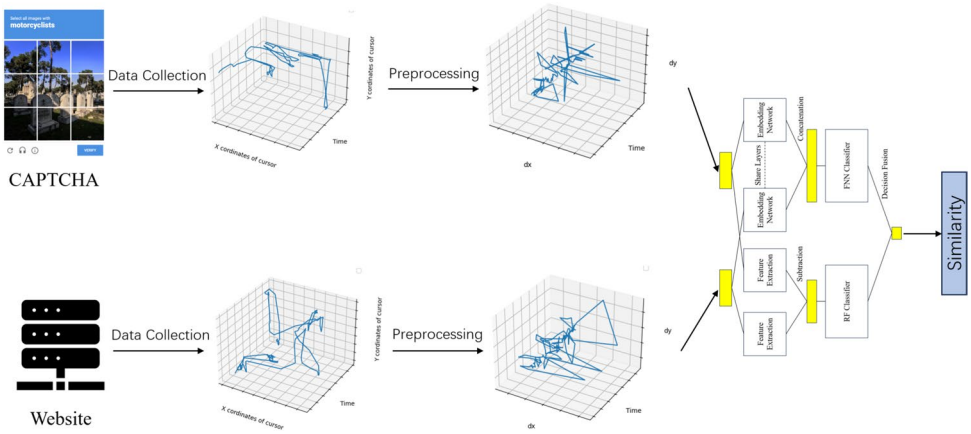


Fig. 4 CAPTCHA Farms Detection. During each session, the mouse movement data used to solve the CAPTCHA will be stored until the session ends. The website can collect the user's mouse movements when the user takes subsequent actions, such as booking a ticket or voting. These records will then be preprocessed and fed into the model to measure their identity similarity. Since the subsequent actions are not performed by the CAPTCHA solvers, attackers using CAPTCHA farms can be detected. The image of CAPTCHA comes from Google reCAPTCHA(https://developers.google.com/recaptcha)

3.3 Mouse-based authentication

The proposed system can also be used for user authentication. The main distinction between authentication and CAPTCHA farm detection is that, during the verification process of a claimed user's mouse trajectory, we can access the mouse trajectories recorded from this user during registration.

Problem definition Given mouse behavior data from registered users, a mouse trajectory, and a target user ID, mouse-based authentication is to predict whether the record comes from the claimed registered user.

Most existing mouse-based authentication methods train a classifier for each registered user using these records, leading to large training and storage costs. Due to the similarity between mouse-based authentication and identity inconsistency detection, the framework presented above can also be used for authentication and tackle this problem. The user's mouse trajectory will be collected during registration and stored just like the password in a password-based authentication system. After collecting data, the model can be trained and used as mentioned above. The authentication system can either use mouse trajectories as passwords, i.e., requiring the user to perform mouse movements before allowing the user to access, or it can be combined with other authentication systems to ensure continuous identity consistency throughout the session, also known as continuous authentication. It is worth mentioning that when new users register, unlike other methods that need to train a new classifier, the proposed framework does not necessarily require retraining since it can work on unseen users.

Enrollment sample selection When performing identity inconsistency detection, both samples of the input sample pair come from unseen users, whereas one sample comes from the training data for authentication. Given an input sample and a target user ID, we combine the sample with a sample from the target user and pass it to the model to determine whether the user is the target user. However, there may be multiple samples available from the training data, and the choice of which sample to use can impact the model's performance. To address this, we propose a mechanism for selecting the enrollment sample. Specifically speaking, after training the model, we assess the representativeness of the available samples using a validation set and select the most representative samples for authentication. We encountered a combinatorial explosion issue when evaluating the base samples from the training set. To address this, we randomly chose 20 samples from each user's validation set and 20 samples from other users' validation sets and used them to evaluate the samples from the training set. We used the binary cross entropy loss function to evaluate the samples' representativeness. The process of selecting enrollment samples can be represented as:

$$\min_j \frac{1}{40} * \sum_{n=0}^{20} -(log(f(S_j^i, S_n'^i)) + log(1 - f(S_j^i, S_n'^X)))$$

where f is the classifier, S'^i is the shuffled validation set of user i, and X has the discrete uniform distribution $f(x) = \frac{1}{N-1}$ for $x \in \{1, 2, \ldots, N\} \setminus \{i\}$. N represents the total number of users.

Dynamic authentication

Using mouse trajectories as biometrics offers the advantage of continuous data collection throughout a session. Unlike identity inconsistency detection time, which is constrained by CAPTCHA solving time, the authentication time can be more flexible due to the availability of a datastream of mouse trajectory. We leverage this difference to enhance the classifier's performance. Because of the diverse nature of mouse trajectories, conducting multiple classifications using different samples improves authentication accuracy. Specifically, by using a sliding window and utilizing new samples adjacent to the latest input, we can generate new predictions of identity similarity. As a result, dynamic authentication provides real-time confidence in the authentication outcome and identifies identity inconsistencies when the user changes. To ensure diversity, we use a half-overlapping sliding window. This approach can be repeated several times to optimize authentication accuracy at the expense of longer authentication time. Given sample S_j^i, let e_j^i be the number of segments included in sample S_j^i, the expanded sample set can be expressed as:

$$\{S_j^i, E(S_j^i), E^2(S_j^i), \ldots, E^{samp_n - 1}(S_j^i)\}$$

where $E(S_j^i) = S^i_{j+\frac{e_j^i}{2}}$ and $samp_n$ is an adjustable parameter representing the number of samples. The mean output of the classifications is calculated to provide a more reliable prediction.

4 Experiments

In this section, we introduce two datasets collected in guided and unguided environments and the evaluation metrics, which we used to test the proposed framework. Then, we present the experiments and results for CAPTCHA farm detection and user authentication.

Table 1 Datasets

Dataset	SapiMouse	Balabit
No. users	120	10
Environment	Guided	Unguided
Avg. time of mouse movements	156s	9984s
Avg. time of data	247s	63752s

4.1 Datasets

We utilized two public datasets for evaluation[3]. The SapiMouse dataset includes mouse trajectories from 120 users (92 male and 28 female) from the Sapientia University, aged between 18 and 53 years [6]. For each user, a 3-min and a 1-min session were recorded. This dataset was collected in a controlled environment where users played a short game and attempted to complete as many mouse operations as possible[4]. The participants were required to move the mouse to a randomly appearing icon and perform specific actions. The data was obtained using a JavaScript web application installed on the participants'own computers, which varies. The participants used different browsers and mouses with a huge variety of DPI (dot-per-inch) settings, in some cases they used touch-pad. This data collection process closely resembles CAPTCHA solving in terms of mouse movements. The Balabit Mouse Dynamics Challenge dataset consists of mouse trajectories from 10 users when they perform unspecified administrative tasks [47]. A network monitoring device is set between the client and the remote computer that inspects all traffic as described by the RDP protocol. 5–7 sessions were recorded for each user. The session length varies from 40 min to 7 h. Although some shorter sessions are available as test data, we did not utilize them due to lack of labeling and identity information. In our experiments, we combined the SapiMouse dataset with the Balabit Mouse Dynamics Challenge dataset to create a comprehensive dataset that incorporates data from both guided and unguided environments, which include varied behaviors. More details can be found in Table 1. The Balabit dataset provides significantly more data for each user.

We normalized all coordinates based on the monitor's resolution, which varies between users and can lead to an overly optimistic estimate of mouse movement. After this normalization, we calculated the time required for mouse movement for classification. We excluded data with less meaningful mouse movement. On average, it takes 18.5 s of effective mouse movement to generate a sample.

Both the Balabit dataset and the SapiMouse dataset contain data from multiple sessions. Unlike the sessions from the SapiMouse dataset, the sessions from the Balabit dataset vary in length. The split into training, validation, and test sets was done after we obtained the samples from different sessions and arranged them in order. We did not shuffle the data to ensure that the training, validation, and test sets were as independent as possible.

4.2 Evaluation metrics

False Accept Rate (FAR) and False Reject Rate (FRR) are two important metrics used to evaluate a biometric system. FAR measures the possibility of allowing an attacker to gain unauthorized access, while FRR measures the possibility of denying a legitimate user's access. Typically, the biometric system produces a probability rather than a definite result. The threshold for accepting or rejecting a match can be adjusted to make the system more secure (higher FRR and lower FAR) or to reduce inconvenience for legitimate users (higher FAR and lower FRR). To provide a more comprehensive view, the FRR-FAR curve is plotted. The Equal Error Rate (EER) is the point at which FAR and FRR are equal, and it is widely used as a metric to reflect the balance between FAR and FRR. Additionally, the area under the curve (AUC) of the receiver operating characteristic (ROC) curve is commonly used as a quantitative measurement of the overall performance of the classifier.

We also adopted two metrics concerning time, namely, the training time of the classifier and the authentication time. The training time indicates how quickly a classifier can be trained. The authentication time refers to the amount of mouse movement data needed to complete the authentication process, demonstrating the convenience of the biometric system.

4.3 CAPTCHA farm detection

In order to simulate the threat scenario of a CAPTCHA farm attack, we randomly divided the users in the dataset into two groups: training users and testing users. There was no overlap between the two groups to ensure that the testing users were unfamiliar to the classifier. The model was then trained using the training users and tested on the unseen testing users.

4.3.1 Comparison with time series classification models

Since very few existing studies have attempted to address this type of attack, we employed three state-of-the-art time

[3] It is tempting to test the trained model in a realistic environment. However, since the CAPTCHA farms only render the CAPTCHA, we are unable to collect mouse trajectories from the CAPTCHA farms as only the CAPTCHA providers can access their mouse trajectories.

[4] https://mousedynamicsdatalogger.netlify.app/

Table 2 Comparation of MultiROCKET, HIVE-COTEv2, InceptionTime, and our model on identity inconsistency detection

Method	AUC (%)	EER (%)	Training time
Hydra + MultiROCKET [16]	87.7	20.0	798.7 min
HIVE-COTEv2 [32]	85.4	20.4	505.2 min
InceptionTime [23]	93.6	13.2	527.9 min
Our work	96.9	8.9	72.0 min

series classification models as baselines: Hydra + MultiROCKET, HIVE-COTEv2, and InceptionTime.

- Random Convolutional Kernel Transform (ROCKET) uses a large number of randomly parameterized convolutional kernels to transform the data into features [14]. MiniROCKET, a variation of ROCKET, accelerates ROCKET by fixing many random components of it [15]. MultiROCKET further improves MiniROCKET by considering the first-order differences of time series and adding multiple pooling operators and transformations [43]. Hybrid Dictionary–Rocket Architecture (Hydra) combined dictionary method for time series classification and ROCKET. Among tested variations, the combination of Hydra and MultiROCKET, i.e., Hydra + MultiROCKET yields the best results [16].
- The Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE) is a heterogeneous meta ensemble for time series classification [27]. HIVE-COTE 2.0 [32], as its upgraded variation, is an ensemble of four representative TSC models that use different approaches: Temporal Dictionary Ensemble (TDE) for dictionary-based approaches [31], Diverse Representation Canonical Interval Forest (DrCIF) [30] for interval based approaches, an ensemble of ROCKET called Arsenal for convolutional based approaches, and Shapelet Transform Classifier (STC) for shapelet approaches [8].
- InceptionTime, a representative deep learning approach, is an ensemble of five deep learning networks based on Inception networks that are initialized randomly [23]. Matthew et al. evaluated the state-of-the-art TSC algorithms with 112 datasets from the University of California, Riverside (UCR) archive [9, 13] and 30 new datasets. The averaged ranked performance of HIVE-COTEv2, Hydra + MultiROCKET, and InceptionTime are 2.75, 3.2545, and 4.3661, ranked in the top three, respectively.

We implemented MultiROCKET, HIVE-COTEv2, and InceptionTime using `aeon`, a Python toolkit for learning from time series[5]. For Hydra, we are using the multivariate version[6]. Given their relatively long training times, we adjusted their training parameters and datasets to ensure that their training time is within an order of magnitude longer than our model's (Table 2).

- The number of training epochs of InceptionTime is set to 200 to align with our model.
- The scale of the training dataset for Hydra + MultiROCKET is limited to 20,000 instances randomly selected from the original training dataset.
- Similar to Hydra + MultiROCKET, the training dataset for HIVE-COTEv2 is limited to 10,000 instances. The training time limit was set to 720 min.

Although our limitation for Hydra + MultiROCKET and HIVE-COTEv2 reduced the scale of their training dataset, their training time was 7 to 10 times ours even with the limitations, and removing the limitations would lead to unbearable training time. These models take one multivariate time series as input. To meet their requirements, we concatenate the two samples in each instance along the variates direction, leading to an input including dx, dy, $\frac{dx}{dt}$, and $\frac{dy}{dt}$ for both samples. We used 5-fold cross-validation to ensure every user was considered during model performance evaluations. 25% of the training users were reserved as the validation set in each fold. The experiments were conducted on a machine with an Intel Xeon Gold 5220 CPU at 2.20 GHz and a NVIDIA Tesla V100 GPU. Fig. 5a shows the mean FRR-FAR curves. Our model achieved the lowest FRR and FAR at 8.9%. In contrast to the results obtained using the UCR archive, among the state-of-the-art time series classification methods, InceptionTime demonstrated significantly better performance than Hydra + MultiROCKET and HIVE-COTEv2, which obtained less promising results. This is likely because Hydra + MultiROCKET and HIVE-COTEv2 focused more on the frequency domain of the time series, which is not as relevant for analyzing mouse trajectories. This suggests the clear advantage of deep learning in processing mouse trajectory data. The averaged AUC and EER are presented in Table 1. The fact that our approach achieved the highest AUC and the lowest EER with a significantly shorter training time proves its effectiveness and simplicity.

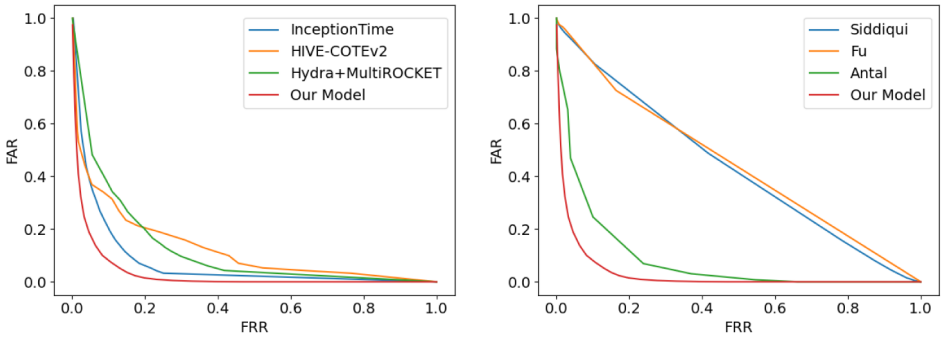
4.3.2 Comparison with mouse-based authentication systems

Since other mouse-based authentication systems can theoretically be used to detect identity inconsistencies, we have implemented three different state-of-the-art mouse-based

[5] https://www.aeon-toolkit.org/

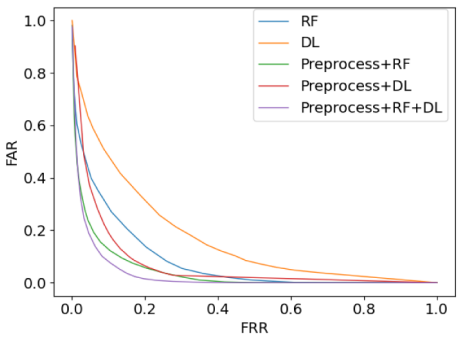
[6] https://github.com/angus924/hydra

Fig. 5 The FRR-FAR curves in identity inconsistency detection experiments



(a) Mean FRR-FAR curves of Hydra+MultiROCKET, HIVE-COTEv2, InceptionTime, and our model.

(b) Mean FRR-FAR curves when using other mouse-based authentication systems and our model.



(c) Mean FRR-FAR curves when using different ablation settings.

Table 3 Comparation of three other mouse-based authentication systems and our model on identity inconsistency detection

Method	AUC (%)	EER (%)	Verification time
Siddiqui et al. [41]	56.8	45.4	4.845 s
Fu et al. [18]	58.2	44.3	191.073 s
Antal et al. [4]	92.8	13.8	0.141 s
Our work	96.9	8.9	0.005 s

Table 4 The mean AUC and EER when using different ablation settings

Method	AUC (%)	EER (%)	Verification time
RF	92.0	15.2	<0.001 s
DL	83.0	25.1	0.005 s
Preprocess + RF	96.0	10.2	<0.001 s
Preprocess + DL	93.3	13.2	0.005 s
Preprocess + RF+DL	96.9	8.9	0.005 s

authentication systems. This provides a more comprehensive comparison, including two deep learning-based approaches and one traditional machine learning-based approach. In order to detect inconsistencies in identity, we use one of the two provided mouse trajectories to register and train a new classifier to determine whether the other mouse trajectory belongs to the same user. As training new classifiers for thousands of instances can be time-consuming, especially for deep learning-based approaches, we have sampled two 30-s mouse trajectories from each user for evaluation purposes.

As presented in Fig. 5b and Table 3, the first two methods based on 1D CNN and 1D CNN + BiLSTM architectures were ineffective at identifying identity inconsistencies. This was primarily due to the lack of training data. It is also worth noting that their verification times are longer, especially Fu et al.'s approach, which takes 191.073 s to train a model and perform verification. However, the third approach, which involved feature engineering and a Random Forest classifier, was less affected by this issue. In fact, it outperformed the first two approaches and achieved an AUC of 92.8% and an EER of 13.2%. Furthermore, it only takes 0.141 s to train a classifier and produce the verification result. Our approach achieved even better results, with a 3.9% higher AUC, a 4.3% lower

EER, and a verification time that was 28 times faster since we avoided training a new classifier.

4.3.3 Ablation

We divided the proposed model into three components and conducted an ablation experiment to assess its effectiveness: the preprocessing procedure, the RF-based classifier, and the DL-based classifier. In the preprocessing step, we omitted the segment filter and sliding window sampling. Neighboring mouse events that have a time gap of over 10 s are treated as separate samples, without concatenation. While the structures for feature extraction, the embedding network, and the classifiers remain unchanged, the RF-based and DL-based classifiers are trained and evaluated independently.

The results are presented in Fig. 5c and Table 4. The RF-based classifier outperformed the DL-based classifier, particularly when the latter was trained without our preprocessing procedure. One possible explanation for this is that the lack of sliding window sampling resulted in a smaller training set, which significantly lowered the performance of the DL-based classifier. Additionally, the RF-based classifier demonstrated a clear advantage in verification time. When utilizing the RF-based classifier, the preprocessing procedure led to an increase of 4.0% in the AUC and a decrease of 5.0% in the EER. The improvements were even more pronounced for the DL-based classifier, with the AUC increasing by 10.3% and the EER decreasing by 11.9%. It can also be observed that all three components contributed to obtaining the highest AUC and lowest EER in this ablation experiment.

In summary, the two experiments presented above have shown the advantages of the proposed system for detecting identity inconsistency. Our approach achieved better performance in a shorter time for both training and verification.

4.4 Authentication

Unlike the former experiment, the training user set and the testing user set are intersected in this scenario. To test the performance when using different user sets as the registered users, we evaluate our model using the following settings:

Setting I The users from both datasets are mixed and shuffled. 80% of the users are considered registered users, while the remaining 20% are imposters. 80% of the samples from the registered users are used for training, and the remaining 20% of the samples, along with all samples from the imposters, are used for testing.

Setting II The same training/test split method is used for the SapiMouse dataset. Users from the Balabit dataset are considered additional imposters to expand the test set.

Setting III The users in both datasets are mixed and shuffled, and we split them into training and test sets using the same method. However, the users from the SapiMouse dataset that are used for training will not be treated as registered users. This is to prevent overfitting that can occur when training with only the Balabit dataset, which is a relatively small training set.

We conducted two experiments to verify the effectiveness of the enrollment sample selection and dynamic authentication and evaluate the performance of our model.

Effect of enrollment sample selection and dynamic authentication In our first experiment, we demonstrate how selecting enrollment samples and using dynamic authentication can enhance the performance of our model. This experiment was carried out under Setting I. For the baseline, we replaced the selected enrollment samples with a randomly chosen sample from the user's training set. As illustrated in Fig. 6, enrollment sample selection led to higher AUC, lower FAR, and lower FRR. The improvement is also evident with an increase in the number of samples, with or without enrolling sample selection. We stopped at 7 samples because beyond this point, authentication time increases significantly, taking more than a minute on average. Compared to the worst result obtained without enrollment sample selection and dynamic authentication, the best result shows an increase of 0.8% in AUC, a decrease of 4.0% in FAR, and a decrease of 1.2% in FRR. This demonstrates that enrollment sample selection and dynamic authentication can greatly improve our model's performance. For all subsequent experiments, we used enrollment sample selection and 7 samples.

Authentication performance Next, we evaluate the performance of our model compared to other studies under 3 different settings. We will now assess our model's performance in comparison to other studies using three different settings. This is necessary because while mixing the dataset can demonstrate how well our model generalizes, the varying data volumes between the datasets significantly impact performance, leading to unavoidable bias. As seen in Table 5, the AUC score for Setting I is 98.5%, which is 1.5% higher than when using Setting II and 0.6% lower than when using Setting III. In Fig. 7, the differences in FAR and FRR show a similar trend, indicating that the model performs better on the Balabit dataset. Furthermore, Fig. 8 illustrates the distribution of the AUC for each user. It's evident that the AUC of users in the SapiMouse dataset is more widely spread. This could be due to the fact that more training data can be obtained from the Balabit dataset for each user than from the SapiMouse dataset.

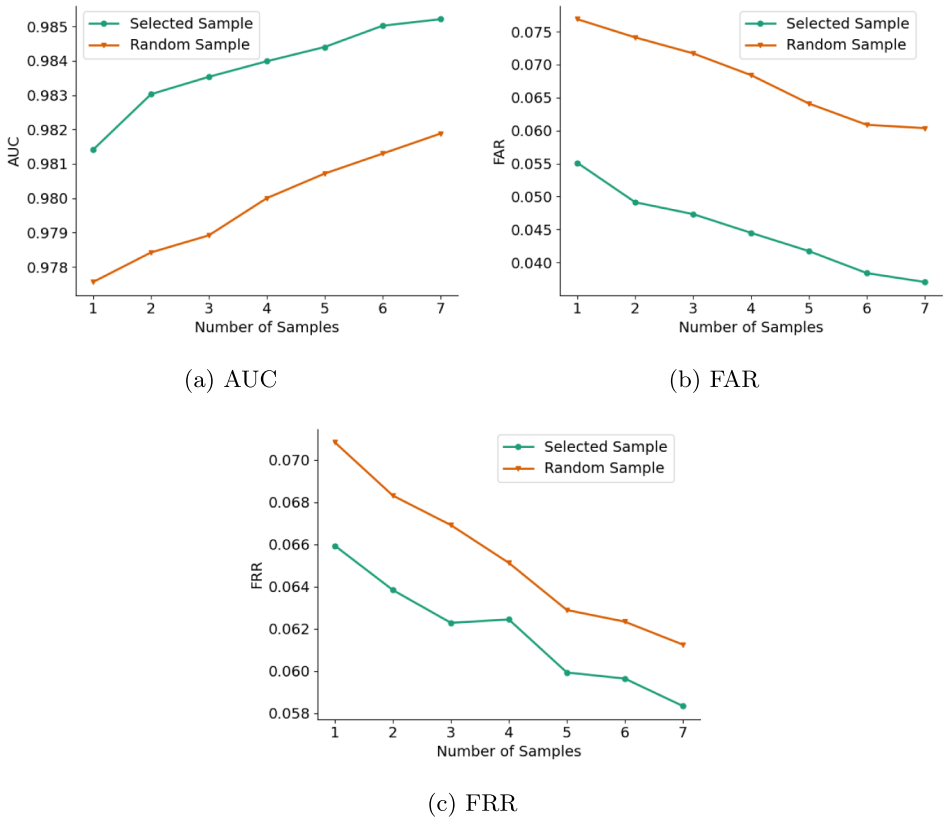
In Table 5, other mouse-based authentication approaches are also detailed. Some studies have achieved more accurate authentication systems [2, 20, 38, 46]. However, they require

Table 5 Comparation of the result of our authentication experiment and other representative studies

Study	No. subjects	FAR (%)	FRR (%)	AUC (%)	No. classifiers	Authentication time
Gamboa and Fred [20][a]	50	0.2	0.2		50	10–15 min
Ahmed and Traore [2][a]	22	2.5	2.5		22	13.55 min
Zheng et al. [46][a]	30	1.3	1.3		30	37.73 min
Shen et al. [38][a]	159	0.1	1.0		159	5 min
Fu et al.[18]	15	3.2	3.2	99.4	15	6.11 sec
Antal et al.[5][b]	10 (Balabit)			98	10	
Antal et al.[6]	120 (SapiMouse)	17	17	88	120	15 s
Shen et al.[37][a]	37	8.7	7.7		1	11.8 s
Chong et al.[10]	10 (Balabit)	10	10	96	1	10 s
Our Work (Setting I)	130	5.1	5.1	98.5	1	74 s
Our Work (Setting II)	130	9.1	9.1	97.0	1	74 s
Our Work (Setting III)	130	3.2	3.2	99.1	1	74 s

[a]This study did not provide the AUC score

[b]This study did not provide the FAR, FRR and authentication time



(a) AUC (b) FAR

(c) FRR

Fig. 6 The effect of enrollment sample selection and dynamic authentication. Enrollment sample selection and longer input led to higher AUC, lower FAR, and lower FRR. Compared to the worst result obtained without enrollment sample selection and dynamic authentication, the best result shows an increase of 0.8% in AUC, a decrease of 4.0% in FAR, and a decrease of 1.2% in FRR

much longer authentication times and training a classifier for each user. Fu et al.'s approach performs very similarly to ours, with a short authentication time [18]. However, we have implemented this approach in the experiment presented in the previous section and demonstrated its extended training time for each user. To our knowledge, only two existing studies [10, 37] have managed to achieve mouse-based authentication using a single classifier, and our model outperforms both of them. Therefore, our model is competitive with other state-of-the-art mouse-based authentication approaches. It is also worth noting that through the use of dynamic authentication, our model offers greater flexibility in terms of authentication timing and can be further adjusted to meet specific requirements.

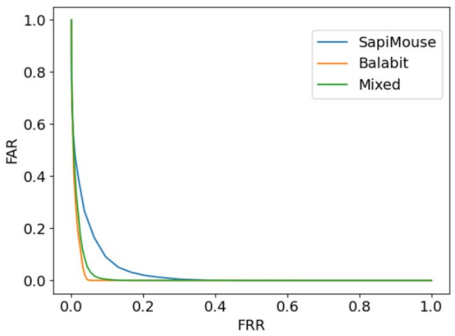


Fig. 7 Authentication FRR-FAR curves using different registered user sets. The model performs better on the Balabit dataset

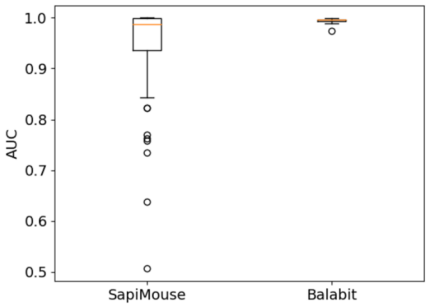


Fig. 8 Distribution of AUC when viewing users in one dataset as registered users. The AUC of users in the SapiMouse dataset is more widely spread

5 Conclusion and discussion

We recognized a lack of research on identifying CAPTCHA farms and the need to simplify mouse-based authentication systems. To address this, we created a new system that combines CAPTCHA farm detection and identity inconsistency detection. Taking inspiration from fingerprints and facial recognition, we developed a single model to assess the similarity between two mouse movements for authentication and identity verification. Furthermore, we suggest using diverse mouse movement data to implement enrollment sample selection and dynamic authentication, enhancing both security and flexibility during authentication. Our model was tested on a dataset with 130 users, achieving a 96.9% AUC score for identity inconsistency detection and 99.1% for authentication. These results indicate that our model, powered by a single classifier, performs exceptionally well and can detect inconsistencies in the identity of new users, making it a promising approach for detecting CAPTCHA farm attacks and authentication.

The potential impact of AI advancements The latest advancements in AI, particularly Multimodal Large Language Models (MLLMs), have raised new concerns about CAPTCHA security. Despite their impressive understanding capabilities, MLLMs do not have any advantages over human workers in CAPTCHA farms when it comes to solving CAPTCHAs, apart from the cost. It's also important to note that successfully passing CAPTCHAs involves not only finding the correct solution but also entering it without being identified as a bot. Given the advancements in mouse-based CAPTCHAs (as discussed in Sect. 2), it is questionable whether replacing human workers with AIs is feasible and cost-effective. Furthermore, even if the arms race between CAPTCHA providers and CAPTCHA farms changes the landscape of attack techniques, the proposed system will remain effective as long as solving CAPTCHAs requires the use of a mouse.

Corner cases: legitimate helpers With the information that the website can retrieve while solving CAPTCHA, CAPTCHA farms and helpers legitimated by the user cannot be distinguished, since the CAPTCHA solvers can use virtual machines and proxies or remote control to perfectly fake that they are legitimate helpers. Furthermore, it is difficult to draw a line between illegitimate helpers and legitimate helpers in some situations. For example, except for the APIs for programmers, some CAPTCHA farms also offer browser plugins for the convenience of normal users, which indicates that some unmalicious users are also using their services. Thus, to address this problem, additional information and measures are required. Still, if trading user convenience for the defense against CAPTCHA farms is unacceptable, the proposed classifier can be used as an alarm to indicate whether further verification is needed.

Aknowledgement This work is supported by the Provincial Key Research and Development Program of Anhui (202423l10050033) and the Ministry of Public Security Science and Technology Plan (Project Number 2023JSZ01).

Author Contributions Rui Jin designed and implemented the research under the guidance of Yong Liao.

Data Availability Data sharing is not applicable to this article as no new data were created or analyzed in this study.

Declarations

Conflict of interest All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

References

1. Acien, A., Morales, A., Fierrez, J., et al.: Becaptcha-mouse: synthetic mouse trajectories and improved bot detection. Pattern Recognit. **127**, 108643 (2022)

2. Ahmed, A.A.E., Traore, I.: A new biometric technology based on mouse dynamics. IEEE Trans. Depend. Secure Comput. **4**(3), 165–179 (2007)
3. Antal, M., Denes-Fazakas L.: User verification based on mouse dynamics: a comparison of public data sets. In: 2019 IEEE 13th International Symposium on Applied Computational Intelligence and Informatics (SACI), IEEE, pp. 143–148 (2019)
4. Antal, M., Egyed-Zsigmond, E.: Intrusion detection using mouse dynamics. IET Biom. **8**(5), 285–294 (2019)
5. Antal, M., Fejér, N.: Mouse dynamics based user recognition using deep learning. Acta Universitatis Sapientiae Informatica **12**(1), 39–50 (2020)
6. Antal, M., Fejér, N., Buza, K.: Sapimouse: mouse dynamics-based user authentication using deep feature learning. In: 2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI), IEEE, pp. 61–66 (2021)
7. Bera, A., Bhattacharjee, D., Shum, H.P.: Two-stage human verification using handcaptcha and anti-spoofed finger biometrics with feature selection. Expert Syst. Appl. **171**, 114583 (2021)
8. Bostrom, A., Bagnall, A.: Binary shapelet transform for multiclass time series classification. In: Special Issue on Big Data Analytics and Knowledge Discovery, Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXII, pp. 24–46 (2017)
9. Chen, Y., Keogh, E., Hu, B., et al.: The ucr time series classification archive. www.cs.ucr.edu/~eamonn/time_series_data/ (2015)
10. Chong, P., Elovici, Y., Binder, A.: User authentication based on mouse dynamics using deep neural networks: a comprehensive study. IEEE Trans. Inf. Forensics Secur. **15**, 1086–1101 (2019)
11. Chu, Z., Gianvecchio, S., Koehl, A., et al.: Blog or block: detecting blog bots through behavioral biometrics. Comput. Netw. **57**(3), 634–646 (2013)
12. Chu, Z., Gianvecchio, S., Wang, H.: Bot or human? A behavior-based online bot detection system. In: From Database to Cyber Security: Essays Dedicated to Sushil Jajodia on the Occasion of His 70th Birthday pp. 432–449 (2018)
13. Dau, H.A., Keogh, E., Kamgar, K., et al.: The ucr time series classification archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/ (2018)
14. Dempster, A., Petitjean, F., Webb, G.I.: Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. Data Min. Knowl. Discov. **34**(5), 1454–1495 (2020)
15. Dempster, A., Schmidt, DF., Webb, G.I.: Minirocket: a very fast (almost) deterministic transform for time series classification. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 248–257 (2021)
16. Dempster, A., Schmidt, D.F., Webb, G.I.: Hydra: competing convolutional kernels for fast and accurate time series classification. Data Min. Knowl. Discov. 1–27 (2023)
17. Everitt, R.A., McOwan, P.W.: Java-based internet biometric authentication system. IEEE Trans. Pattern Anal. Mach. Intell. **25**(9), 1166–1172 (2003)
18. Fu, S., Qin, D., Qiao, D., et al.: Rumba-mouse: rapid user mouse-behavior authentication using a cnn-rnn approach. In: 2020 IEEE Conference on Communications and Network Security (CNS), IEEE, pp. 1–9 (2020)
19. Fu, S., Qin, D., Amariucai, G., et al.: Artificial intelligence meets kinesthetic intelligence: mouse-based user authentication based on hybrid human-machine learning. In: Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security, pp. 1034–1048 (2022)
20. Gamboa, H., Fred, A.: A behavioral biometric system based on human-computer interaction. In: Biometric Technology for Human Identification, SPIE, pp. 381–392 (2004)
21. Gao, L., Lian, Y., Yang, H., et al.: Continuous authentication of mouse dynamics based on decision level fusion. In: 2020 International Wireless Communications and Mobile Computing (IWCMC), IEEE, pp. 210–214 (2020)
22. Hu, T., Niu, W., Zhang, X., et al.: An insider threat detection approach based on mouse dynamics and deep learning. Secur. Commun. Netw. **2019**, 3898951 (2019)
23. Ismail Fawaz, H., Lucas, B., Forestier, G., et al.: Inceptiontime: finding alexnet for time series classification. Data Min. Knowl. Discov. **34**(6), 1936–1962 (2020)
24. Kaixin, W., Hongri, L., Bailing, W., et al.: A user authentication and identification model based on mouse dynamics. In: Proceedings of the 6th International Conference on Information Engineering, pp. 1–6 (2017)
25. Kang, S.J., Kim, S.K.: User interface-based repeated sequence detection method for authentication. Intell. Autom. Soft Comput. **35**(3), 2573–2588 (2023)
26. Khan, S., Devlen, C., Manno, M., et al.: Mouse dynamics behavioral biometrics: a survey. ACM Comput. Surv. **56**(6), 1–33 (2024)
27. Lines, J., Taylor, S., Bagnall, A.: Time series classification with hive-cote: the hierarchical vote collective of transformation-based ensembles. ACM Trans. Knowl. Discov. Data (TKDD) **12**(5), 1–35 (2018)
28. Longe, O.B.: Mitigating captcha relay attacks using multiple challenge-response mechanism. Comput. Inf. Syst. **14**(3) (2010)
29. López, C., Solano, J., Rivera, E., et al.: Adversarial attacks against mouse-and keyboard-based biometric authentication: black-box versus domain-specific techniques. Int. J. Inf. Secur. **22**(6), 1665–1685 (2023)
30. Middlehurst, M., Large, J., Bagnall, A.: The canonical interval forest (cif) classifier for time series classification. In: 2020 IEEE International Conference on Big Data (Big Data), IEEE, pp. 188–195 (2020)
31. Middlehurst, M., Large, J., Cawley, G., et al.: The temporal dictionary ensemble (tde) classifier for time series classification. In: Part, I. (ed.) Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings. Springer, pp. 660–676 (2021)
32. Middlehurst, M., Large, J., Flynn, M., et al.: Hive-cote 2.0: a new meta ensemble for time series classification. Mach. Learn. **110**(11–12), 3211–3243 (2021)
33. Mohamed, M., Gao, S., Saxena, N., et al.: Dynamic cognitive game captcha usability and detection of streaming-based farming (2014)
34. Motoyama, M., Levchenko, K., Kanich, C., et al.: Re: captchas-understanding captcha-solving services in an economic context. In: USENIX Security Symposium, p. 3 (2010)
35. Salman, OA., Hameed, SM.: Using mouse dynamics for continuous user authentication. In: Proceedings of the Future Technologies Conference (FTC) 2018: Vol. 1. Springer, pp. 776–787 (2019)
36. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: a unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 815–823 (2015)
37. Shen, C., Cai, Z., Guan, X., et al.: User authentication through mouse dynamics. IEEE Trans. Inf. Forensics Secur. **8**(1), 16–30 (2012)
38. Shen, C., Chen, Y., Guan, X., et al.: Pattern-growth based mining mouse-interaction behavior for an active user authentication system. IEEE Trans. Depend. Secure Comput. **17**(2), 335–349 (2017)
39. Shi, Y., Wang, X., Zheng, K., et al.: User authentication method based on keystroke dynamics and mouse dynamics using hda. Multim. Syst. **29**(2), 653–668 (2023)

40. Siddiqui, N., Dave, R., Seliya, N.: Continuous user authentication using mouse dynamics, machine learning, and minecraft. In: 2021 International Conference on Electrical. IEEE, Computer and Energy Technologies (ICECET), pp. 1–6 (2021)
41. Siddiqui, N., Dave, R., Vanamala, M., et al.: Machine and deep learning applications to mouse dynamics for continuous user authentication. Mach. Learn. Knowl. Extract. **4**(2), 502–518 (2022)
42. Süzen, A.A.: Uni-captcha: a novel robust and dynamic user-non-interaction captcha model based on hybrid bilstm+ softmax. J. Inf. Secur. Appl. **63**, 103036 (2021)
43. Tan, C.W., Dempster, A., Bergmeir, C., et al.: Multirocket: multiple pooling operators and transformations for fast and effective time series classification. Data Min. Knowl. Discov. **36**(5), 1623–1646 (2022)
44. Von Ahn, L., Blum, M., Hopper, NJ., et al.: Captcha: using hard ai problems for security. In: Eurocrypt. Springer, pp. 294–311 (2003)
45. Weng, H., Zhao, B., Ji, S., et al.: Towards understanding the security of modern image captchas and underground captcha-solving services. Big Data Min. Anal. **2**(2), 118–144 (2019)
46. Zheng, N., Paloski, A., Wang, H.: An efficient user verification system using angle-based mouse movement biometrics. ACM Trans. Inf. Syst. Secur. (TISSEC) **18**(3), 1–27 (2016)
47. Fülöp Á., Kovács, L., Kurics, T., Windhager-Pokol, E.: Balabit mouse dynamics challenge data set. https://github.com/balabit/Mouse-Dynamics-Challenge (2016)