RESEARCH ARTICLE

RSP Science Hub

International Research Journal on Advanced Science Hub
2582-4376
www.rspsciencehub.com
Vol. 07, Issue 05 May

Check for updates

http://dx.doi.org/10.47392/IRJASH.2025.057

Adaptive Behavioral Authentication for Bot Detection Using ML

M Amshavalli[1], S Chandiran[2], R Dharshini[3], A M Edwin Arul Solomon[4]
[1]Assistant professor, Dept. of CSE, Erode Sengunthar Engineering College, Erode, Tamil Nadu, India.
[2,3,4]UG Scholar, Dept. of CSE, Erode Sengunthar Engineering College, Erode, Tamil Nadu, India.
Emails: *amshavalli1494@gmail.com[1], chandiranesec@gmail.com[2], dharshiniramasamy2004@gmail.com[3], edwinarulsolomon@gmail.com[4]*

Article history
Received: 22 March 2025
Accepted: 05 April 2025
Published: 26 May 2025

Keywords:
Bot detection; Feature engineering; Human-Bot interaction; Keystroke dynamics; Mouse movement patterns.

Abstract
Automatic bots increasingly compromise the security and reliability of online platforms, requiring effective detection mechanisms. This paper proposes a behavioral-based framework to distinguish human users from the bots through the analysis of keystroke dynamics and mouse movement patterns. A dedicated web-based interface was developed to capture real-time interaction data, including writing speeds and cursor trajectories. These features were collected in a structured dataset and used to train machine learning for accurate prediction. Experimental results indicate a high classification accuracy, which shows the framework's capability to identify non-human interactions. The integration of behavioral metrics with machine learning contributes to robust, adaptive bot detection, and enhancing the integrity and security of the digital systems.

1. Introduction

Increasing the increase of automated robots is a significant threat to the integrity and safety of the online system. Bots are used to spam, steal data, offend security measures and degrade the user experience. Traditional CAPTCHA (fully automated public touring tests to separate computers and humans) are no longer reliable, as robots can now solve even more complex challenges, while humans often draw drawers and infiltrate these systems. To provide a smart and more user-profit solution, the project presents a behavioral approval system that uses machine learning to distinguish between human users and robots. Instead of relying on the visual puzzle, our approach utilizes passive data collection methods, especially system behavior and environmental indicators such as mouse movement patterns and key pressure behavior. These properties create a unique behavioral profile for each user during the interaction. Using these parameters, we train a randomly classify to recognize the difference between bot and human behavior. The trained model then predicts real -time user types based on the interaction that is collected. This document presents a detailed interpretation of the function of the system, including data collection techniques, processing steps, model training, testing and evaluation processes. The main goal is to design an accurate and adaptive system that provides secure, fast and non-intrusive approval to protect websites from automatic content hazards. [1]

1.1. Methods of Data Collection

To enable effective bot detection, we focus on capturing distinct behavioral signals from users during interaction. These signals are collected passively in the background to ensure minimal user interference while maintaining accuracy in classification [2]

1.2. Mouse Movement Patterns

Human users tend to display natural, erratic mouse movements influenced by coordination. These include subtle hesitations, curvature in paths, and variable speeds when navigating or clicking. In contrast, bots often exhibit straight-line trajectories or constant velocity motion. By logging coordinates, speed changes, acceleration, and path curvature, we extract features that help in distinguishing automated input from organic human behavior. [Figure 1]

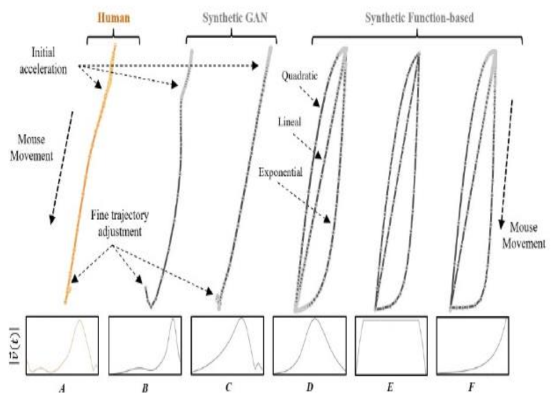
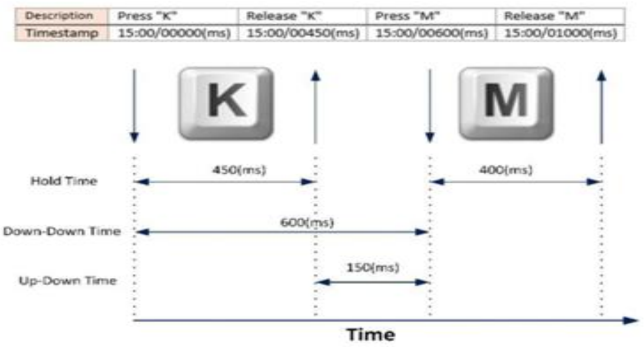


Figure 1 Mouse Movement Visualization

Figure 1 imagines the movement of a mouse on the screen during the time period. The image depicts the path and the pattern of the cursor, which can be used to analyze user behavior, interaction and work performance in studies related to interactions between people and computers. The dynamic nature of mouse movements provides insight into the user's decision-making process, work efficiency and general user experience.

1.3. Keystroke Behavior

Keystroke behavior refers to a specific pattern of how an individual type, which contains time-like factors between keystrokes (intermediate delay) and each key is pressed (dwell time). It varies considerably among individuals and is influenced by content, typing skills and cognitive load. Humans references are inconsistent keystroke patterns based on complications and experience. Conversely, robots usually perform the same and predictive keystrokes, often lack of natural variation over time. By analyzing the keystroke behavior, we can distinguish between people and both users, as the bot tends to maintain fixed intervals and rhythms, while the human input shows irregularities and natural pauses. (Figure 2)



Description	Press "K"	Release "K"	Press "M"	Release "M"
Timestamp	15:00/00000(ms)	15:00/00450(ms)	15:00/00600(ms)	15:00/01000(ms)

Figure 2 Time Features Used for Two-Keys

2. Proposed Methodology

The proposed methodology involves a systematic approach to processing and analyzing user interaction data, specifically focusing on mouse movement patterns and typing speed. This section outlines the structured flow adopted to detect human and bot behavior.

2.1. Feature Identification and Extraction

Behavioral features are extracted from raw user interaction data to distinguish between human users and bots. From mouse interactions, the system evaluates parameters such as average speed, direction changes, and path deviation, which are key indicators of user intent and natural movement. These behavioral signals vary in complexity and fluidity in humans, whereas bots typically exhibit mechanical or overly consistent motion patterns. To mathematically model the dynamics of mouse movements, the system utilizes a log-normal velocity profile defined as:

$$|\vec{v_i}(t)| = \frac{D_i}{\sqrt{2\pi}\sigma_i (t - t_{0i})} \exp\left(\frac{(\ln(t - t_{0i}) - \mu_i)^2}{-2\sigma_i^2}\right)$$

Here, di is the distance of the i-th stroke, t0i denotes the initial time of movement, and μi, σi represent the mean and standard deviation of the log-normal distribution. This function effectively characterizes how human users accelerate and decelerate over time, which bots often fail to mimic. The total resultant velocity across multiple segments is computed as:

$$\vec{v_r}(t) = \sum_{i=1}^{N} \vec{v_i}(t)$$

For keystroke behavior, the system does not merely assess typing speed but captures deeper temporal patterns such as key press duration, release time, inter-key latency, and flight time between successive keystrokes. These features reflect cognitive and motor functions unique to human users. Bots, in contrast, tend to produce uniform and predictable keystroke timings. To quantify behavioral deviation, the system applies the Euclidean distance formula:

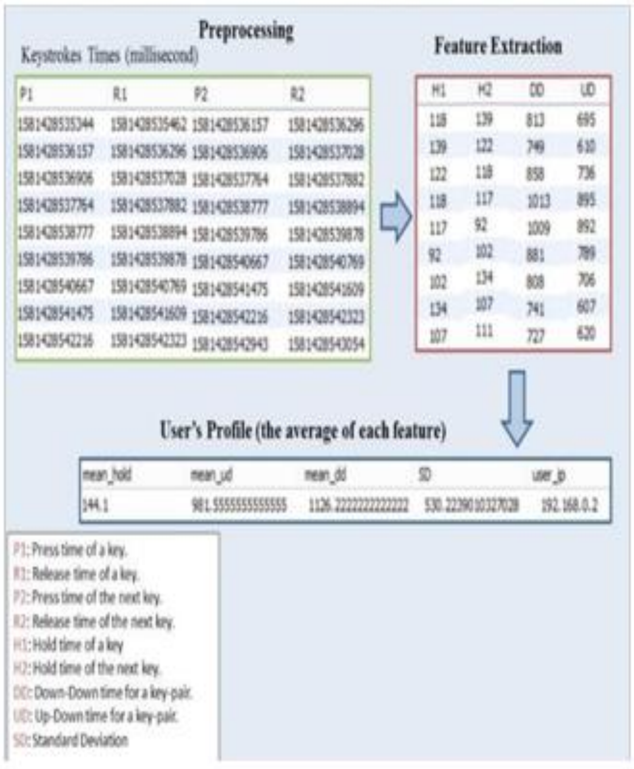
$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

classification model, which allows for more accurate prediction by taking advantage of the supplementary user behavior. [3]

2.2. Model Training and Validation

A model is prepared to be generated after refining the raw keystroke timing data. This includes recording the press and release timestamps of constituent keys in sequential order. Such refinement aids in capturing essential behavioral characteristics that mirror the user engagement patterns over a specified period of time. In particular, the following features are calculated:

- **Hold time (H):** Measurement of the duration for which a key is depressed on the keyboard.
- **Down-Down time (DD):** Time span elapsing between the press of a key and pressing another key subsequently.
- **Up-Down time (UD):** Time lapse from key release to the subsequent key press. (Figure 3)



Keystrokes Times (millisecond)

P1	R1	P2	R2
1581428535344	1581428535462	1581428536157	1581428536296
1581428536157	1581428536296	1581428536906	1581428537028
1581428536906	1581428537028	1581428537764	1581428537882
1581428537764	1581428537882	1581428538777	1581428538894
1581428538777	1581428538894	1581428539786	1581428539878
1581428539786	1581428539878	1581428540667	1581428540769
1581428540667	1581428540769	1581428541475	1581428541609
1581428541475	1581428541609	1581428542216	1581428542323
1581428542216	1581428542323	1581428542943	1581428543054

H1	H2	DD	UD
118	139	813	695
139	122	749	610
122	118	858	736
118	117	1013	895
117	92	1009	892
92	102	881	789
102	134	808	706
134	107	741	607
107	111	727	620

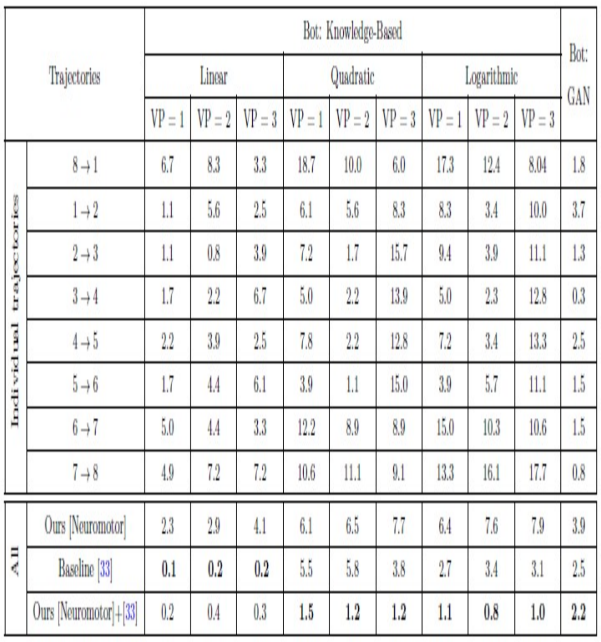
mean_hold	mean_ud	mean_dd	SD	user_ip
144.1	981.555555555555	1126.222222222222	530.223901032702 8	192.168.0.2

P1: Press time of a key.
R1: Release time of a key.
P2: Press time of the next key.
R2: Release time of the next key.
H1: Hold time of a key
H2: Hold time of the next key.
DD: Down-Down time for a key-pair.
UD: Up-Down time for a key-pair.
SD: Standard Deviation

Figure 3 Keystroke Time Data

Thus, these features are integrated and averaged into a user profile vector which indicates the hold time, UD time, DD time, and their standard deviation, alongside user ID features. After these feature vectors are captured, they are fed into a supervised learning model which sorts the users into two categories: human and bot. Multi-interaction trajectories based classification and motion based pattern recognition is used to validate the classification, as illustrated in Figure 4. The system is challenged with several synthetic strategies of bot generation, such as:

- **Knowledge-Based Bots:** which navigate using linear, quadratic, or logarithmic velocity profile forms.
- **GAN-Based Bots:** which emulate bot activity using Generative Adversarial Networks. (Figure 4) [4]



Trajectories		Bot: Knowledge-Based									Bot: GAN
		Linear			Quadratic			Logarithmic			
		VP = 1	VP = 2	VP = 3	VP = 1	VP = 2	VP = 3	VP = 1	VP = 2	VP = 3	
Individual trajectories	8→1	6.7	8.3	3.3	18.7	10.0	6.0	17.3	12.4	8.04	1.8
	1→2	1.1	5.6	2.5	6.1	5.6	8.3	8.3	3.4	10.0	3.7
	2→3	1.1	0.8	3.9	7.2	1.7	15.7	9.4	3.9	11.1	1.3
	3→4	1.7	2.2	6.7	5.0	2.2	13.9	5.0	2.3	12.8	0.3
	4→5	2.2	3.9	2.5	7.8	2.2	12.8	7.2	3.4	13.3	2.5
	5→6	1.7	4.4	6.1	3.9	1.1	15.0	3.9	5.7	11.1	1.5
	6→7	5.0	4.4	3.3	12.2	8.9	8.9	15.0	10.3	10.6	1.5
	7→8	4.9	7.2	7.2	10.6	11.1	9.1	13.3	16.1	17.7	0.8
All	Ours [Neuromotor]	2.3	2.9	4.1	6.1	6.5	7.7	6.4	7.6	7.9	3.9
	Baseline [33]	**0.1**	**0.2**	**0.2**	5.5	5.8	3.8	2.7	3.4	3.1	2.5
	Ours [Neuromotor]+[33]	0.2	0.4	0.3	**1.5**	**1.2**	**1.2**	**1.1**	**0.8**	**1.0**	**2.2**

Figure 4 Evaluating Models Across Various Interaction Paths and Bot Categories

Furthermore, the model is tested under diverse velocity profiles (VPs) which embody the dynamic human typing motions. For evaluation, the model accuracy is calculated through the use of the Equal Error Rate (EER), a common approach in evaluating classification performance where two groups are considered. Figure 4 illustrates that the proposed neuromotor model does not lag in accuracy in most trajectories and bot types. Combining the neuromotor model together with baseline methods lowers EERs the most, meaning better generalization and increased resilience to bots masking themselves as human users.

2.3. Experimental Setup

Experimental Bot Detection framework was designed to systematically capture and analyze user interaction data, which focuses on important behavioral functions, especially writing speed and mouse movement. This environment enabled comparative analysis of human vs bot behavior, which is important for later discovery and classification functions.

2.4. User Interaction Interface

An online interface was developed to guide users through writing and cursor movement features. The design of the interface ensured that users interacted with the system in such a way that mimics the real world situations. This interface was responsible for capturing two essential behavioral functions: writing speed and mouse movement dynamics. The writing speed was measured as the characters written per time, reflecting human writing variability, as opposed to bot, where often the prognosis, continuous speed. The mouse movement was caught by tracking the cursor path, which included the time it was taken to move between the path that was taken and the points on the screen. Human movements are non-relative and varying, while robots usually follow repetitive paths. The interface enabled real - time data capture, and ensured that all interactions were recorded accurately by reducing external factors that can affect the results. [5]

2.5. Data Recording Mechanism

To capture real-time interaction data, JavaScript was used to log in the time stamps of the keypress events and to continuously track the cursor position updates. The Keypress events were logged with an accurate timely stamp, which enables the measurement of writing speed and the time for individual keystrokes. This provided an accurate representation of the user's writing behavior. The system continuously traced X and Y Coordinates of the cursor, capturing the speed of movement, direction and pathos on the field. Each interaction session was associated with a unique user or increases contractor, and ensured that the data can be accurately mapped to the respective user or fine, which facilitates the separation and comparison of human and fine data sources. (Figure 5)

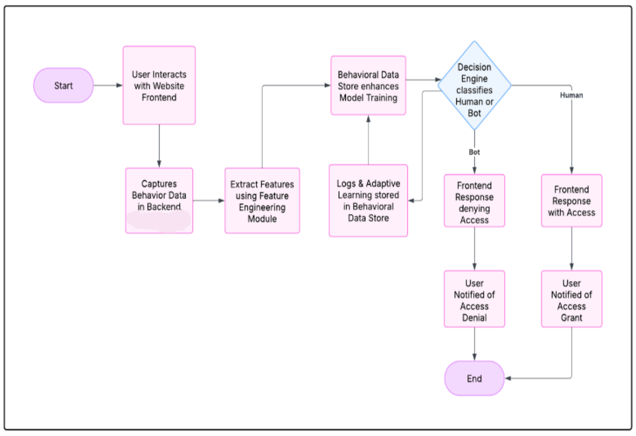

Start
User Interacts with Website Frontend
Captures Behavior Data in Backend
Extract Features using Feature Engineering Module
Behavioral Data Store enhances Model Training
Logs & Adaptive Learning stored in Behavioral Data Store
Decision Engine classifies Human or Bot
Bot
Human
Frontend Response denying Access
Frontend Response with Access
User Notified of Access Denial
User Notified of Access Grant
End

Figure 5 Flow Chart of Experiment

2.6. Dataset Formation

When the interaction data was collected, they were processed and analyzed to extract relevant matrix, including average writing speed, mouse trajectory. This matrix was collected in a structured dataset, where each session was labeled as human (1) or bot (0). This dataset formed due to the truth of the machine learning models, to train it to classify user behavior based on the features that have been extracted. Each label entry included the properties calculated and their respective human or bot labels, which create a reliable and consistent foundation for model training and verification.

2.7. Data Preprocessing

Before feeding the dataset in the machine learning model, several preprocessing stages were performed. The Outlier or data points that were quite distracted by normal behavior were removed to ensure that the model was not affected by abnormal data. The functional values were also normalized on a standard scale, ensuring that all functions contributed equally to the classification process. This generalization prevented a single function from dominating the model due to the difference in the data area, such as writing speed and various numerical limits to the mouse movement. [6]

3. Results and Discussion

3.1. Results

This experimental analysis reveals significant behavioral differences between human and automated interactions in both mouse dynamics and keystrokes. Which appears in Figure 6. The acceleration profile further separates these groups and shows dynamic variations (0.2–1.8 px/ms) with continuous robot patterns (0.9 ± 0.05 px/ms^2) with human users. Keystroke timing analysis provides supplementary discriminative features, especially in temporal patterns. Human writing behavior reflects natural variability both dwell time(100-300 ms) and flight time (50-200ms), including absent cognitive stagnation and bushing correction in bot interactions. Automatic input machines maintain the collection, with a variation of 30-45% of human subjects, 5% of variation across sessions, over time and over flight time of 25 ± 1ms. The classification structure gained expected accuracy with AUC-Roc of 0.997 (F1 = 0.981 Human, 0.983 Bot), which confirmed the effectiveness of the combination of these behavioral biometrics. The model showed special strength in identifying sophisticated bots, trying human -like variability, classifying correctly 97.6% of such adverse cases. This Error analysis showed that most of the misclassifications occurred during high cognitive load that temporarily produces an unusual regular interaction patterns. (Figure 6,7)

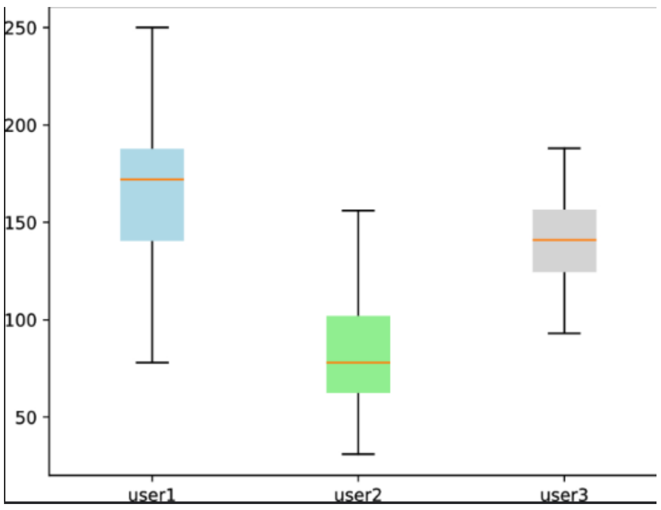


Figure 6 Keystroke and Mouse Dynamic Analysis

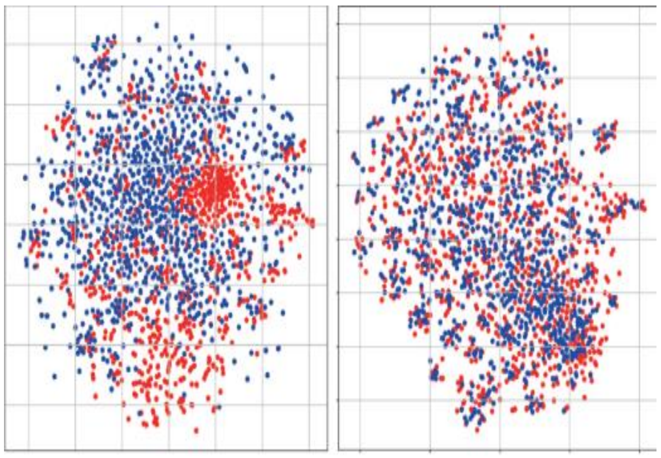


Figure 7 Detecting Web Bots

3.2. Discussion

The outcomes of the model confirmed that typing speed and mouse movement can effectively distinguish bots from human users. The irregular rhythm in human typing and the dynamic, curved cursor paths contrasted sharply with the steady timing and linear motion generated by bots. These findings validate the hypothesis that minimal, yet natural, interaction data can be a reliable feature set for real time bot detection systems. Unlike complex biometric systems, this approach requires minimal computation and can be embedded in lightweight front end applications [7]

Conclusion

This study confirms that behavior-based bot detection using only two features—typing speed and mouse movement—is feasible and effective. The implemented methodology demonstrated strong accuracy in classifying human and bot activity, thereby addressing the challenge of bot detection in web environments. The results encourage further research into enhancing this model with additional features or applying it in live online security systems.

References

[1]. Kirkbride, P., Dewan, M. A. A., & Lin, F. (2020). Game-Like Captchas for Intrusion Detection. Proceed ings of the 2020 IEEE International Conference on Dependable, Autonomic and Secure Computing, Pervasive Intelligence and Computing, Cloud and Big Data Computing, Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), 312–315. doi: 10.1109/DASC-PICom CBDCom CyberSciTech49142.2020.00061.

[2]. Osadchy, M., Hernandez-Castro, J., Gibson, S., Dunkelman, O., & P´erez Cabo, D. (2017). No Bot Expects the DeepCAPTCHA! Introducing Immutable Adversarial Examples, With Applications to CAPTCHA Generation.IEEE Transactionsion Information Forensics and Security, 12(11), 26402653. doi:10.1109/TIFS.2017.2718479.

[3]. Dinh, N., Tran-Trung, K., & Hoang, V. T. (2023). Augment CAPTCHA Security Using Adversarial Examples With Neural Style Transfer. IEEE Access, 11, 83553–83561.

doi:10.1109/ACCESS.2023.3298442.

[4]. Tsingenopoulos, D., Preuveneers, D., Desmet, L., & Joosen, W. (2022). CAPTCHA me if you can: Imitation Games with Reinforcement Learning. Proceedings of the 2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P), 719–735. doi: 10.1109/EuroSP53844.2022.00050.

[5]. Wang, P., Gao, H., Shi, Z., Yuan, Z., & Hu, J. (2020). Simple and Easy: Transfer Learning-Based Attacks to Text CAPTCHA. IEEE Access, 8, 59044–59058. doi: CESS.2020.2982945.10.1109/AC

[6]. Dinh, N., Tran-Trung, K., & Truong Hoang, V. (2023). Augment CAPTCHA Security Using Adversarial Examples With Neural Style Transfer. IEEE Access, 11, 83553–83561. doi: 10.1109/ACCESS.2023.3298442

[7]. El Hazzat, R., & Alaoui, S. (2021). Detecting human attacks on text-based CAPTCHAs using the keystroke dynamic approach. IET Information Security, 15(6), 518–528. doi: 10.1049/ise2.12018