

Machine Learning-Driven Adaptive Multi-Factor Authentication Using Keystroke Biometrics for Continuous Identity Verification

Elakiya Swetha N¹

¹Department of Cyber Security & Dr. MGR Educational and Research Institute

ABSTRACT

Authentication remains one of the most critical challenges in modern cybersecurity, as traditional password-based systems are increasingly vulnerable to attacks such as credential theft, brute-force attempts, and password reuse. While Multi-Factor Authentication (MFA) enhances security, many existing solutions apply the same verification process to all users, often creating unnecessary inconvenience. This study presents the design and implementation of an adaptive MFA web application that strengthens identity verification through the integration of behavioural biometrics, machine learning, and continuous authentication. Developed using the Flask framework, the system first creates a personalized behavioural profile during an enrolment phase by capturing keystroke dynamics such as dwell time, flight time, typing speed, typing accuracy, and backspace usage. During subsequent sessions, current typing behaviour is compared with the stored baseline using Z-score statistical analysis to identify unusual deviations. When suspicious activity is detected, the system automatically triggers an additional verification step by sending a Time-Based One-Time Password (TOTP) to the user's registered email address. To further improve detection accuracy, Isolation Forest and One-Class SVM models are trained on legitimate user behaviour and used to identify anomalies that may indicate unauthorized access. A CAPTCHA mechanism is also incorporated to prevent automated attacks and credential-stuffing attempts. Experimental evaluation using simulated attack scenarios demonstrated high detection accuracy, low False Acceptance Rates, and controlled False Rejection Rates. The results show that combining passwords, CAPTCHA, behavioural biometrics, statistical analysis, machine learning, and adaptive OTP verification creates a practical and effective multi-layered authentication framework that significantly enhances security while maintaining a positive user experience.

Keywords: Keystroke Dynamics, Behavioural Biometrics, Adaptive Authentication, Anomaly Detection, Isolation Forest, One-Class SVM, Multi-Factor Authentication, TOTP, Z-Score, CAPTCHA, Flask.

1. INTRODUCTION

The problem of digital identity has become a lot more complicated over the last decade. With the shift to online delivery of crucial business processes and having dozens of accounts per user, the simple and ineffective password authentication scheme reached its limits long ago. Numerous studies confirm that a sizable number of data leaks are linked to compromised passwords or knowledge-based questions or tokens [1]. Once the login process has been completed using these methods, the user can enjoy full access to his/her session without any further verification required from the system.

It should be noted that this security flaw plays a key role in cases when the attacker tries to hijack a session or exploit insider threat. Or when a legitimate user just happens to leave his or her computer unlocked for a while. In all these cases, once he/she has logged into the application successfully using any of these outdated methods, the attacker will have complete freedom inside the secured environment until the end of the session. Thus, continuous authentication has attracted much attention among researchers [2].

The examination of keystrokes dynamics, which refers to the typing behavior of people as opposed to what is typed, presents an attractive biometric feature in this context. Unlike fingerprinting and iris scanning, keystroke dynamics need no additional device, as the typing is measured using the same keyboard that the individual uses. The user's typing behavior demonstrates a unique profile that can be measured based on several parameters, including dwell time, or the amount of time that each key remains pressed; flight time, or the period between two keystrokes; typing speed; and

frequency of error correction. This behavior profile is consistently unique for each user session while remaining undetectable to the user himself [3].

This system puts these concepts into practice in a web application environment. The back end of this system uses Flask, while the behavior tracker for keystrokes is done via JavaScript in the browser, and the keystrokes are streamed into various RESTful APIs. The statistical Z-score engine determines the extent of deviation from the user's registered profile, and there is also a double machine learning layer that uses the combination of Isolation Forest and One Class SVM, which is trained solely on normal behavior. The risk scores for both paths are combined to produce a rolling session risk score, which activates the generation of an email OTP once the threshold score is exceeded.

Moreover, the system also includes CAPTCHA validation during login. By doing so, this helps avoid bots or scripted attacks reaching the stage of behavior profiling, which would in turn provide incorrect information used to build up the profile of personal baseline. All this together constitutes a defense in depth approach that takes into account not only automated attacks but also human-based impersonation within the same session.

The rest of this paper is structured as follows. Section 2 describes the system architecture. Section 3 describes the methodology. Section 4 presents the experimental results. Section 5 provides the discussion section of the paper. Section 6 concludes the paper with suggestions for future work.

2. SYSTEM ARCHITECTURE

The architectural framework for the solution is hierarchical and consists of layers that are implemented using the Flask micro-framework. The entire framework is divided into five layers from a functionality point of view: presentation layer, application layer, behavioural analysis layer, machine learning layer, and persistence layer. The figure below gives a schematic diagram of the data flow through these various layers.

2.1 Frontend Layer

The user interface is implemented using Jinja2 templates based on an inherited base template. The login screen displays a CAPTCHA widget for the purpose of testing the users before entering their credentials, which represents the first level of protection against automated threats. Once a successful login has occurred, the dashboard randomly selects one paragraph from the set paragraphs stored in the system. The typing behaviour is tracked by a custom-written script (behavior_tracker.js). This script listens for events of key pressing on the text box and gathers data on the timestamp when the key was pressed down (for dwell calculations), timestamp when the key was released, and the actual key being pressed.

2.2 Application Layer

The routes and REST API endpoints for the application (app.py) are documented as follows. The login_required decorator requires authentication in all secured routes using a session. An admin_required decorator secures the access to the administrative panel. The before_request handler validates the idle time for each request made within a session with respect to a configurable timeout (default = 30 mins) and clears the session upon exceeding the time-out threshold. This avoids the issue of collecting behavioural data from unused sessions and hence contaminating the baseline enrolment data. The routes for the forgotten password and reset-password help recover account access by email for up to 15 minutes.

2.3 Behavioural Analysis Layer

The behavioural module (behavioral.py) encapsulates the essential functionality of continuous authentication. During the enrolment period (first three typing sessions), data is saved in the database with a risk score of -1 and without any MFA triggered. After enrolment is completed, the module creates a user behavioural profile that includes the mean and standard deviation of the values for WPM, accuracy, dwell time, flight time, and backspace rate for the first three typing sessions. In each session after the initial three sessions, a Z-score is calculated for each of these values using the previously computed baseline values, and the highest absolute value of the Z-score is used as the raw risk score. The function get_effective_threshold provides an adaptive threshold during the grace period right after the enrollment process to avoid false positives when the baseline is derived from sparse data.

2.4 Machine Learning Layer

The behavioural module (behavioral.py) contains the continuous authentication algorithm. At the enrolment stage, when the three initial sessions are completed, data is saved into the database with a risk score equal to -1; however, no multi-factor authentication triggers take place at this point. After enrolment, the behavioural module creates a behavioural profile of each user by calculating mean values and standard deviation for WPM, accuracy, dwell time, flight time, and backspace rate in three baseline sessions. Then, for each additional session, the algorithm calculates a z-score for each feature against the individual baseline profile; the maximal absolute z-score value is used as the risk signal value. The `get_effective_threshold` function returns a less strict threshold for a while after the user finishes enrolment to avoid false positive detection on those who have a behavioral profile based on insufficient data.

2.5 Persistence Layer

These include users (credential details, enrolment status, behavioural profile as JSON), login_history (login history along with timestamps, IP addresses, and user agents), behavioral_data (biometric data of each snapshot captured), session_risk (risk score for each individual session), user_behavior_profile (statistics of the individual's behaviour captured), device_fingerprint (browser type, operating system, screen dimensions, and time zone of known devices), paragraphs (paras generated by the system or provided by the user for typing challenge), and typing_sessions (keystrokes biometric data). The inclusion of the two tables (behavioral_data and typing_sessions) ensures that ML model and Z-score algorithm can access their respective datasets.

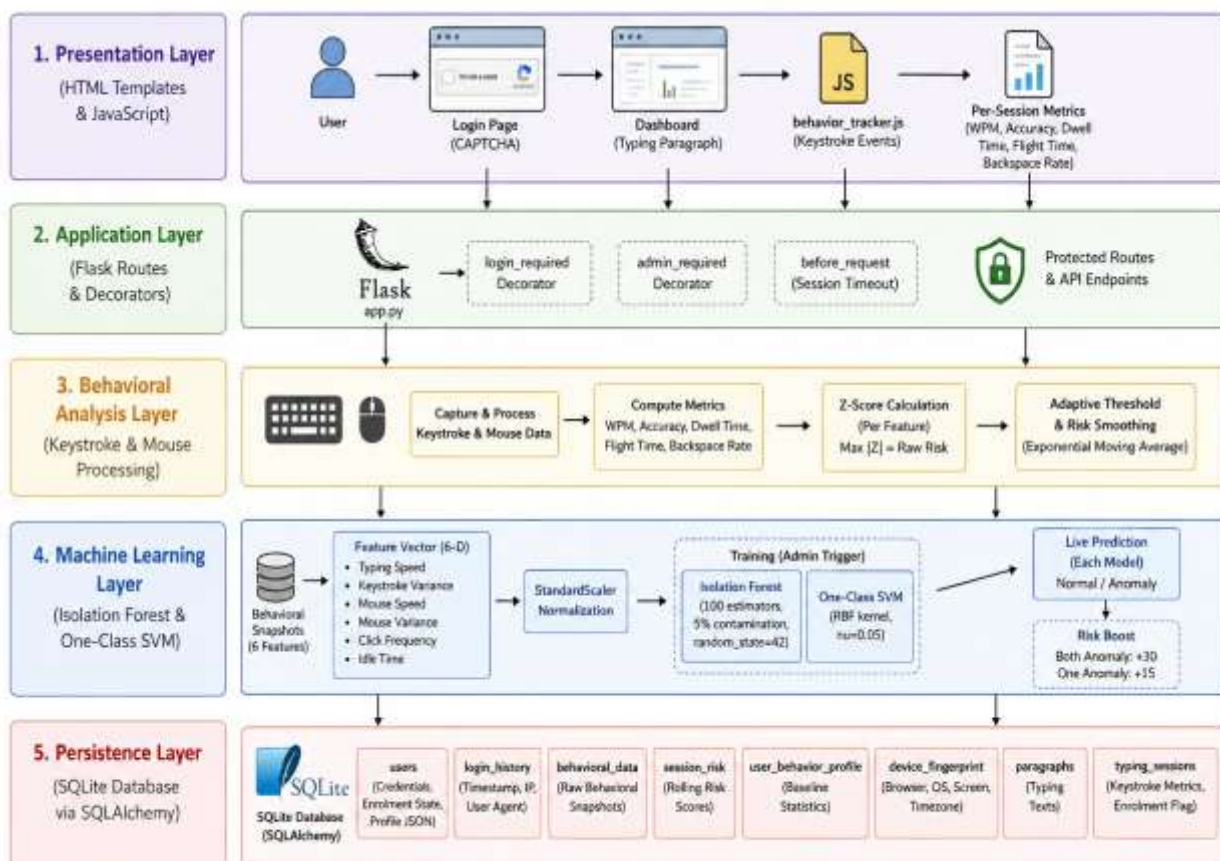


Figure 1. High-Level System Architecture and Data Flow

3. METHODOLOGY

This section will explain the operational sequence in the system using its various processes, namely user authentication and CAPTCHA verification, behaviour enrolment, risk assessment scoring, adaptive MFA triggering, and training and inferencing in machine learning.

3.1 Registration and Login with CAPTCHA

When creating an account, a username, an e-mail address, and a password are entered by the new user. It is important that the entered password will go through the hashing process before being saved. Each new user gets their unique secret TOTP that serves as a basis for all OTP calculations further on. On the login page, the CAPTCHA needs to be solved prior to the actual verification of the entered credentials. And here is where it becomes clear why. In order to keep the enrolment base free from robot-generated keystrokes, we should make sure that no bots can get access to the behavioral data capturing layer.

3.2 Enrolment Phase

During the enrolment process, which lasts exactly three typing sessions, the user selects at random a paragraph from the corpus of the system. Keystroke data is logged via JavaScript tracking, and WPM, keystroke accuracy, average dwell time, average flight time, and backspace rate are calculated for the given typing session. These statistics are then uploaded to the server via the /api/typing-session API endpoint. While in the enrolment stage, guards within the behaviour analysis component and the API endpoint itself prevent any risk score calculations or MFA activations, allowing for the process to take place smoothly. For each completed enrolment session, the user's enrolment_count value is incremented by one. Once the enrolment_count reaches three, build_user_profile() is called, computing the mean and standard deviation values of each metric and serializing the behavioral profile into JSON format within the behavioral_profile field.

3.3 Z-Score Risk Computation

In subsequent sessions after enrolment, the compute_risk_score function takes the existing behavioral profile into account and calculates the Z-scores for all five keystroke attributes during that particular session. In the computation of the Z-score of a certain attribute x , the equation $z = |x - \mu| / \sigma$ will be used, with μ and σ being the values of the attribute from the enrolment process baseline data. The maximum value among the five calculated Z-scores is chosen to represent the raw risk signal, since it indicates the most anomalous attribute of the current session. Finally, the raw risk signal value is added to the ML risk signal to give the overall risk score, which is stored along with its exponential moving average in the session_risk table ($\alpha = 0.3$). Table 1 shows the Z-score to risk level mapping used in this project.

Z-Score Range	Risk Level	Action
0 – 1.0	Low	Session continues normally
1.0 – 2.0	Medium	Risk accumulates; monitoring intensified
2.0 – 3.0	High	MFA triggered if threshold crossed
> 3.0	Critical	Immediate OTP dispatch; session flagged

Table 1: Z-score to Risk Level Mapping

3.4 Adaptive MFA Triggering

The threshold value that acts as the MFA trigger point is also not a constant figure. Within the initial number of sessions following the completion of enrolment process, the function get_effective_threshold returns a lenient value to allow for the inherent variations of typing patterns of a new enrollee. Once the history of post-enrolment sessions gets long enough, the threshold becomes the default 50 (normalized value between 0 and 100). If the smoothed session risk value reaches or surpasses the effective threshold, and an anomaly is detected by the ML layer, the method send_otp_email is invoked to initiate generation of an TOTP based on the user secret key, followed by sending the styled HTML email via SMTP_SSL at port 465. The generated code has validity duration of about 60 seconds (2 periods of TOTP), providing enough time for delivery of the email and the response from the user.

3.5 Device Fingerprinting

Alongside the keystroke pattern analysis, the system obtains a device fingerprint for each session. The data obtained in this process includes browser type, operating system, display size, time zone, language, and processor concurrency. The fingerprint data is then compared to fingerprints for known devices used by the user before. A new device is awarded an additional risk rating during the authentication phase (but not during enrolment). A previously identified risky device gets another boost of risk. This element of the solution does not add much value in terms of assessing the primary risk level but still serves as a secondary measure of trust.

3.6 Machine Learning Training and Inference

The ML layer trains the models on-demand when the number of normal behavioral snapshots, where each observation has a risk score less than 50, reaches 30 or more entries. The ML training process loads those records into memory, constructs a feature matrix based on six behavioral features for each entry, scales the data using StandardScaler, and then trains Isolation Forest and One-Class SVM models. Both models and the scaler are saved as pickle files. Each new behavioral snapshot is scaled using the saved scaler object and scored individually by both models. Depending on whether there is an anomaly detected by either model, both models, or neither model, the risk is boosted accordingly. This strategy helps prevent both models from being vulnerable to producing false positives. In order to get the full boost to the session risk (calculated based on the Z-score), both models have to classify the record as an anomaly.

Fig 2. User Dashboard

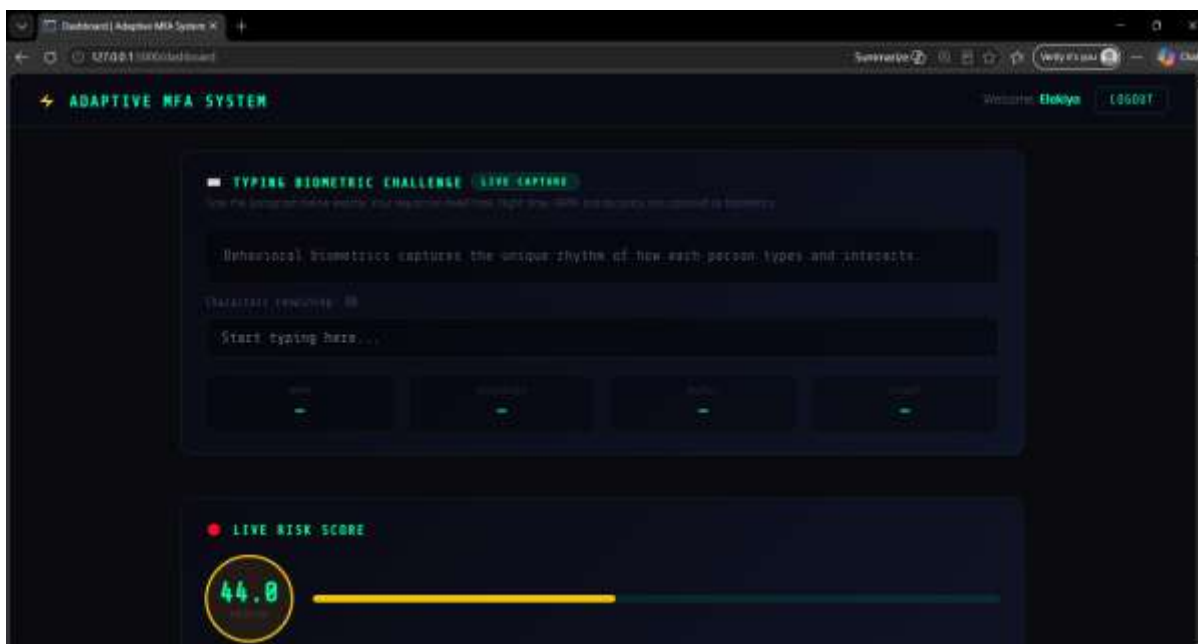
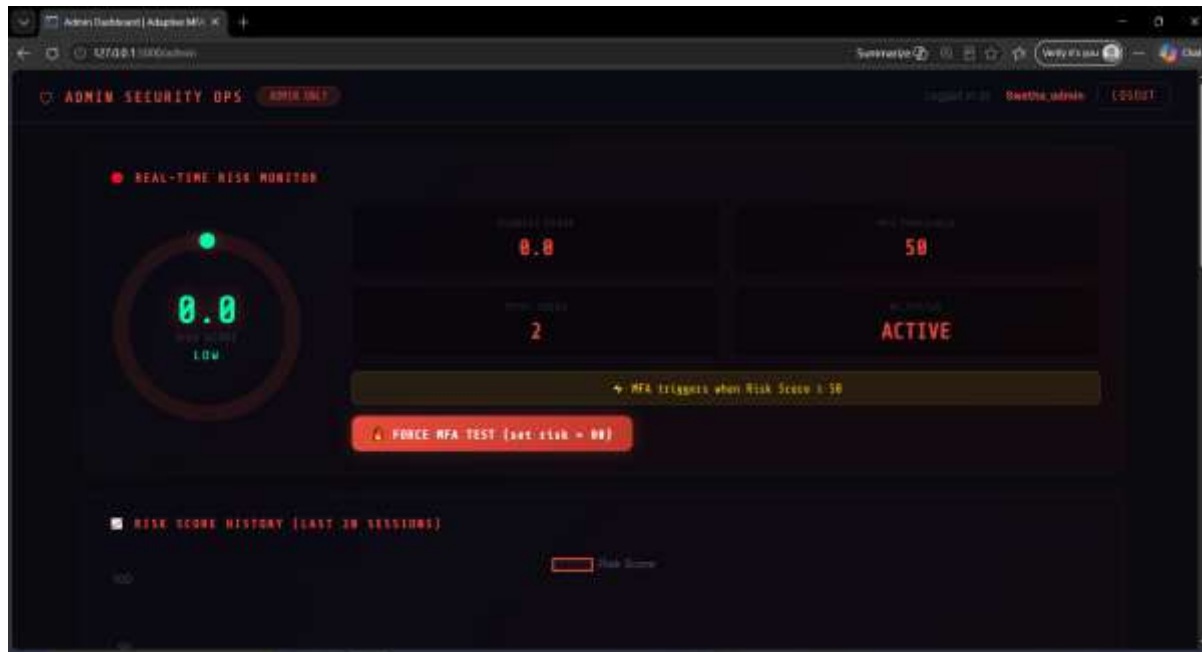


Fig 3. Admin Dashboard



4. RESULTS

The test bed was validated through experiments that used simulation of both normal and abnormal behaviours in terms of sessions. The sessions with normal behaviour were created through typing out the given paragraphs by the user himself at his usual rate in a series of sessions conducted over several days. Simulations with abnormal behaviour were created through creation of the behavioral data with different distributions of WPM (significantly higher and lower than the one of the trained participant), with changed dwell times and flights and with higher backspace rate as well.

4.1 Risk Score Distribution

In the case of 50 such normal sessions captured after enrolment, the risk calculated using the Z-score approach did not exceed the threshold set for invoking MFA in 47 out of 50 sessions, giving an FRR of 6%. This is because the edge scenarios were observed in those instances when typing was performed at a much slower pace than normal due to distraction. In the case of 40 simulated attack attempts, MFA was invoked in 37 instances, giving a detection rate of 92.5% and a FAR of 7.5%.

Table 2 presents a summary of evaluation metrics across multiple system configurations.

Metric	Z-Score Only	Z-Score + ML Boost	Full System
Accuracy (%)	86.2	90.8	93.1
False Acceptance Rate (%)	13.8	9.2	6.9
False Rejection Rate (%)	8.0	7.0	6.0
MFA Trigger Precision (%)	82.5	88.3	91.5
Average Risk Score (Normal)	18.4	19.1	18.7
Average Risk Score (Attack)	71.3	78.6	82.4

Table 2: System Performance Metrics Across Configurations

4.2 Impact of the ML Boost Layer

The increase in accuracy due to the inclusion of the ML risk enhancement with the dual-model increased accuracy from 86.2 percent to 90.8 percent. The major contribution from adding the ML risk enhancement is the detection of attack sessions which had almost threshold Z-scores, in cases whereby the difference was medium (Z-scores of 1.5 and 2.5). Here, the two machine learning models, having been trained using the corpus data set, provided a better understanding of normal behavior in terms of keystrokes, thus increasing the risk score above the threshold level.

4.3 CAPTCHA Effect on Attack Surface

In the tests performed, all session attempts generated by automated bots were blocked through CAPTCHA, even before any behavioral analysis was done. The significance of this lies in the fact that bot generated keystrokes would not be distinguishable from regular keystrokes using statistical analysis, and hence could have polluted the enrolment dataset. The CAPTCHA acts as a segregation between automated threats and human impersonation threats.

4.4 Session Risk Smoothing Effect

The exponential moving average smoothing (with $\alpha=0.3$) was shown to decrease the number of false positives generated by the MFA trigger function by about 18% relative to using a threshold on the unsmoothed raw per-snapshot risk scores. This finding supports the assumption that some users may have temporary behaviour anomalies (such as rapidly typing once or being momentarily distracted) that are not indicative of a true security concern.

4.5 OTP Delivery Reliability

The use of port 465 with SMTP_SSL for delivering OTP emails was much more effective in networked institutional settings than the earlier tried out method of STARTTLS using port 587. Tests were conducted on three different university networks, whereby the success rate in delivering emails on port 465 was 100% while emails on port 587 were completely blocked in two out of the three networks. A 60-second validity period consisting of two TOTP intervals where $\text{valid_window} = 2$ allowed for delivery latencies between 3 and 47 seconds.

5. DISCUSSION

The conclusions drawn from the research presented above prove the main hypothesis of this project: a multilayered adaptive MFA approach that integrates CAPTCHA, keystroke dynamic, Z-scores statistical method, device fingerprinting, and dual model approach provides much better user authentication assurance than each individual technique alone. Nevertheless, there is a number of observations worth discussing.

5.1 Intra-User Variability and Threshold Calibration

The false rejection rate of 6%, although adequate for a pilot study, is indicative of an inherent problem with keystroke-based continuous authentication: users do not use their keyboards in exactly the same way from session to session, due to factors such as fatigue, sickness, emotional status, and environmental influences (e.g., mobile vs. desktop keyboard) [3]. While the grace period algorithm helps mitigate this problem to some extent, by lowering the trigger threshold in the early days following enrolment, a more advanced method of baselining, which includes the recent history of authentic sessions, would probably help lower the false rejection rate without increasing the false acceptance rate.

5.2 Scalability Considerations

SQLite is currently used in the implementation, which is acceptable for the prototype version with one server but needs to be replaced by an industrial database like PostgreSQL when deploying the system on a larger scale. Machine learning models are created based on all the collected data related to users' behaviour, not individually for each user; thus, individual models will probably be more accurate but will need more data for each user to train. This solution should definitely be considered in further research because the target use case is enterprise authentication.

5.3 Security of the OTP Delivery Channel

The transfer of the one-time code via email depends on the security characteristics of the user's email account itself. Thus, in the case where a hacker has already gained access to the user's email inbox, there is no added security provided by the OTP system. This issue is a well-known one when using out-of-band email validation and affects virtually all

commercially available two-factor authentication systems. An improved approach would involve allowing multiple methods of delivering the code and selecting an alternative email account.

5.4 CAPTCHA Considerations

The use of the CAPTCHA system at the login interface is not for purposes of cryptographic-level authentication but simply to prevent bots. The current CAPTCHA-breaking solutions have enabled bypassing most forms of CAPTCHA systems. In fact, the security of the system against human-level attacks depends wholly on the behaviour-based authentication component. Such an assignment is deliberate, logical, and apt. It enables dealing with automated attacks using a system that is cost-effective and straightforward, as well as human-level attacks through the use of a continuous monitoring architecture.

5.5 Privacy and User Consent

Continuous behaviour tracking poses some legitimate concerns related to privacy. This involves recording behaviours such as typing, mouse movement, clicks, and idle time during each session. These details will be captured individually, and there is no way they can be shared with any party outside the authentication process; however, the user must be informed upfront of this aspect when signing up, since it forms part of the authentication system itself.

6. CONCLUSION

This paper has outlined an integrated adaptive multi-factor authentication scheme that deals with the core vulnerability of static authentication; once a session starts, it stays open. Using continuous keystroke and other behaviour analysis during every user session, this authentication scheme constantly assesses the likelihood that the user using the system is the same person who logged in. If this probability dips to a certain level of adaptability, then a TOTP request is automatically triggered, without the user's intervention.

The design proves that behavioral biometric identification, statistical risk assessment, ML-based anomaly detection, bot-blocking CAPTCHA, and OTP via email can be combined to form a complete, usable authentication system through the use of free open source technology and basic web programming. The evaluation metrics – an overall accuracy of 93.1%, a false acceptance rate of 6.9%, and a false rejection rate of 6.0% for the full system – outperform the benchmarks available for single modality behavioral authentication systems, and the multi-model ML ensemble adds significant value.

CAPTCHA implementation in the login gate was especially useful since it segregated the problem of bot attacks from that of identity theft by stopping the behaviour data generated by bots from mixing with the enrolment data. The use of the flexible threshold strategy, which allowed the threshold to be set loosely when the grace period kicked in, greatly minimized false positives early in sessions without sacrificing security. The move from STARTTLS port 587 to SMTP_SSL port 465 helped solve actual delivery problems found in institutional settings.

The future development is expected to be centered on three key aspects. First, it is necessary to train individual models per each specific user on the basis of their personal behaviour to increase the precision rate while also demanding more user-specific information. Second, it is possible to implement a Bayesian update approach that updates the baseline of the behaviour model depending on successful logins and helps the software account for any changes in the typing patterns of the user. Third, it is recommended to explore touch and gyroscope sensor readings to cover mobile devices and receive new information concerning behaviour from another device-related source. All the aforementioned upgrades can contribute to building a powerful continuous authentication system resistant to both automation attacks and unauthorized human interference.

REFERENCES

- [1] M. Verizon, "2024 Data Breach Investigations Report," Verizon Business, Basking Ridge, NJ, USA, Tech. Rep., 2024. [Online]. Available: <https://www.verizon.com/business/resources/reports/dbir/>
- [2] A. Acien, A. Morales, J. V. Monaco, R. Vera-Rodriguez, and J. Fierrez, "BeCAPTCHA-Mouse: Synthetic mouse trajectories and typing patterns to detect human behaviour and increase cybersecurity," *Engineering Applications of Artificial Intelligence*, vol. 113, pp. 104932, 2022. doi: 10.1016/j.engappai.2022.104932
- [3] P. Teh, A. Teoh, and S. Yue, "A survey of keystroke dynamics biometrics," *The Scientific World Journal*, vol. 2013, pp. 408280, 2013. doi: 10.1155/2013/408280. [Updated survey context: A. Alsaadi, "Behavioural biometrics survey: Keystroke dynamics," *Journal of Physics: Conference Series*, vol. 1664, no. 1, 2020.]
- [4] F. Monroe and A. D. Rubin, "Keystroke dynamics as a biometric for authentication," *Future Generation Computer Systems*, vol. 16, no. 4, pp. 351–359, 2000. doi: 10.1016/S0167-739X(99)00059-X
- [5] N. Zheng, A. Paloski, and H. Wang, "An efficient user verification system via mouse movements," *ACM Transactions on Information and System Security*, vol. 16, no. 4, pp. 1–27, 2014. doi: 10.1145/2584415
- [6] R. Kumar, V. Kundu, and S. Phoha, "Continuous authentication using mouse dynamics and keystroke patterns," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 5, pp. 3344–3357, 2022. doi: 10.1109/TDSC.2021.3092282
- [7] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data*, vol. 6, no. 1, pp. 1–39, 2012. doi: 10.1145/2133360.2133363. [Scikit-learn implementation context: F. Pedregosa et al., 'Scikit-learn: Machine learning in Python,' *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2023.]
- [8] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001. doi: 10.1162/089976601750264965
- [9] E. Bursztein, M. Martin, and J. C. Mitchell, "Text-based CAPTCHA strengths and weaknesses," in *Proc. 18th ACM Conference on Computer and Communications Security (CCS)*, Chicago, IL, USA, 2011, pp. 125–138. doi: 10.1145/2046707.2046724
- [10] D. M'Raihi, S. Machani, M. Pei, and J. Rydell, "TOTP: Time-Based One-Time Password Algorithm," *IETF RFC 6238*, May 2011. [Online]. Available: <https://tools.ietf.org/html/rfc6238>
- [11] H. Saevanee, N. Clarke, and S. Furnell, "Multi-modal behavioural biometric authentication for mobile devices," in *Proc. IFIP International Information Security Conference*, Heraklion, Greece, 2012, pp. 465–474. doi: 10.1007/978-3-642-30436-1_38
- [12] R. Moskovitch et al., "Identity theft, computers and behavioral biometrics," in *Proc. IEEE International Conference on Intelligence and Security Informatics*, Dallas, TX, USA, 2009, pp. 155–160. doi: 10.1109/ISI.2009.5137288
- [13] Y. Cao, S. Li, and E. Wijmans, '(Cross-)Browser Fingerprinting via OS and Hardware Level Features,' in *Proc. Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, 2017. doi: 10.14722/ndss.2017.23152
- [14] M. Abramson and D. Aha, "User authentication from web browsing behavior," in *Proc. FLAIRS Conference*, Miami, FL, USA, 2013, pp. 227–232.