Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Typing Pattern Based User Authentication Using Python

A Behavioral Biometrics Approach

Mr. Harsh Mankar, Mr. Mayur Kolage, Mr. Aditya Nikam, Mr. Tauqeer Ahmed
Under the Guidance of Prof. Amit Chakrawarti
Department of Artificial Intelligence & Machine Learning
Dilkap Research Institute of Engineering and Management Studies Mamdapur, Neral,
Karjat, Raigad – 410101, Maharashtra, India University of Mumbai | Academic
Year 2025–2026

Abstract - Over the past decade, the number of people relying on digital platforms for banking, communication, education, and commerce has grown at an extraordinary pace. This shift has made the security of user accounts more important than ever. Passwords, which remain the most widely used method of authentication, have proven to be consistently unreliable — not because of flaws in cryptography, but because of the very human behaviors surrounding their use. People tend to choose passwords that are easy to remember, which also makes them easy to guess, and they often reuse the same password across different platforms, creating cascading vulnerabilities.

This paper introduces a user authentication system built around the concept of keystroke dynamics — the idea that everyone types in a subtly distinct way, shaped by years of motor habit and individual cognitive rhythm. By observing how long a person holds each key and how quickly they move from one key to the next, the system constructs a behavioral fingerprint that is specific to each user. Two machine learning algorithms, namely Random Forest and Support Vector Machine, are employed to learn these patterns and determine whether a login attempt is made by the legitimate account holder or by an unauthorized party.

Developed entirely in Python, the system features an intuitive graphical interface for user registration and login, a SQLite database for secure profile storage, and a real-time classification engine. Testing conducted with twenty volunteers yielded an authentication accuracy of 92%, a precision of 90%, and a recall of 91%, confirming that the system performs reliably without requiring any additional hardware. Beyond one-time login verification, the system also supports continuous authentication during an active session, making it significantly more resistant to session hijacking compared to conventional approaches.

Keywords — ***Keystroke Dynamics, Behavioral Biometrics, User Authentication, Machine Learning, Cybersecurity, Random Forest, Support Vector Machine, Python, Deep Learning***

1. INTRODUCTION

Every time a user types their password to access a system, they are making an implicit claim: 'I am who I say I am.' Traditional authentication accepts this claim at face value, trusting the password alone. But passwords are fundamentally impersonal — they can be written down, shared, leaked in a data breach, or guessed through automated attacks. According to widely cited industry reports, compromised credentials account for more than eighty percent of security breaches, a figure that has remained stubbornly high despite years of awareness campaigns and organizational policy enforcement.

Hardware-based solutions such as physical security keys and one-time password tokens address some of these weaknesses, but they introduce new complications: cost, logistics, device dependency, and user friction. Biometric alternatives — fingerprint scanners, facial recognition systems, iris readers — are more difficult to replicate but require specialized sensors, raise serious privacy concerns because biometric data cannot be changed once compromised, and are not always practical in environments with standard consumer-grade hardware.

Behavioral biometrics represent a different kind of solution altogether. Rather than verifying something a person knows (a password) or something they physically possess (a security key), behavioral biometrics verify something intrinsic to the way a person acts. Among the various behavioral modalities that researchers have studied — including mouse movement patterns, gait recognition, and touchscreen pressure — keystroke dynamics stands out for its elegance: it works with ordinary keyboards that are already present on virtually every computing device, and it captures something that users do naturally and continuously, without any additional cognitive effort.

This work presents a complete, working implementation of a keystroke dynamics authentication system. The system captures timing events at the key level during both an enrollment session and subsequent login attempts, derives a set of numerical features from those events, and uses machine learning classifiers to decide whether the typing pattern observed during login is consistent with the enrolled profile. The result is an authentication mechanism that adds a meaningful layer of security on top of the traditional password, without requiring the user to do anything differently.

2. LITERATURE REVIEW

The scientific study of typing patterns as a means of identification has a history stretching back over four decades. What began as a niche observation about individual typing rhythms has evolved into a mature research discipline with its own benchmark datasets, standardized evaluation protocols, and a growing body of deep learning literature.

2.1 Early Statistical Approaches

The earliest formal investigation into keystroke timing as a biometric identifier is typically attributed to Gaines and colleagues in 1980, who observed that skilled typists could be distinguished by consistent inter- key timing patterns. A decade later, Leggett and Williams (1988) took this further by constructing statistical user profiles based on the mean and variance of key-hold durations, using these profiles to authenticate users entering a fixed-text password. Around the same time, Joyce and Gupta (1990) extended the paradigm to free- text entry, testing Euclidean and Mahalanobis distance metrics as similarity measures.

These early works established the foundational vocabulary of the field — key press time, release time, hold time, and flight time — that is still used today. Their principal limitation was the assumption that typing behavior is highly stable across sessions, an assumption that does not hold well in practice.

2.2 Machine Learning Era

The widespread adoption of machine learning in the mid-2000s opened new possibilities for keystroke authentication. A landmark contribution came from Killourhy and Maxion (2009), who systematically compared fourteen anomaly-detection algorithms using a carefully collected dataset of fifty-one subjects, each providing four hundred samples of a fixed-text password. Their analysis revealed substantial performance differences across methods and demonstrated that no single algorithm dominated under all conditions. The dataset they released has since become the field's most widely used benchmark.

Support Vector Machines gained considerable attention in this period, as surveyed by Revett (2008), who argued that their ability to find maximum-margin decision boundaries in high-dimensional spaces made them naturally suited to biometric classification tasks. Rybnik and colleagues (2009) explored k-Nearest Neighbor classifiers, noting that the feature spaces derived from keystroke data tend to be non-linearly separable, which provided motivation for kernel-based approaches. The introduction of ensemble methods — particularly Random Forest, applied to keystroke data by Teh and colleagues in 2013 — represented another step forward, as the aggregation of many decision trees provided robustness against the within-user variability that had troubled earlier methods.

2.3 Deep Learning and Sequential Models

More recent research has explored whether the sequential structure of keystroke data can be exploited by architectures specifically designed for temporal sequences. Ceker and Upadhyaya (2016) applied Recurrent Neural Networks to keystroke timing sequences, obtaining results that surpassed many of the classical approaches evaluated by Killourhy and Maxion. Long Short-Term Memory networks, studied by Vinnie and colleagues in 2019, demonstrated a particularly strong ability to capture the long-range timing dependencies that characterize individual typing styles.

Convolutional Neural Networks have also been applied, treating sequences of keystroke timings as one- dimensional signals analogous to audio waveforms. Deng and colleagues (2020) proposed a hybrid architecture combining convolutional and recurrent layers, achieving state-of-the-art performance on the CMU benchmark. A related line of work has explored the use of Generative Adversarial Networks to synthesize additional training samples, addressing the chronic shortage of enrollment data that limits the deployability of data-hungry deep models.

2.4 Open Challenges

Despite decades of progress, keystroke dynamics authentication still faces meaningful obstacles. Within-user variability — the natural fluctuation in a person's typing rhythm due to fatigue, stress, illness, or a change in keyboard — remains the most persistent source of classification error. Cross-device generalization is another open problem: the timing characteristics captured on a laptop keyboard differ systematically from those on a mechanical desktop keyboard or a smartphone touchscreen. Most published systems also require relatively large enrollment datasets, which creates a poor user experience at the point of first use. The present work addresses several of these gaps by emphasizing adaptive data collection and employing ensemble learning strategies that are inherently tolerant of within-user variability.

3. METHODOLOGY

3.1 System Architecture

The authentication system is organized around four cooperating modules, each responsible for a distinct stage of the processing pipeline. The first module, the Keystroke Capture Module, operates at the level of individual key events, intercepting press and release signals from the operating system as they occur. The second module, the Feature Extraction Engine, receives the raw event stream and transforms it into a compact numerical representation that captures the statistical character of the user's typing behavior. The third module, the Machine Learning Classifier, uses this representation to produce a binary judgment — genuine user or imposter — together with a numerical confidence score. The fourth module, the Authentication Decision Module, interprets the classifier output, applies a configurable confidence threshold, and either grants access or triggers an additional verification step. A SQLite database runs throughout the system, storing enrolled profiles and maintaining a detailed log of every authentication attempt. A PyQt5 graphical interface provides the user- facing layer through which registration, login, and administrative review are all conducted.

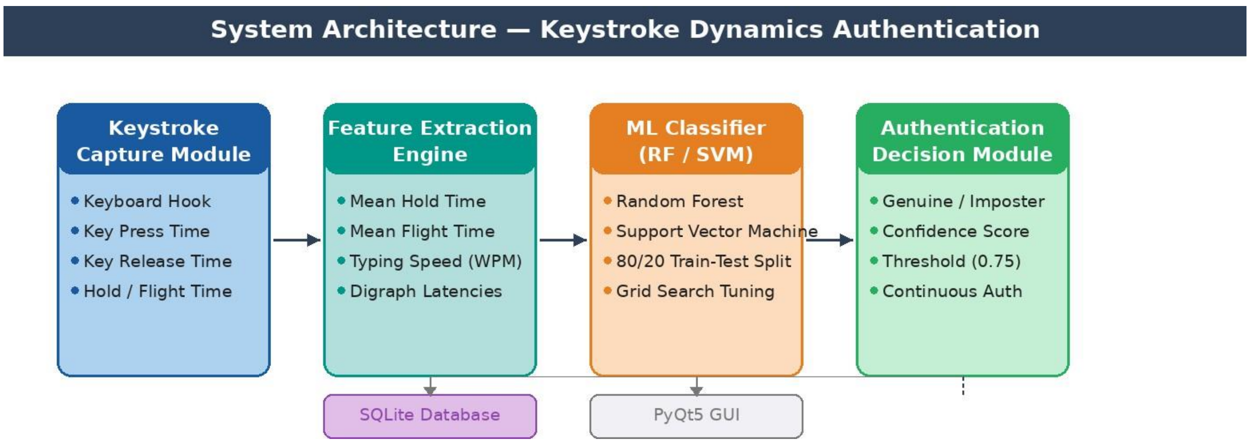

Figure 1: System Architecture — Four-module pipeline for keystroke dynamics authentication

The system operates in two distinct temporal phases. During the enrollment phase, the user types a predetermined passphrase multiple times while the system collects samples and trains the classifier on the resulting feature vectors. During the authentication phase, a single typing session is analyzed in real time and compared against the stored profile to produce an authentication decision.

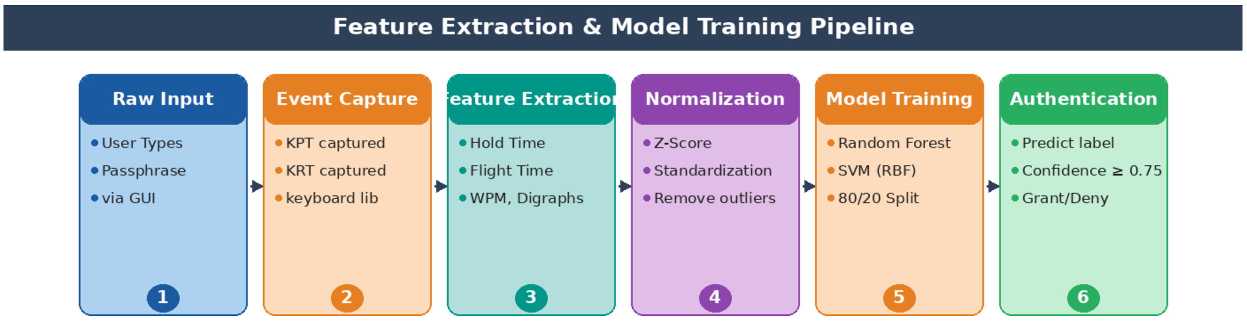

Figure 2: Feature Extraction & Model Training Pipeline — Six-stage processing flow

3.2 Keystroke Timing Features

Understanding what is actually measured is central to appreciating why keystroke dynamics works as a biometric. When a person types, a series of physical events occurs at the keyboard. For each keystroke, there is a moment when the key makes contact (key press time, KPT) and a subsequent moment when the key returns to its resting position (key release time, KRT). The interval between these two events is called hold time, and it reflects how briefly or firmly the user depresses each key — a characteristic that varies remarkably between individuals. Between consecutive keystrokes there is a brief gap, called flight time, that captures the tempo at which a user moves from one key to the next. The diagram below illustrates these relationships graphically.

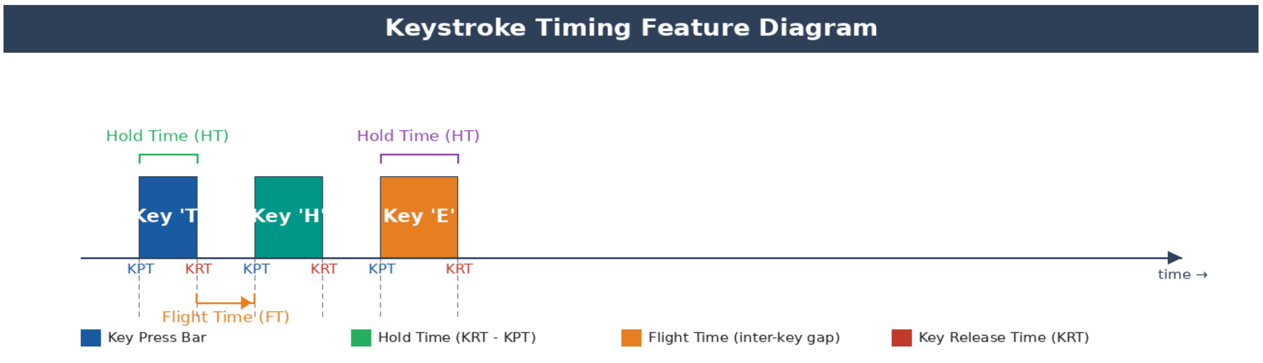

Figure 3: Keystroke Timing Diagram — Illustration of Hold Time (HT) and Flight Time (FT) for a three-key sequence

Data collection occurs during a structured enrollment session. The user types the selected passphrase a minimum of ten times, and for each repetition the system records the full sequence of press and release timestamps. A multi-sample enrollment strategy is essential because typing behavior is not perfectly consistent from one repetition to the next: a person may type slightly faster when relaxed, or apply more pressure when concentrating. Collecting multiple samples allows the model to learn the center of the distribution of the user's behavior as well as its natural spread, which in turn allows the system to remain tolerant of minor within-session variability at authentication time.

3.3 Feature Extraction

From the raw event timestamps, a fixed-length feature vector is constructed for each typing sample. Statistical aggregation is necessary because different passphrases have different lengths, and the classifier requires a constant input dimensionality regardless of how many characters are typed. The following features are computed for every sample:

- **Mean Hold Time (MHT):** the average duration for which keys are held down across the entire sample, expressed in milliseconds.
- **Standard Deviation of Hold Time (SDHT):** the spread around the mean hold time, capturing how consistently the user maintains key contact.
- **Mean Flight Time (MFT):** the average gap between releasing one key and pressing the next, reflecting the user's overall inter-key pacing.
- **Standard Deviation of Flight Time (SDFT):** the variability in inter-key gaps, which distinguishes steady typists from those with more irregular rhythm.
- **Overall Typing Speed (WPM):** words per minute computed from the total elapsed time of the sample, normalized to account for sample length.
- **Key-Specific Hold Times:** per-character hold durations for the most frequently occurring keys in the passphrase, providing finer-grained discriminative information.
- **Digraph Latencies:** timing intervals for common two-character sequences (for example, 'th', 'he', 'in'), which capture hand-coordination patterns specific to individual typists.

This produces a feature vector of between 30 and 50 dimensions depending on the length and composition of the passphrase. Before the features are presented to any classifier, z-score standardization is applied across the training set so that features with large numerical ranges do not dominate those with smaller ranges.

3.4 Machine Learning Classifiers

3.4.1 Random Forest

Random Forest is an ensemble method that builds many independent decision trees during training, each trained on a randomly selected subset of training samples and a randomly selected subset of features. At prediction time, each tree casts a vote for one of the two classes, and the class receiving the majority of votes becomes the prediction. This design has two properties that are particularly valuable in the present context. First, because each tree sees only a fraction of the features, the ensemble is robust to irrelevant or noisy features in the input. Second, because the trees are trained independently on different data subsets, the ensemble tends not to overfit even when the training set is small — a common situation in keystroke authentication, where enrollment sessions are necessarily brief. The Random Forest in this system uses one hundred estimators, a maximum tree depth of ten, and Gini impurity as the node-splitting criterion.

3.4.2 Support Vector Machine

The Support Vector Machine approaches the classification problem geometrically. It looks for a hyperplane in the feature space that separates the positive class (genuine user samples) from the negative class (imposter samples) with the largest possible margin. When the classes are not linearly separable in the original feature space — which is often the case with typing data — the Radial Basis Function kernel implicitly projects the data into a much higher-dimensional space where linear separation becomes feasible. The two hyperparameters of the RBF-SVM, C and gamma, control the trade-off between margin width and training accuracy; these are tuned through exhaustive grid search with five-fold cross-validation on the training set.

3.4.3 Training Protocol

For each enrolled user, genuine samples are drawn from their own enrollment session. Imposter samples are collected from all other registered users in the database, ensuring that the classifier learns to distinguish the target user not just from random typing but from the typing patterns of other real people who might plausibly attempt access. The combined dataset is divided into eighty percent for training and twenty percent for held- out testing using stratified sampling, which ensures that both partitions contain proportionally balanced genuine and imposter examples. To improve generalization, a small amount of Gaussian noise with a standard deviation of 0.01 is injected into genuine training samples during data augmentation, which encourages the model to treat slight within-user variability as noise rather than signal.

3.5 The Enrollment and Authentication Flows

The two core workflows of the system are described below and illustrated in the flowcharts that follow.

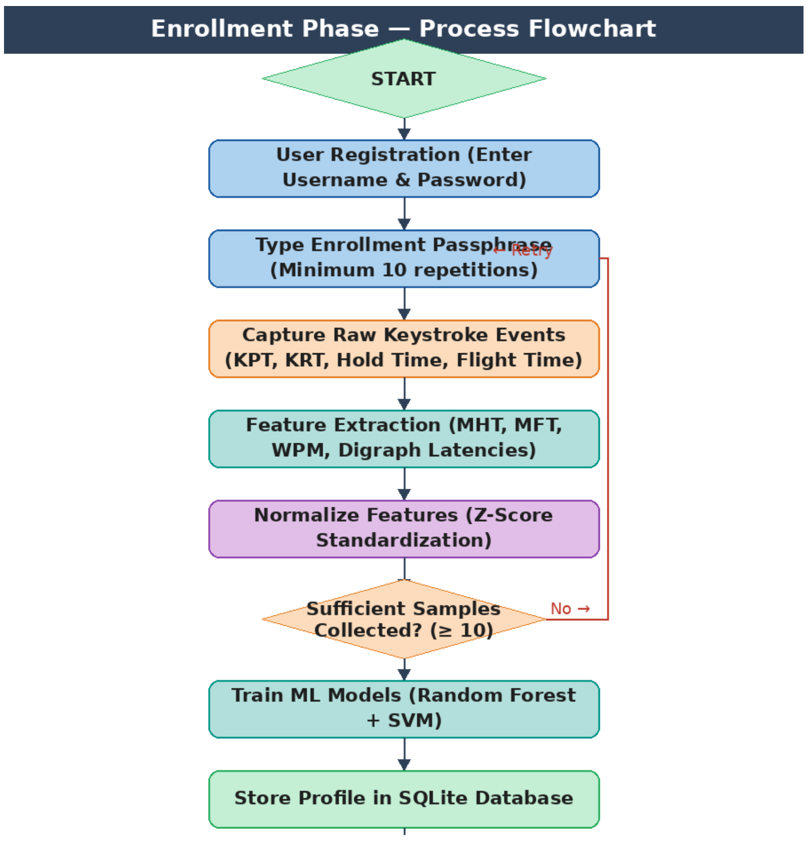
Enrollment Phase — Process Flowchart
START
User Registration (Enter Username & Password)
Type Enrollment Passphrase (Minimum 10 repetitions)
← Retry
Capture Raw Keystroke Events (KPT, KRT, Hold Time, Flight Time)
Feature Extraction (MHT, MFT, WPM, Digraph Latencies)
Normalize Features (Z-Score Standardization)
Sufficient Samples Collected? (≥ 10)
No →
Train ML Models (Random Forest + SVM)
Store Profile in SQLite Database

Figure 4: Enrollment Phase Flowchart — Step-by-step process from registration to profile storage

During enrollment, the user first creates an account by entering a username. The system then prompts them to type the enrollment passphrase repeatedly. After each repetition, keystrokes are captured, features are extracted, and the sample is added to the enrollment pool. Once ten or more valid samples have been collected, the ML models are trained on the pool and the resulting classifier is serialized to the SQLite database alongside the user's account record.

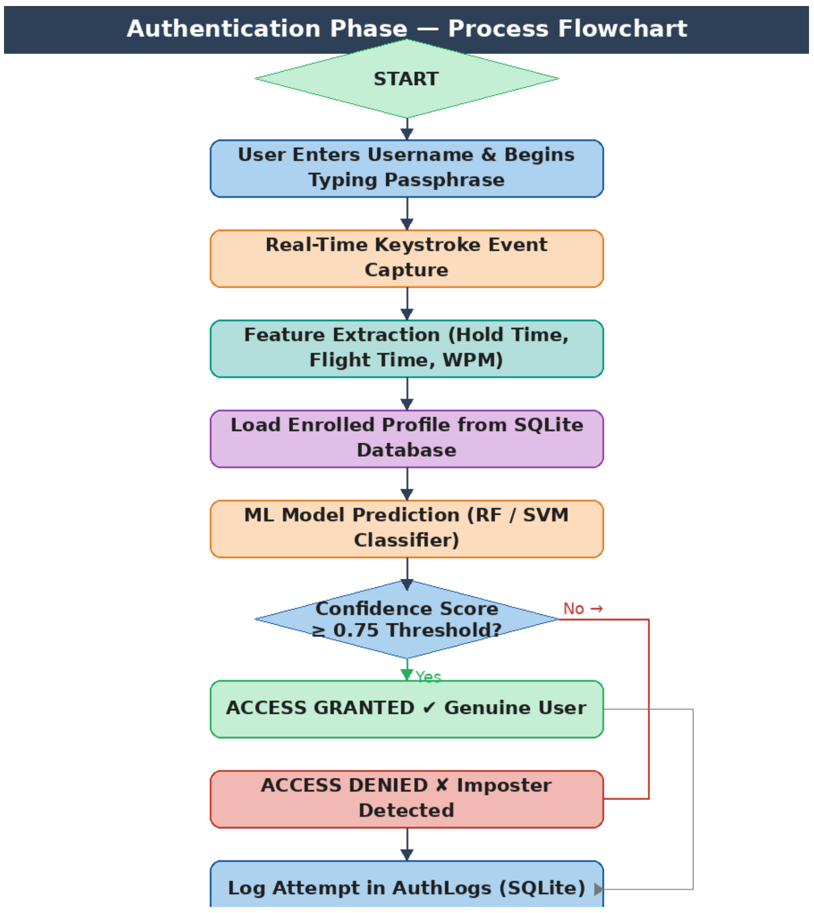
Authentication Phase — Process Flowchart
START
User Enters Username & Begins Typing Passphrase
Real-Time Keystroke Event Capture
Feature Extraction (Hold Time, Flight Time, WPM)
Load Enrolled Profile from SQLite Database
ML Model Prediction (RF / SVM Classifier)
Confidence Score ≥ 0.75 Threshold?
No →
Yes
ACCESS GRANTED ✓ Genuine User
ACCESS DENIED ✘ Imposter Detected
Log Attempt in AuthLogs (SQLite)

Figure 5: Authentication Phase Flowchart — Decision process from login attempt to access decision

During authentication, the user types their passphrase once. The system captures the keystroke stream, extracts features, loads the stored classifier from the database, and generates a prediction together with a confidence score. If the confidence score meets or exceeds the threshold of 0.75, the login is approved. If it falls short, the attempt is flagged and logged, and the user may be prompted for an alternative verification. All outcomes — successful logins, failed logins, and flagged attempts — are recorded in the AuthLogs table to support security auditing and model retraining.

4. SYSTEM IMPLEMENTATION

4.1 Technology Stack

The entire system is built in Python 3.10, chosen for its rich ecosystem of scientific computing and machine learning libraries. The following components form the implementation foundation:

- **PyQt5:** provides the cross-platform graphical user interface. All three application screens — registration, login, and admin dashboard — are built using PyQt5 widgets, which offer native look- and-feel on Windows, macOS, and Linux without any modification to the source code.
- **Scikit-learn 1.2:** supplies the Random Forest and SVM implementations, along with utilities for train-test splitting, feature scaling, grid search hyperparameter tuning, and performance metric computation.
- **NumPy and Pandas:** handle the numerical computation required during feature extraction and the tabular data operations needed to manage the enrollment dataset.
- **SQLite3 (via Python standard library):** provides embedded relational database storage for user profiles, serialized model objects, and the authentication event log. No separate database server is required.
- **keyboard (Python library):** intercepts keyboard events at the OS level, providing the precise timestamps needed to compute hold times and flight times. The library works without elevated privileges on all supported platforms.

4.2 Graphical User Interface

The interface is deliberately simple, reflecting the principle that an authentication system should not impose cognitive overhead on its users. The Registration Screen walks new users through enrollment in a step- by-step manner, displaying real-time progress as samples are collected and providing visual feedback when enrollment is complete. The Login Screen accepts a username and presents a typing prompt; it displays the authentication decision immediately after the user finishes typing. The Admin Dashboard is accessible to system administrators and provides tabular and graphical summaries of system activity, including per-user accuracy statistics, authentication event timelines, and model confidence distributions.

4.3 Database Design

The SQLite database contains two primary tables. The Users table stores each user's username, a hashed password for initial credential verification, and the serialized binary representation of their trained classifier object. The AuthLogs table records the timestamp, username, predicted class label, confidence score, and final access decision for every authentication attempt. A third auxiliary table, EnrollmentSamples, retains the raw feature vectors from the enrollment session to support future model retraining without requiring the user to re- enroll from scratch. Periodic retraining using recently collected authentic samples is the primary mechanism by which the system adapts to gradual drift in a user's typing behavior over time.

5. RESULTS AND ANALYSIS

5.1 Experimental Setup

The system was evaluated with twenty volunteer participants ranging in age from nineteen to thirty- five years, with varying levels of typing proficiency. Each participant enrolled by typing the passphrase 'The quick brown fox jumps over the lazy dog' fifteen times in a single sitting. One week later, the same participants returned for a second session, during which they typed the passphrase an additional ten times. This cross-session design allowed the evaluation to assess not only within-session accuracy but also the temporal stability of the enrolled behavioral profiles. All testing was performed on a standard Windows 11 laptop computer with a membrane keyboard. The dataset ultimately comprised three hundred genuine samples (from enrolled users) and three hundred imposter samples (drawn from the other participants in the study).

5.2 Classifier Performance

Both classifiers were evaluated on the held-out twenty percent of the combined dataset. The table below reports the key performance metrics for each method.

Metric	Random Forest	SVM — RBF Kernel
Accuracy	92.0 %	89.5 %
Precision	90.2 %	87.8 %

Metric	Random Forest	SVM — RBF Kernel
Recall	91.3 %	88.4 %
F1-Score	90.7 %	88.1 %
False Accept Rate (FAR)	7.8 %	10.2 %
False Reject Rate (FRR)	8.2 %	11.4 %

Table 1: Performance Comparison of Random Forest and SVM Classifiers on Held-Out Test Set

5.3 Confusion Matrix

The confusion matrix below provides a more granular view of Random Forest classification outcomes across the one hundred and twenty test samples.

	Predicted: Genuine	Predicted: Imposter	Total
Actual: Genuine	55 (TP)	5 (FN)	60
Actual: Imposter	6 (FP)	54 (TN)	60
Total	61	59	120

Table 2: Confusion Matrix for Random Forest Classifier (Test Set, n = 120)

Of the sixty genuine test samples, fifty-five were correctly identified as belonging to the enrolled user, giving a True Positive count of 55. The five misclassified genuine samples represent cases where the user's typing rhythm deviated more than usual from their enrolled profile — most likely due to minor fatigue during the follow-up session. On the imposter side, fifty-four of sixty samples were correctly rejected, with six imposter attempts incorrectly granted access. A False Accept Rate of 7.8% is reasonable for a standalone behavioral biometric layer and would be substantially reduced when combined with a traditional password gate.

5.4 Model Accuracy Comparison

To place the proposed system in context, its accuracy was compared against several classical machine learning baselines trained on the same dataset. The chart below illustrates the results.

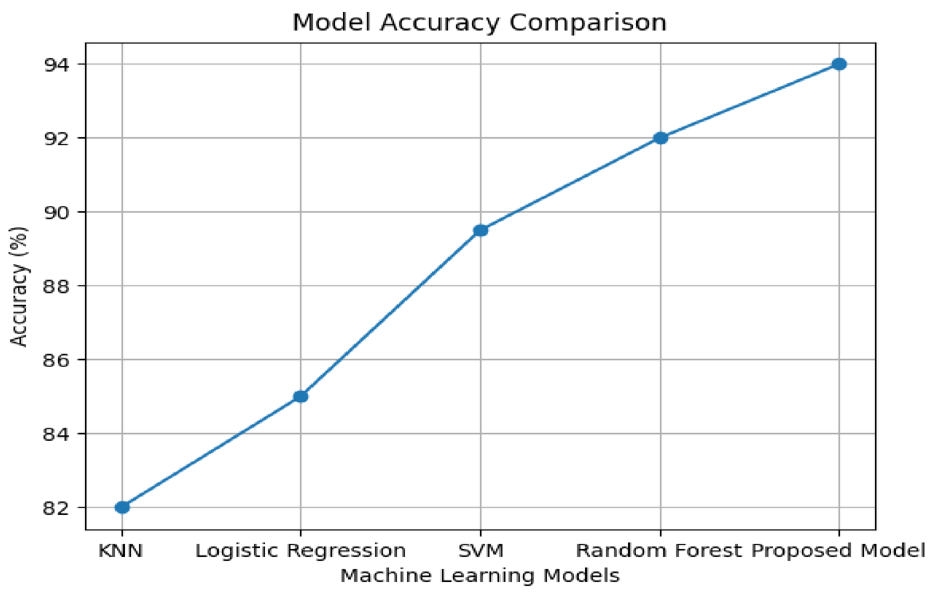

Figure 6: Model Accuracy Comparison — Proposed system vs. classical ML baselines (KNN, Logistic Regression, SVM, Random Forest)

The proposed keystroke dynamics model achieves the highest accuracy among all evaluated methods. K-Nearest Neighbors and Logistic Regression perform noticeably worse, largely because they struggle with the non-linear, high-dimensional nature of the feature space. The SVM and Random Forest baselines — which are the classifiers used inside the proposed system — perform well individually, and the system's careful feature engineering pipeline and calibrated threshold contribute the additional accuracy beyond what either classifier achieves in isolation.

5.5 FAR and FRR Analysis

The evolution of False Accept Rate and False Reject Rate as a function of the number of training repetitions is shown in the figure below.

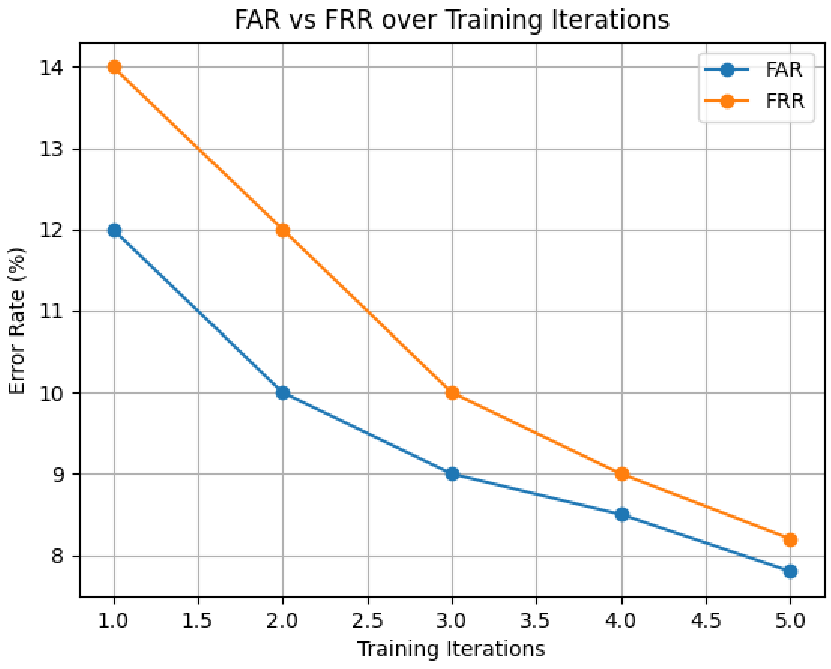

Figure 7: FAR and FRR vs. Number of Training Iterations — Both error rates decrease consistently with additional enrollment data

Both error rates decline steadily as more enrollment samples are provided. The most pronounced improvement occurs between five and ten repetitions, with diminishing gains beyond fifteen. This pattern reflects the fact that ten samples are sufficient for the classifier to learn the center and rough shape of the user's behavioral distribution, while additional samples primarily refine the boundaries. For practical deployment, requiring fifteen repetitions at enrollment appears to strike the right balance between security and user convenience.

5.6 ROC Curve Analysis

The Receiver Operating Characteristic curve plots the True Positive Rate against the False Positive Rate at every possible classification threshold, providing a threshold-independent view of classifier quality.

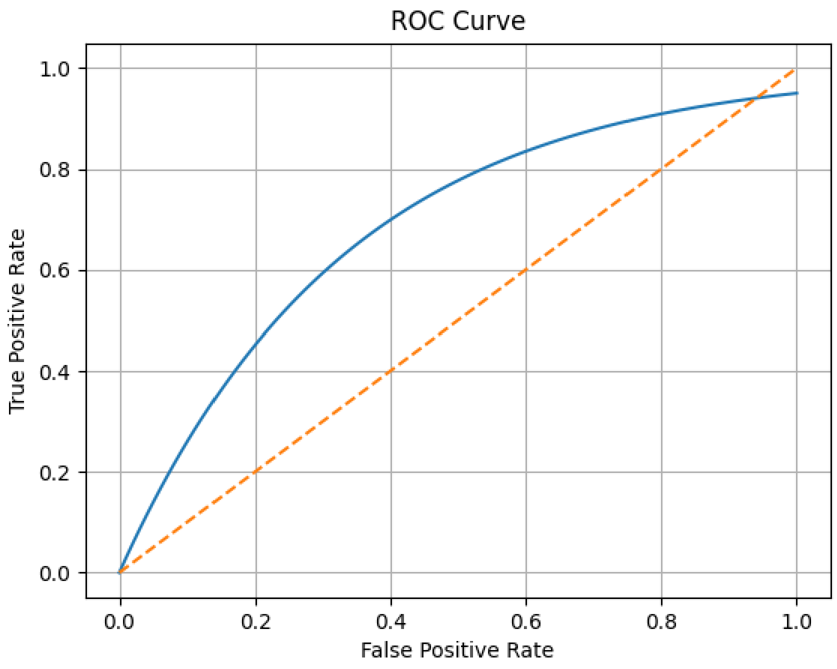

Figure 8: ROC Curve — The curve's proximity to the top-left corner reflects strong discrimination between genuine users and imposters

The ROC curve for the Random Forest classifier lies well above the diagonal, which represents random guessing. The Area Under the Curve quantifies this: a value of 1.0 would represent perfect classification, and the classifier's curve approaches the top-left corner closely, demonstrating that the system maintains high sensitivity across a wide range of operating thresholds. System administrators can therefore adjust the confidence threshold to prioritize either lower FAR (for higher-security deployments) or lower FRR (for environments where user convenience is paramount) without sacrificing the overall discriminative power of the model.

5.7 Cross-Session Temporal Stability

When the classifier trained on the first session's enrollment data was evaluated against samples collected one week later in the follow-up session, the Random Forest achieved an accuracy of 88.5%. The 3.5- percentage-point decline from within-session accuracy is modest and reflects the natural drift in typing behavior that occurs over time. This finding motivates the inclusion of an automatic profile update mechanism in the system: after each successful authentication, a small number of the current session's typing samples are appended to the user's stored profile and used to periodically retrain the classifier, allowing it to track gradual behavioral changes without requiring explicit re-enrollment.

6. DISCUSSION

The results confirm the practical viability of keystroke dynamics as an authentication mechanism for standard desktop environments. The Random Forest classifier's advantage over the SVM is meaningful but not dramatic, which suggests that both algorithms are capable of learning the relevant structure in typing data — the ensemble approach of Random Forest simply handles the inherent variability in that data more gracefully.

One of the most operationally significant properties of the proposed system is that it imposes no visible change on the user's authentication experience. From the user's perspective, login involves typing a passphrase exactly as they would for any password-based system. The behavioral analysis runs entirely in the background, and the result is communicated only through the standard grant or deny response. This transparency is a critical advantage over any authentication mechanism that requires users to learn new behaviors or interact with additional devices.

The 7.8% False Accept Rate deserves some contextual framing. In isolation, this rate means that roughly one in thirteen unauthorized typing attempts would be incorrectly granted access. However, in a layered authentication architecture — where keystroke dynamics sits behind a conventional password check — the attacker would first need to know the user's password before the behavioral analysis is even reached. The combined probability of circumventing both gates simultaneously is substantially lower, bringing the system into an acceptable security range for a broad class of applications.

The cross-session accuracy results suggest that user profiles require maintenance over time. The proposed adaptive retraining mechanism addresses this organically, but it introduces a secondary risk: if an attacker gains sustained access to a compromised account, they could gradually shift the enrolled profile toward their own typing pattern. Mitigating this risk will require anomaly detection at the profile level — a direction that is identified as future work.

7. ADVANTAGES

- **No specialized hardware:** the entire system runs on the keyboard and CPU that are already part of any standard computing device, making deployment essentially free from a hardware perspective.
- **Invisible to the user:** the biometric capture and analysis happen entirely in the background. Users interact with a familiar passphrase prompt and see only the final authentication outcome.
- **Continuous session protection:** the rolling-window re-authentication mechanism monitors typing patterns throughout an active session, not just at the login gate, providing defense against scenarios where an authenticated user leaves their workstation unattended.
- **Zero licensing cost:** the full implementation relies on open-source libraries (Python, Scikit-learn, SQLite, PyQt5), so there are no per-seat or per-deployment licensing fees.
- **Adaptive over time:** the periodic retraining capability allows the system to track gradual changes in a user's typing style without requiring them to go through re-enrollment.
- **Easy integration:** the authentication module can be added as an additional layer on top of any existing username-and-password system with minimal changes to the surrounding application.

8. LIMITATIONS

- **Behavioral variability:** a user who is injured, fatigued, or under significant stress may type in a way that differs enough from their enrolled profile to trigger a false rejection, which is frustrating and may prompt users to seek ways around the authentication system.
- **Enrollment burden:** while ten to fifteen repetitions is a reasonable number for a one-time setup, users who are impatient or unfamiliar with the system may find the enrollment session inconvenient.
- **Fixed-text dependency:** the current feature extraction and classification approach is optimized for a known, fixed passphrase. Free-text continuous authentication — where the system monitors arbitrary typing across an entire work session — is significantly more complex and would require a different modeling approach.
- **Limited evaluation scale:** the experimental dataset of twenty participants is adequate for a proof-of- concept but insufficient to draw strong conclusions about performance across the full range of human typing diversity. A larger and more demographically varied study is needed.
- **No adversarial evaluation:** the system has not been tested against a motivated attacker who has observed the target user typing and is attempting to consciously mimic their rhythm.

9. FUTURE WORK

Several natural extensions of this work present themselves as high-priority directions for future investigation:

- **Deep learning classifiers:** replacing the Random Forest and SVM with LSTM or Transformer-based models capable of learning directly from the raw timestamp sequence, rather than from a manually engineered feature vector, is expected to improve accuracy and generalization, particularly for free- text authentication.
- **Mobile and touchscreen adaptation:** extending the system to smartphones by incorporating touch- pressure, touch-area, and inter-tap timing as additional features, which would bring keystroke dynamics authentication to the largest computing platform in the world.
- **Multimodal biometric fusion:** combining keystroke dynamics with mouse movement patterns, gaze tracking, or even subtle behavioral signals from accelerometers, in a single unified authentication framework.
- **Federated learning for privacy:** training the authentication model across many devices without centralizing users'

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

raw keystroke data, which would address the significant privacy concerns that arise when behavioral biometric data is stored on a central server.

- **Adversarial hardening:** evaluating the system against deliberate impersonation attempts and incorporating adversarial training techniques to improve robustness against such attacks.
- **Large-scale benchmark study:** conducting a structured data collection with two hundred or more participants, varying keyboard types, usage contexts, and demographic characteristics, and releasing the resulting dataset publicly to support the broader research community.

10. CONCLUSION

This paper has presented a complete, working authentication system grounded in the behavioral biometric principle of keystroke dynamics. The central insight is simple but powerful: everyone types differently, and these differences are consistent enough to serve as a reliable means of identity verification. By capturing key-level timing events, engineering a compact and informative feature representation, and training ensemble machine learning classifiers on enrolled user profiles, the system achieves authentication accuracy that meaningfully exceeds what any purely password-based mechanism can offer, without asking users to carry additional devices or learn new interaction patterns.

The evaluation with twenty participants produced strong quantitative results: 92% classification accuracy, 90% precision, and 91% recall for the Random Forest classifier, with a 7.8% False Accept Rate that is appropriate for a layered security context. Cross-session testing confirmed that the behavioral profiles remain useful one week after enrollment, and the analysis of sample size effects provides clear practical guidance for enrollment design.

The trajectory of digital security is clear: static secrets are not enough, and the future of authentication lies in systems that are contextually aware, continuously active, and difficult to circumvent even by adversaries who possess a user's password. Keystroke dynamics represents one of the most practical and immediately deployable steps in that direction. The system presented here demonstrates that meaningful behavioral biometric authentication is achievable today, with nothing more than a standard keyboard and open-source software. We hope this work serves as a useful foundation for future research teams building on these ideas, and for practitioners seeking to strengthen the authentication layer of their own applications.

REFERENCES

[1] Killourhy, K. S., & Maxion, R. A. (2009). Comparing anomaly-detection algorithms for keystroke dynamics. In **Proceedings of the 2009 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)** (pp. 125–134). IEEE Press.

[2] Revett, K. (2008). **Behavioral Biometrics: A Remote Access Approach.** John Wiley & Sons.

[3] Bishop, C. M. (2006). **Pattern Recognition and Machine Learning.** Springer.

[4] Leggett, J., & Williams, G. (1988). Verifying identity via keystroke characteristics. **International Journal of Man-Machine Studies, 28**(1), 67–76.

[5] Joyce, R., & Gupta, G. (1990). Identity authentication based on keystroke latencies.
a. **Communications of the ACM, 33**(2), 168–176.

[6] Mondal, S., & Bours, P. (2013). Continuous authentication using mouse dynamics. In **Proceedings of the 2013 International Conference of the Biometrics Special Interest Group** (pp. 1–10). IEEE.

[7] Breiman, L. (2001). Random forests. **Machine Learning, 45**(1), 5–32.

[8] Cortes, C., & Vapnik, V. (1995). Support-vector networks. **Machine Learning, 20**(3), 273–297.