PPMLAuth: Privacy-Preserving and Tamper-Resistant Behavioral Authentication

David Monschein[0000−0003−4303−0712], Alexander Niedermayer[0009−0009−2702−5797], and Oliver P. Waldhorst[0000−0002−5468−2073]

Institute of Data-Centric Software Systems (IDSS)
Karlsruhe University of Applied Sciences, Karlsruhe, Germany
{david.monschein,alexander.niedermayer,oliver.waldhorst}@h-ka.de

Abstract. Behavioral authentication offers a convenient solution for client–server applications. To address the inherent privacy concerns associated with sending behavioral data to the server, current work proposes analysis based on homomorphic encryption (HE). However, existing approaches remain limited: they only support low-complexity analysis and rely on assumptions that rarely hold in real-world settings. We present PPMLAuth, which addresses these shortcomings by integrating machine learning (ML) with HE-based encrypted analysis in a secure architecture. Client devices encrypt behavioral data before transmission, and the server performs ML-based inference on the encrypted data, enabling the fusion of multiple data sources without exposing sensitive information. To securely reveal the encrypted authentication decision to the server, we employ a zero-knowledge proof. Additionally, a key rollover strategy addresses cryptographic vulnerabilities. In a detailed evaluation, we show that PPMLAuth provides strong security and privacy guarantees, matches or exceeds the performance of existing methods, and ensures practical authentication latency and network overhead.

Keywords: Homomorphic Encryption · Machine Learning · Zero-Knowledge Proofs · Neural Networks · Multimodal Authentication · Key Rotation

1 Introduction

Mobile online services, such as social media platforms and e-commerce applications, are part of most people's daily lives. Typically, such services are structured as client–server applications, where the server must ensure robust authentication of the clients. In this context, behavioral authentication [40] has emerged as an alternative to traditional multi-factor authentication [1]. It relies on analyzing information about the user's behavior, which can be reflected in various data sources available on modern smartphones (e.g., accelerometer and touchscreen interactions) [1]. The main advantage is that users can be authenticated continuously without disrupting their interaction with the service [1].

To enable behavioral authentication in remote settings, the server needs to collect and analyze behavioral information of the client. However, this raises serious privacy concerns, as such data can reveal sensitive details. For instance,

motion data from accelerometers and gyroscopes can be used to infer physical activities, gait patterns, or even individual identity [23]. In response to this, current approaches adopt homomorphic encryption (HE), which enables computations on encrypted data [23,46]. More specifically, the client encrypts behavioral data using HE before sending it to the server. The server then analyzes the data in encrypted form without accessing the underlying information.

However, HE-based approaches for behavioral authentication face several problems. First, the high computational effort of HE limits the complexity of the behavioral analysis. As a result, the **fusion of multiple behavioral data sources**, which can enhance authentication performance, becomes challenging, as it typically relies on machine learning (ML) techniques [1]. Another problem is that the analysis result remains encrypted and a **secure procedure is required for disclosing** it to the server. In particular, the server needs to learn the result without risking manipulation or leaking behavioral information. Finally, **recent cryptographic vulnerabilities** of HE schemes endanger security and privacy, as attackers may reconstruct secret keys [8,11].

Existing approaches that address these problems can be divided into three categories. The first category uses HE schemes that limit the behavioral analysis to distance metrics [5,6,17,18,36,46]. The second category applies modern HE schemes to enable ML-based analysis on the server-side [30,31,41]. The third category of approaches relies on methods other than HE [9,15,22,23,37,38]. Across these categories, existing solutions either restrict analysis expressiveness, rely on strong trust assumptions, lack joint analysis of multiple behavioral data sources, or employ vulnerable cryptographic constructions.

To tackle the shortcomings of existing work, we present the *PPMLAuth (Privacy-Preserving and ML-based Authentication)* framework. Our approach combines two state-of-the-art HE schemes [12,13] to perform an ML-based analysis of encrypted behavioral information that effectively fuses multiple data sources. Further, we employ an innovative disclosure process in which the client informs the server about the decrypted authentication result along with a zero-knowledge proof that demonstrates correct decryption. Overall, PPMLAuth ensures strong security and privacy guarantees under realistic assumptions, supporting broad practical applicability. Ultimately, PPMLAuth incorporates a key rollover strategy to mitigate vulnerabilities of HE schemes [8,11].

In summary, the key contributions of our approach are:

- Realizing an ML-based, server-side analysis of encrypted behavioral information that supports the recognition of complex patterns across data sources to improve authentication effectiveness and robustness against adversaries.
- Establishing a secure and privacy-preserving architecture that fits real-world constraints. To this end, we develop an efficient ZKP-based mechanism that allows the server to obtain the decrypted analysis result while offering strong privacy guarantees and preventing result manipulation.
- Introducing a seamless key rollover strategy that integrates directly into the authentication workflow, mitigating cryptographic vulnerabilities while eliminating disruptions associated with key renewal.

To evaluate PPMLAuth, we combine a formal analysis with an evaluation based on two real-world datasets [39,33]. We show that the proposed approach provides strong security and privacy guarantees, outperforms prior work in relevant scenarios, and matches or exceeds the performance of approaches without privacy protection. In particular, the results demonstrate that our framework improves authentication effectiveness by fusing multiple behavioral data sources, while maintaining reasonable latency, network traffic, key rollover overhead, and scalability characteristics.

The paper is organized as follows. Section 2 covers underlying concepts. In Section 3, we describe our framework and delve into its building blocks. Section 4 includes an analysis of the security and privacy characteristics. The evaluation setup and results are presented in Section 5. Lastly, Section 6 summarizes related work and Section 7 discusses key findings.

2 Foundations

This section outlines the key concepts underlying our framework.

2.1 Homomorphic Encryption

Homomorphic encryption (HE) allows performing computations on encrypted data without the need for decryption [28], but introduces significant computational overhead. There are several HE schemes that vary in supported operations and efficiency. For example, the Cheon-Kim-Kim-Song (CKKS) scheme [12] supports real number arithmetic, while Fast Fully Homomorphic Encryption over the Torus (TFHE) and related approaches enable boolean circuit evaluation [13]. CKKS and TFHE are currently the most practical schemes, as they enable efficient execution of complex computations, such as neural network inference [28].

CKKS uses a public key for encryption and a secret key for decryption, while relying on additional public key material to enable operations on encrypted data [12]. Encrypted data is organized into ciphertexts with a certain amount of slots, where each slot holds a real number. Further, CKKS performs approximate arithmetic, meaning that ciphertexts represent values with a small error that grows with certain operations [12]. In terms of HE operations, multiplications and additions are supported. Some operations that are not supported by CKKS (e.g., comparisons) can be realized through scheme switching, which converts CKKS ciphertexts to a TFHE-style evaluation regime (hereafter also referred to as TFHE for simplicity) [3]. Scheme switching requires the generation of a transformation key, which is derived from the CKKS secret key. During the conversion, CKKS slots are mapped into the boolean domain [3].

Based on the CKKS scheme, there are approaches that facilitate ML model inferences on encrypted data [2]. Training on encrypted data is only possible to a very limited extent according to the current state of research [28]. Despite numerous advantages, ML model inference with HE has several limitations. First, the computational effort of HE can increase inference duration by orders of

magnitude [28]. In addition, common activation functions of neural networks (e.g., ReLU [32]) cannot be calculated efficiently with HE and have to be replaced by polynomial approximations such as $f(x) = ax^2 + bx$ [32].

Recent research uncovered vulnerabilities affecting HE schemes like CKKS and TFHE [8,11]. Of relevance are key recovery attacks, where an attacker with knowledge of ciphertexts and their corresponding plaintexts can derive details about the secret key. In the context of behavioral authentication, such a situation arises when the server learns the decrypted outcome of an authentication. However, attacks typically require access to a large number of ciphertexts and the knowledge of their exact decrypted values. In practice, periodic renewal of cryptographic key material provides an effective and sustainable defense [8].

2.2 Zero-Knowledge Proofs

Zero-knowledge proofs (ZKPs) allow a prover to demonstrate the validity of a statement without disclosing additional information [34]. Modern ZKP systems employ non-interactive constructions, where the prover produces a standalone proof that the verifier can check without further communication. To enable such proofs, the underlying computation is represented as a *circuit* that specifies all constraints a valid input must satisfy [34]. The prover produces a proof showing that their secret inputs satisfy the circuit's constraints, and the verifier can confirm this from the proof alone, without seeing private inputs.

Among current proof systems, zk-SNARKs are widely used in practice, particularly in applications requiring private data processing [34]. Most zk-SNARK constructions achieve small proof sizes and fast verification times, but rely on a trusted setup that produces a *proving key* and a *verification key*.

Groth16 is a popular zk-SNARK construction that supports efficient proof generation and verification with constant proof size [20]. It requires a circuit-specific trusted setup, where a structured reference string (SRS) must be generated [20]. If the secret randomness used in this setup is not securely removed, an adversary with the SRS could forge proofs for false statements.

3 Authentication Framework

In this section, we describe *PPMLAuth (Privacy-Preserving and ML-based Authentication)*, which enables behavioral authentication for client–server applications with mobile clients. We first provide an overview of our framework and then explain the core components that make up our contributions. To keep the setting clear, we equate the client with the actual user below.

3.1 Framework Overview

Our framework operates in the following three phases.

Initialization: At first start of the application, each client initializes cryptographic material by generating a key pair for the CKKS HE scheme consisting of

a public key pk_c, and a secret key sk_c. Next, the client derives a transformation key tk, which enables the transition from CKKS to TFHE. In this process, the client also obtains the public key pk_t and the secret key sk_t for TFHE evaluation. For simplicity, we always refer to a single public key that represents the entire public key material, since HE typically uses multiple key components. The transformation key tk, as well as the public keys pk_c and pk_t are transferred to the server. Furthermore, a hash commitment com_{sk} of the secret key sk_t is sent to the server to support the ZKP described later in Section 3.4.

Enrollment phase: Initially, no user-specific behavioral information is available for ML-based analysis. During enrollment, the server builds a behavioral baseline consisting of reference samples, while relying on conventional authentication factors (e.g., multi-factor authentication) [40]. In case an authentication is successful, the server stores the encrypted behavioral data received (*behavioral encoding*). The enrollment phase ends once sufficient reference encodings have been collected to enable behavioral analysis (see Section 3.3).

Authentication phase: Figure 1 outlines the main tasks of PPMLAuth's behavioral authentication process, which is triggered according to an application-defined policy (e.g., upon relevant context changes or periodically during interaction). In the first step, the client preprocesses behavioral signals using a pipeline

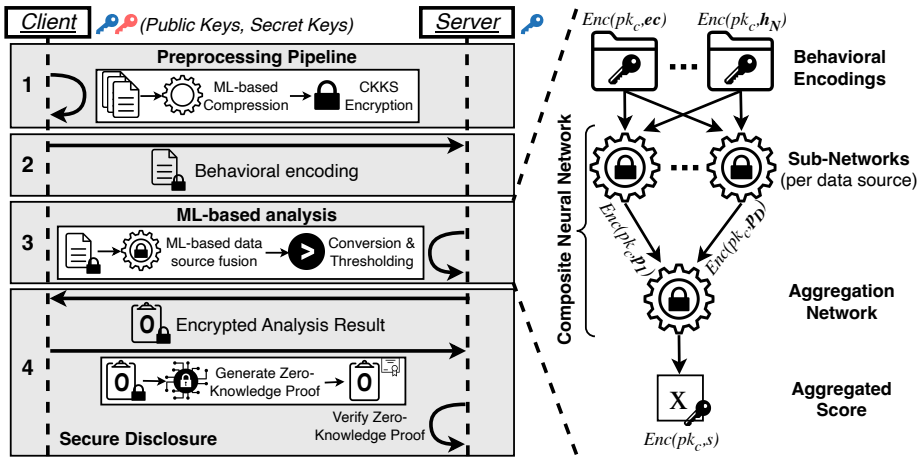

Fig. 1: Overview of the PPMLAuth behavioral authentication process

(1) that includes an ML-based compression stage employing transformer neural networks [44]. The pipeline output is encrypted with the CKKS HE scheme to produce a *behavioral encoding*. Details about the pipeline are introduced in Section 3.2. The client then transmits the behavioral encoding to the server (2). Along with the user's reference encodings collected during the enrollment phase, the current behavioral encoding is fed into a composite neural network. It merges data source-specific characteristics and uses scheme switching to de-

rive a TFHE ciphertext that encodes a binary authentication decision (3). The ML-based analysis and the scheme conversion are described in Section 3.3, while Section 3.6 introduces the strategy for training the neural networks.

To disclose the encrypted authentication decision, the server sends the corresponding ciphertext to the client, which decrypts it and generates a ZKP that attests an honest decryption. Subsequently, the client returns the decrypted result together with the corresponding ZKP to the server. In this way, PPMLAuth prevents a malicious client from manipulating the authentication outcome. This *secure disclosure* mechanism is explained in Section 3.4. To mitigate the risk posed by vulnerabilities in the HE schemes used, our framework includes an efficient strategy for periodic *key rollovers*. These are prepared during regular authentication, and ensure that key transitions occur without substantial interruptions. Details on key rollovers are provided in Section 3.5.

In the upcoming sections, we denote variables by a, parameters by A, vectors by $\boldsymbol{a}$ and matrices by $\boldsymbol{A}$. Encryption and decryption are written as $\mathsf{Enc}(pk_c, \cdot)$ and $\mathsf{Dec}(sk, \cdot)$.

3.2 ML-based Preprocessing Pipeline

Once authentication is required, the client uses monitored behavioral data as input for a preprocessing pipeline. The input data is arranged as a sequence of matrices $\boldsymbol{X}_1 \dots \boldsymbol{X}_D$, with each matrix containing information on one data source and D being the number of data sources considered. For example, accelerometer data contains the device's acceleration along the x-, y- and z-axes over time. In general, PPMLAuth is flexible regarding the number and type of data sources. The preprocessing pipeline executes four transformations in sequence:

1. **Feature Extraction:** First, the matrices $\boldsymbol{X}_1 \dots \boldsymbol{X}_D$ are transformed separately for each data source into feature matrices $\boldsymbol{F}_1 \dots \boldsymbol{F}_D$ by extracting characteristic behavioral patterns [40].
2. **Normalization:** $\boldsymbol{F}_1 \dots \boldsymbol{F}_D$ are standardized to produce $\boldsymbol{S}_1 \dots \boldsymbol{S}_D$, which retain the same dimensions and structure but contain normalized values, facilitating stable and effective training of ML models.
3. **ML-based Compression:** Next, $\boldsymbol{S}_1 \dots \boldsymbol{S}_D$ are fed into one neural network each which compresses the matrix to a vector of fixed-length. We employ a transformer-based neural network architecture, which is well suited for supporting different input sizes and capturing long-range temporal dependencies in behavioral data [44]. This architecture aligns with strategies used in prior work [24,40]. The architecture comprises three convolutional layers, followed by a multi-head attention layer, a maximum pooling layer and a final dense layer. The result forms a fixed-length *embedding* with *128 entries*, yielding vectors $\boldsymbol{e}_1 \dots \boldsymbol{e}_D$ for the different data sources.
4. **Encryption:** The vectors $\boldsymbol{e}_1 \dots \boldsymbol{e}_D$ are concatenated to form a vector $\boldsymbol{ec}$, which is encrypted into a single ciphertext $Enc(pk_c, \boldsymbol{ec})$. The CKKS configuration must ensure that there are enough slots to store $\boldsymbol{ec}$ (see Section 5.1).

The output of the pipeline consists of a single ciphertext $Enc(pk_c, \boldsymbol{ec})$ (*behavioral encoding*), which is transmitted to the server.

3.3 ML-based Behavioral Analysis

Upon receiving the behavioral encoding from the client, the server performs an ML-based analysis to determine whether a legitimate user is present, with the design summarized on the right-hand side of Figure 1.

The encoding $\mathsf{Enc}(pk_c, \boldsymbol{ec})$ represents the currently observed behavior of the client. Additionally, the server retrieves N reference encodings, denoted as $\mathsf{Enc}(pk_c, \boldsymbol{h_1}) \dots \mathsf{Enc}(pk_c, \boldsymbol{h_N})$, from its local database. To ensure that N encodings are available, we employ the enrollment phase outlined in Section 3.1. Next, the current behavioral encoding $\mathsf{Enc}(pk_c, \boldsymbol{ec})$ and the encodings from previous sessions $\mathsf{Enc}(pk_c, \boldsymbol{h_1}) \dots \mathsf{Enc}(pk_c, \boldsymbol{h_N})$ are used as input for a composite ML model. This model consists of one sub-network per data source included in the behavioral encoding and a subsequent part for deriving the final result.

To this end, we first split each encoding into sub-encodings that represent individual data sources. As each sub-encoding spans a specific subset of the CKKS ciphertext's slots, we isolate them by multiplying with binary masks that contain ones in the desired slots and zeros elsewhere. Each sub-encoding is then processed by one of D sub-networks, all sharing the same architecture. They start with two convolutional layers, each followed by an average pooling layer. The result of the second average pooling layer is fed into a dense layer. Finally, the encrypted outputs of all sub-networks $\mathsf{Enc}(pk_c, \boldsymbol{p_1}) \dots \mathsf{Enc}(pk_c, \boldsymbol{p_D})$ are concatenated and analyzed by two consecutive dense layers. The output of the final dense layer $\mathsf{Enc}(pk_c, s)$ contains the calculated *score* s, which indicates whether the encodings are consistent in terms of user behavior.

We selected this architecture after exploring different model designs, aiming to strike the best balance between inference time, accuracy and flexibility. Except for the last layer, we use $f(x) = ax^2 + bx$ as activation function because it can be efficiently realized on encrypted data (see Section 2.1). The last layer applies the *sigmoid* activation $\sigma(x) = \frac{1}{1+e^{-x}}$. However, e^{-x} cannot be computed efficiently with HE [32]. Therefore, we disregard this activation function during encrypted inference and apply an adjusted decision threshold t' instead. This is possible since σ is strictly increasing, implying $\sigma(s) > t \iff s > \sigma^{-1}(t) =: t'$ for any original threshold t and output value s.

The neural network layers required for the analysis of the behavioral encodings and the subsequent aggregation are trained together on plain data, which is explained in Section 3.6. The main advantage of using a composite neural network is its ability to learn how to combine information from different data sources and its modularity, which allows the system to handle missing data sources. In contrast, existing work often relies on predefined linear combinations of analysis results from different data sources [40,39]. As we show in Appendix B, this is error-prone and prevents the incorporation of more complex dependencies.

After executing the neural network inference, the server uses the HE properties to subtract the threshold t' from the encrypted result $\mathsf{Enc}(pk_c, s)$. The resulting ciphertext ct is converted from CKKS to TFHE using the transformation key tk that was initially sent to the server (see Section 3.1). The transformation maps the slot that contains the calculated score s to a TFHE ciphertext cts_0. As

TFHE supports boolean arithmetic, we then evaluate whether the underlying value is greater than or equal to zero. Finally, we obtain a ciphertext ct_b with a value of 0 (accept) if the calculated score s is lower than the threshold t'.

However, the server cannot decrypt the ciphertext ct_b that contains the binary analysis result, as it lacks access to the secret key sk_t. For this reason, we employ a disclosure mechanism that securely reveals the result to the server.

3.4 Secure Disclosure

The secure disclosure procedure pursues two fundamental objectives:

1. **Classifier Decision Integrity:** The server learns the binary analysis result without the risk of manipulation by the client.
2. **Confidentiality:** Neither the client nor the server obtains behavioral information *other* than the binary analysis result.

To address these objectives, the server first transfers the encrypted analysis result ct_b to the client, which decrypts it to obtain the binary value b_c. Additionally, it generates a ZKP attesting that the decryption of ct_b yields b_c. The structure of the corresponding ZKP circuit is designed to confirm that (i) $\mathsf{hash}(sk_t) = com_{sk}$ *(Secret key commitment)* and that (ii) $\mathsf{Dec}(ctb, sk_t) = b_c$ *(Decryption correctness)*. Here, the variables used are defined as follows:

– sk_t: TFHE secret key of the client *(private input)*
– com_{sk}: commitment of the secret key as described in Section 3.1 *(public input)*
– ct_b: ciphertext encoding the binary result of the analysis *(public input)*
– b_c: claimed decryption result of ct_b *(public input)*

The use of the commitment com_{sk} is crucial to prevent manipulation by the client. Without this binding, a dishonest client can craft a specific secret key sk'_t that decrypts ct_b to a desired b'_c. By enforcing that the hash matches the initial commitment, the system ensures that the secret key is fixed during enrollment. We use the Poseidon hash function [19], which is optimized for ZKPs. This is important because the proof must be generated by the mobile client, which typically operates under limited computational resources (e.g., a smartphone).

Once generated, the proof is sent to the server together with b_c, the decryption of ct_b. The server verifies the ZKP and uses b_c to make an authentication decision if verification succeeds, rejecting the client otherwise. It is important to note that HE operations with CKKS and the scheme switching to TFHE introduce small error terms [28,46]. In principle, these errors could lead to a flipped binary value for b_c. However, with careful configuration and error management, such cases are very rare, as we show in our evaluation in Section 5.

To implement the ZKP, we employ the efficient Groth16 zk-SNARK system instantiated over the BN254 elliptic curve [20]. Since Groth16 requires a circuit-specific setup, the server generates the SRS once and derives the proving and verification keys. The server has a strong incentive to perform this setup honestly, as any leakage of the secret randomness would allow malicious clients to forge

proofs. If the server is not trusted to carry out this procedure alone, the setup can be realized via a multi-party computation in which multiple independent participants (e.g., server and client representatives) collaboratively generate the SRS, ensuring security as long as at least one participant behaves honestly [20].

3.5 Key Rollover Strategy

In a HE-based authentication setting, the server obtains one ciphertext and its corresponding plaintext during each result disclosure. Over time, this would allow the server to accumulate such pairs under the same keys, which could enable the exploitation of vulnerabilities in CKKS and TFHE (see Section 2.1). To prevent this, the client needs to periodically renew the cryptographic keys to ensure that only a limited number of these pairs are associated with a single key set. A naive key replacement would require re-establishing the user's behavioral baseline and cause noticeable disruption. Instead, we employ a *key rollover* strategy that gradually establishes new key material during regular authentication.

When a key rollover is initiated, the client generates a new CKKS key pair (pk'_c,sk'_c), and a transformation key tk' that results in a new TFHE key pair (pk'_t,sk'_t). The secret key sk'_t is used to derive the commitment com'_{sk}. Then, the client starts to send an additional behavioral encoding $\mathsf{Enc}(pk'_c, \boldsymbol{ec})$ and the new commitment com'_{sk} to the server during each upcoming authentication.

The server constructs a baseline of reference encodings for the new key pair if R_{count} successful authentications occur consecutively with an unchanged commitment. Any failed authentication or inconsistent commitment invalidates the current run of successes. Here, $R_{count} \geq N$ is a configurable system parameter that must be chosen to be at least as large as the number of behavioral encodings required for analysis (see Section 3.3). If $R_{count} > N$, an application-defined policy determines how the required N reference encodings are derived. Further, R_{count} must be adapted to the effectiveness of the ML-based analysis to ensure that an attacker cannot override the baseline. We discuss the security implications of the key rollover and its parameterization in Section 4 and Section 5.

To finalize the rollover, the client sends the public key material (pk'_c, pk'_t, tk') to the server. This transmission may also occur earlier when a stable network connection (e.g., WiFi) is available to avoid interruptions. After completion, the client sends behavioral encodings only under the new key pair and deletes the old cryptographic keys $(pk_c, sk_c, tk, pk_t, sk_t)$.

PPMLAuth initiates a key rollover after R_{start} authentication attempts with the same key set. If the rollover is not completed by the R_{max}-th attempt, the client destroys the current keys and triggers re-enrollment with a new key set, which is identical to the initial enrollment phase (see Section 3.1). Therefore, the sequence of authentications starting after the R_{start}-th attempt and ending with the R_{max}-th attempt forms the *rollover window*. The server processes rollover-related data only within this window. In this way, both the client and the server can enforce timing and length of the rollover. The upper bound of R_{max} prevents the server from accumulating a critical number of ciphertext–plaintext pairs when a key rollover does not complete within the rollover window. Accordingly,

R_{max} should reflect the current state-of-the-art in cryptanalysis [8]. To avoid disruptive re-enrollments, R_{start} should be chosen to make rollover completion highly likely before reaching R_{max}. In Section 5, we propose concrete values for the parameters and examine their impact on security and usability.

3.6 ML Model Training and Configuration

Our framework uses ML models in preprocessing (see Section 3.2) and for behavioral analysis (see Section 3.3), each of which requires proper training.

The training requires access to datasets containing behavioral information associated with user identities. Furthermore, as the training of ML models on encrypted data is not yet feasible [46], the data needs to be available in unencrypted form. Here, it is possible to make use of various recent datasets [39,33,38,40]. In practice, this does not pose a significant limitation, as the generalizability of behavioral authentication methods is demonstrated in our evaluation in Section 5.3 and further substantiated in recent research [27,24]. Note that the inability to efficiently train ML models on encrypted data also prevents the use of user-specific models, which would require training on each user's behavioral data.

First, we consider each data source individually and train the preprocessing neural network. To this end, we employ contrastive learning based on triplet loss [35], which we found to be particularly effective in our scenario. By explicitly enforcing intra-user similarity and inter-user separation in the embedding space, triplet loss produces representations suitable for behavioral analysis [1,40].

Subsequently, we train the composite model that analyzes embeddings from multiple data sources (see Section 3.3). The preprocessing models are applied to generate embeddings for behavioral information captured during user sessions. Training samples for the composite model are then constructed either from two sessions of the same user, which the model should classify as consistent, or from sessions of different users, which the model should classify as inconsistent. In this way, the composite model learns to compare embeddings and merge analysis results across data sources. To enable modularity, embeddings from individual data sources are randomly omitted in some training samples, allowing the model to learn to operate with missing inputs [40].

After training the composite model, it is used to determine the threshold t' (see Section 3.3), which maximizes separation between legitimate and illegitimate use. Authentication is successful when a score falls below this threshold.

4 Security and Privacy Analysis

In this section, we define the threat model and provide a summary of the achieved security guarantees of PPMLAuth, together with a proof sketch. Formal specifications and the corresponding proofs are presented in Appendix A.

4.1 Adversary and Trust Assumptions

We analyze PPMLAuth under the following adversarial model.

Definition 1 (Adversary Model). *We consider a probabilistic polynomial-time adversary $\mathcal{A}$ that may* statically *corrupt either the client or the server, in which case the corrupted party is malicious and may arbitrarily deviate from the protocol in each protocol invocation [16].*

If the client is corrupted, $\mathcal{A}$ controls the behavioral authentication input and all client-side protocol actions. If the server is corrupted, $\mathcal{A}$ observes all protocol messages and local state and controls all server-side protocol actions.

Enrollment assumption. *Enrollment and re-enrollment are gated by conventional authentication mechanisms (e.g., multi-factor authentication) that are secure against impersonation, even if the client device is compromised.*

Setup assumption. *The initial trusted setup of the ZKP (see Section 3.4) includes at least one honest participant, ensuring that no adversary learns the SRS trapdoor and can break zk-SNARK soundness.*

Scope. *We focus on threats originating from participants in the authentication process via protocol messages and explicit outputs, excluding denial-of-service, selective abort, side-channel leakage, and external threats such as network attacks, which must be mitigated by standard security mechanisms [16].*

4.2 Security Properties

The security properties that PPMLAuth aims to achieve concern the integrity of the authentication decision, the secure transition to new key material, the mitigation of HE vulnerabilities, and the confidentiality of behavioral data.

Definition 2 (Classifier Decision Integrity). *The decision bit learned by the server coincides, except with negligible probability, with the result of the ML-based analysis (classifier). Note that integrity is defined with respect to the classifier output, not the biometric ground truth.*

Definition 3 (Key-Exposure Boundedness). *The rollover policy establishes a limit on the number of ciphertext–plaintext pairs that can be observed by the server under the same cryptographic key material.*

Definition 4 (Profile Update Security). *Within a rollover window spanning T authentication attempts, a malicious client cannot cause a profile update whose determining inputs are adversary-submitted, except with probability at most $T * p^{R_{\mathsf{count}}}$, where p upper-bounds per-attempt acceptance of any malicious client.*

Definition 5 (Restricted Confidentiality). *PPMLAuth protects the confidentiality of behavioral data, up to the minimum leakage unavoidable in the setting considered. Confidentiality is* restricted *in the following sense:*

(i) The client inevitably learns the authentication decision, and
(ii) A malicious server can, in each attempt, force the disclosure of at most a single bit of information via the result–decryption step.

Given the considered scenario, these bounds are optimal, as discussed in Appendix A.2, where we separate client-side and server-side notions for clarity.

4.3 Realization Argument

We briefly outline the proof intuition why PPMLAuth realizes the security properties defined in Section 4.2, with the proofs given in Appendix A.3.

Theorem 1. *Under the semantic security of the HE schemes used, the soundness and zero-knowledge properties of the zk-SNARK, and the enforced rollover policy, PPMLAuth satisfies Classifier Decision Integrity, Key-Exposure Boundedness, Profile Update Security, and Restricted Confidentiality.*

Proof (Intuition). Classifier Decision Integrity follows from secure disclosure via zk-SNARKs. *Key-Exposure Boundedness* follows from the rollover mechanism that limits the lifetime of any key material. *Profile Update Security* follows from the rollover design, which requires R_{count} consecutive successful authentications before updating the baseline. *Restricted Confidentiality* follows from the semantic security of the HE schemes together with the inherent leakage bounds.

4.4 Limitations

Our guarantees hold under the adversary and trust assumptions of our model. We highlight the following limitations:

(i) **Restricted confidentiality:** in case of a malicious server, information of up to a single bit per authentication attempt can be revealed. While the per-attempt leakage is strictly bounded, its long-term accumulation can be critical, as it grows linearly with the number of authentication attempts. For example, with an authentication frequency of one attempt every 60 seconds, this corresponds to a leakage rate of one bit per minute. However, in the same time interval, high-dimensional behavioral embeddings on the order of tens of kilobytes are transmitted, which renders the relative leakage of PPMLAuth minimal. Moreover, most existing approaches either expose substantially richer information or rely on the strong assumption of an honest server (see Section 6).

(ii) **Authenticated enrollment:** enrollment and re-enrollment rely on conventional authentication mechanisms that are secure against impersonation. If an adversary can successfully bypass these mechanisms, no protocol-level guarantees can prevent corruption of the reference profile, as compromised enrollment is a fundamental limitation of authentication in general [1,40].

(iii) **Classifier error:** PPMLAuth preserves the integrity of the classifier decision but does not reduce inherent misclassifications. Such errors need to be addressed at the level of ML model design, training and optimization.

(iv) **Acceptance bound:** The security guarantee of Definition 4 depends heavily on the per-attempt acceptance bound p, which should account for repeated and adaptive attack strategies to ensure a sound practical interpretation.

(v) **Parameter dependence:** while the above properties hold under our model, their practical security margins depend on choosing R_{max} and R_{count} in accordance with classifier effectiveness and the current state of cryptanalysis.

5 Evaluation

In this section, we first describe the evaluation design (Section 5.1) and the considered scenario (Section 5.2). We subsequently present results on authentication effectiveness (Section 5.3) and efficiency (Section 5.4).

5.1 Evaluation Design

We first present the metrics and datasets used in our evaluation.

Evaluation Metrics. We rely on three groups of metrics. First, we use common measures such as overall accuracy (ACC) and the area under the receiver operating characteristic curve (AUC) [40] to quantify authentication quality. The receiver operating characteristic curve is obtained by plotting the true positive rate against the false positive rate at different decision thresholds, where a higher AUC indicates better classification performance.

Second, we use authentication-specific metrics. The false acceptance rate (FAR) measures how often illegitimate users are incorrectly authenticated, while the false rejection rate (FRR) captures how often legitimate users are not successfully authenticated. The equal error rate (EER) is obtained at the threshold where FAR and FRR are equal, with lower values indicating better authentication effectiveness. Since AUC and EER require access to continuous score distributions, we temporarily decrypt the aggregated behavioral scores to compute these metrics, enabling threshold-independent comparisons with prior work.

Third, to interpret latency and network traffic, we rely on statistical measures like minimum (min), maximum (max), arithmetic mean (average, μ), and standard deviation (σ).

Evaluation Datasets. Our evaluation relies on two datasets that incorporate multiple data sources for benchmarking behavioral authentication systems.

BehavePassDB: The BehavePassDB dataset [39] was collected in a controlled scenario with 81 users, who were requested to perform different tasks on a mobile device. The dataset includes touchscreen inputs as well as readings from the accelerometer, gyroscope, linear accelerometer, magnetometer, and gravity sensor. Furthermore, BehavePassDB is split into three subsets: training set (51 users), validation set (10 users) and test set (21 users).

A key feature of the dataset is the inclusion of two attack scenarios: *random attackers* who use a different device without knowledge of the user's behavior, and *skilled attackers* who use the same device and have behavioral information. This allows us to test the ability of the authentication to prevent mimicry attacks.

BrainRun: The BrainRun dataset [33] was collected through a public mobile gaming application. It includes touchscreen gestures and sensor data, such as accelerometer, gyroscope and magnetometer readings. A key characteristic of this dataset is that it was collected from 2218 users under real-world conditions. However, only 420 users have at least three gameplay sessions with sufficient data to conduct behavioral analysis.

5.2 Evaluation Scenario

Based on the datasets introduced in Section 5.1, we emulate a realistic scenario to evaluate the entire authentication process of PPMLAuth. We first train the ML models according to the procedure from Section 3.6. Here, we use the predefined dataset split for BehavePassDB and a 70/10/20 user-level split for BrainRun (training/validation/testing). The validation set is used to tune hyperparameters (e.g. learning rate), while the test set is used to collect the final results. Since BrainRun lacks a dedicated test split, prior work has also adopted user-specific models [25,24] that raise concerns regarding scalability and computational effort (see Section 3.6) [40,25]. Therefore, we compare PPMLAuth only with methods that follow the same user-independent approach.

Each usage session is divided into 40-second slices, and authentication is performed by comparing a query slice against a single reference slice that represents enrollment data (corresponding to $N = 1$, see Section 3.3). Legitimate and illegitimate samples are generated by pairing slices: for BehavePassDB, we use the provided pairs, while for BrainRun, slices are paired either from the same user (legitimate) or from different users (illegitimate), ensuring that pairs do not originate from the same session and that the dataset remains balanced. This pairing strategy reflects real-world usage and aligns with existing work [1].

We parameterize the key rollover mechanism with $R_{start} = 4500$, $R_{max} = 5000$, and $R_{count} = 12$, chosen conservatively given that current attacks on CKKS and TFHE require more than 10^4 ciphertext–plaintext pairs to have high success probability [8,11]. Due to the limited number of sessions per user in both datasets, a natural rollover at R_{start} authentications cannot be observed. Therefore, we explicitly initiate rollover processes for 10 % of the users, while keeping all other parameters unchanged. This allows us to assess overhead, convenience, and security implications of the rollover mechanism.

For the execution of neural network inferences on encrypted data, we use OpenFHE [3] with CKKS as a leveled HE scheme to avoid costly bootstrapping [28]. We use a CKKS ring dimension of $N = 2^{14}$, corresponding to 8192 slots, and enable scheme switching to TFHE for non-linear operations. The chosen parameterization targets 128-bit security, with TFHE ciphertext modulus $\log Q_{\text{ccLWE}} = 16$. For secure disclosure, the ZKP proof and verification system is built on *gnark* [10], and the resulting circuit consists of 128,317 constraints.

To obtain reliable results, the experiment is run 20 times. Subsequently, we consider the median for individual metrics (e.g., accuracy) and analyze the totality of measurements otherwise (e.g., latencies). The setup uses a virtual machine with 4 vCPUs and 8 GB of RAM to emulate clients, while the server runs on a system with two AMD EPYC 7F32 8-core processors and 128 GB of RAM. Data transferred over the network is compressed using Zstandard to reduce size.

5.3 Authentication Effectiveness

We first outline key differences between ML-based and traditional distance-based methods and then assess the authentication effectiveness of PPMLAuth in detail.

ML-based vs Distance-based Approaches. While distance-based classifiers offer fast and interpretable assessments, they are limited to simple comparisons, whereas ML-based methods can learn complex patterns across multiple data sources. We conducted a comparison of both approaches, with details provided in Appendix B. To keep the focus of the evaluation on our framework, we summarize only key observations here. Taken together, ML models achieve higher accuracy when fusing multiple data sources, make more effective use of additional reference samples, and show greater robustness against advanced attackers.

Overall Authentication Effectiveness. Table 1 provides a comprehensive overview of the authentication effectiveness of PPMLAuth across both datasets when considering all available data sources, and compares the results with state-of-the-art approaches. To assess generalization across different datasets, models are additionally trained on BrainRun and evaluated on BehavePassDB, and vice versa. Moreover, to quantify performance degradation introduced by HE, we report results obtained without HE, retaining the original ReLU activation functions instead of polynomial approximations (see Section 3.3). For BehavePassDB, we are limited to the metrics provided by the evaluation platform [39], which is why we provide AUC values for the random attacker scenario (R-AUC), for the skilled attacker scenario (S-AUC) and overall (O-AUC) (see Section 5.1).

Table 1: Comparison of the authentication effectiveness with existing work

Dataset: BehavePassDB

Method	O-AUC%	R-AUC%	S-AUC%
B2CAR [4]	70.63	81.26	60.00
Reference [39]	72.92	80.74	65.11
OPPAoM [31]	68.44	77.38	59.50
$PPMLAuth_{\mathrm{BR}}$	70.12	82.32	57.92
$PPMLAuth^{*}_{\mathrm{BR}}$	70.61	83.16	58.06
$PPMLAuth_{\mathrm{BP}}$	74.72	86.52	62.92
$PPMLAuth^{*}_{\mathrm{BP}}$	75.22	87.31	63.13

Dataset: BrainRun

Method	EER%	AUC%	ACC%	FAR%	FRR%
SPC [29]	15.69	–	–	–	–
$PPMLAuth_{\mathrm{BP}}$	26.90	79.69	67.11	51.09	13.42
$PPMLAuth^{*}_{\mathrm{BP}}$	26.03	80.62	68.30	50.37	12.92
$PPMLAuth_{\mathrm{BR}}$	16.69	91.53	82.89	13.09	21.94
$PPMLAuth^{*}_{\mathrm{BR}}$	*15.65*	*92.33*	*84.54*	*11.39*	*21.57*

* Non-HE variant (no privacy protection)
$\mathrm{PPMLAuth}_{\mathrm{BP}}$: trained on BehavePassDB $\mathrm{PPMLAuth}_{\mathrm{BR}}$: trained on BrainRun

For the BehavePassDB dataset, our approach PPMLAuth in its HE-based variant ($\mathrm{PPMLAuth}_{\mathrm{BP}}$) yields an overall AUC of 74.72 %, thus outperforming all existing approaches, including the only related approach that considers user privacy (OPPAoM) [31] by 5.98 percentage points.

Regarding the BrainRun dataset, PPMLAuth achieves an EER of 16.69 % with privacy protection ($\mathrm{PPMLAuth}_{\mathrm{BR}}$), which is on the same level as comparable work [29] that reaches an EER of 15.69 % without considering user privacy. This performance is also consistent with work that considers similar scenarios on different datasets, where EER values between 15–18 % were observed [26].

Across both datasets, the performance drop attributed to the application of HE ranges from 0.14 to 1.7 percentage points across all metrics, implying that HE has a minor impact on overall effectiveness. Furthermore, the cross

dataset evaluation shows only a moderate performance reduction when training the models on a different dataset, which indicates robust generalization.

Taken together, the results confirm that PPMLAuth matches or exceeds the effectiveness of existing approaches, even though our approach provides stronger privacy guarantees. Moreover, through repeated authentication, PPMLAuth reduces the likelihood that an impostor remains undetected over time.

Data-Source-Specific Effectiveness. Next, we integrate one modality at a time and compute the resulting AUC to be able to quantify the performance gain achieved by data source fusion. Table 2 summarizes the findings, reporting the AUC values for the individual data sources on both datasets.

Table 2: Authentication effectiveness (AUC) for individual data sources

Dataset	A	L	Gy	Gr	M	T
BehavePassDB	70.74%	71.70%	62.42%	60.33%	65.79%	68.12%
BrainRun	75.00%	74.85%	72.49%	—	85.22%	62.94%

A: Accelerometer, **L**: Linear accelerometer, **Gy**: Gyroscope, **Gr**: Gravity sensor, **M**: Magnetometer, **T**: Touchscreen interactions.

Comparing the unimodal results with the overall AUC values presented in Table 1, it can be observed that PPMLAuth consistently improves authentication accuracy by fusing multiple behavioral data sources. For the BehavePassDB dataset, the accelerometer achieves the best unimodal performance with an AUC of 71.70 %, which increases to 74.72 % when combining all available data sources. The effect is even more pronounced for the BrainRun dataset, where the strongest individual modality (magnetometer) reaches an AUC of 85.22 %, while the ML-based fusion achieves 91.53 %.

5.4 Authentication Efficiency

To ensure applicability, PPMLAuth must achieve acceptable authentication latency and network overhead. We consider three different types of tasks that arise throughout PPMLAuth's lifecycle, which differ in frequency and in how strongly they affect runtime efficiency. Below, we discuss each category in turn, followed by a discussion about general scalability considerations.

Global Setup. Tasks that need to be carried out once for all users can be performed offline and do not impact runtime efficiency (e.g., model training and ZKP setup). For this reason, we do not consider them further here.

Initialization and Key Rollovers. The initial tasks performed by each client largely overlap with those of a key rollover. Table 3 provides an overview of these tasks along with their associated execution times and network traffic, including additional encryptions that are only part of key rollovers.

Table 3: Execution times (in seconds) and average network traffic for client-side tasks during initial setup and key rollovers

Task	min	μ	max	σ	∅Net. Traffic
CKKS Key Generation	0.053	0.055	0.074	0.002	59.616 MB
Transformation/TFHE Key Generation	9.162	10.112	11.368	0.671	153.791 MB
Additional Encryption and Commitment	0.041	0.0439	0.049	0.002	2.505 MB
Total	**9.258**	**10.211**	**11.466**	**0.671**	**215.927 MB**

On average, the required operations take 10.21 seconds (s) and transfer 215.93 megabytes (MB) from the client to the server. The overhead of the additional encryptions depends on the number of iterations required for a successful rollover. In particular, each additional encryption and key commitment jointly add approximately 0.06 s of latency and 2.50 MB of network traffic to each authentication until the rollover is completed. Generating the transformation key and the TFHE key material dominates the overall latency, with an average of 10.11 s, while the public key material constitutes the bulk of the network traffic with 213.41 MB. In practice, this overhead can be mitigated by generating key material in advance or during idle periods. Additionally, distributing the transfer of public key material over the entire key rollover duration enables deployment even on mobile devices with limited bandwidth.

Based on the authentication effectiveness observed for BrainRun (see Table 1), the behavior of a key rollover can be quantified more precisely. With $R_{count} = 12$ and a FRR of 21.94 % achieved by PPMLAuth, the expected rollover length N_{rl} corresponds to $\mathbb{E}[N_{rl}] = \frac{1-(1-\mathrm{FRR})^{10}}{(1-\mathrm{FRR})^{10}*\mathrm{FRR}} \approx 84.5$ authentication processes. It follows that the probability of a rollover not being completed after $T = R_{max} - R_{start} = 500$ authentications and thus requiring a re-enrollment is extremely low ($\approx 0.3\,\%$). This leads to an average amortized network traffic of $\frac{85*2.50\,\mathrm{MB}}{4585} \leq 0.046\,\mathrm{MB}$ per authentication, which can be handled efficiently during regular operation. For a conservative estimate of an adversary's per-attempt success probability, we set p to the 95th-percentile FAR observed across all users (i.e., $p = 0.33$), thereby capturing advanced imposter cases, such as repeated or adaptive attack strategies while avoiding extreme outliers. Using this estimate, the probability that an adversary bypasses the authentication R_{count} successive times and thus manipulates the behavioral baseline during a rollover window is at most $T * p^{R_{count}} = 500 * 0.33^{12} \leq 0.00084$ (see Definition 4). All in all, these findings demonstrate that, with appropriate configuration, PPMLAuth enables secure key rollovers while preserving seamless and efficient authentication.

Authentication Process. Per-authentication effort represents the core runtime costs and is most critical for applicability. To quantify this effort, we measure both latency and network traffic of all authentication steps, neglecting network-induced latency as it highly depends on the actual connection. Table 4 summarizes the results for a single authentication process on both datasets.

Table 4: Execution times (in seconds) and average network traffic incurred for each step of the authentication process

BehavePassDB (6 data sources)

Role	Step	*min*	μ	*max*	øNetw.
C	Preprocess	0.06	0.08	0.23	2.50 MB
S	ML Analysis	3.20	3.29	3.59	–
S	Thresholding	0.80	0.92	1.47	10.2 KB
C	ZK Proof	1.10	1.39	1.93	0.21 KB
S	ZK Verify	<0.01	<0.01	0.01	–
Total		**5.24**	**5.69**	**6.34**	**2.51 MB**

BrainRun (5 data sources)

Role	Step	*min*	μ	*max*	øNetw.
C	Preprocess	0.05	0.08	0.20	2.50 MB
S	ML Analysis	2.65	2.73	2.99	–
S	Thresholding	0.77	0.92	1.50	10.2 KB
C	ZK Proof	1.10	1.22	1.60	0.21 KB
S	ZK Verify	<0.01	<0.01	0.01	–
Total		**4.69**	**4.95**	**5.62**	**2.51 MB**

C = client, S = server, øNetw. = average network traffic

The average cumulated latency amounts to 5.69 s for the BehavePassDB dataset and 4.95 s for the BrainRun dataset. As authentication can be performed in the background without requiring explicit user interaction, this latency does not directly impact usability. The ML-based analysis dominates the latency with an average duration between 2.73 s and 3.29 s due to the computational effort of HE. The scheme conversion and the subsequent application of the threshold requires 0.92 s on average. Finally, the ZKP generation and verification together take 1.22 s–1.39 s on average. Notably, ZKP generation is the only computationally intensive operation performed on the client. However, as authentication is triggered only intermittently, the load remains manageable.

The network traffic can be broken down into three transmissions. First, the transfer of the behavioral encoding to the server comprises 2.50 MB for both datasets, as a single CKKS ciphertext is sufficient to hold all embeddings. The second transmission consists of a TFHE ciphertext sent from the server to the client, which encapsulates the authentication decision and has a size of 10.2 KB. The third transmission involves sending one binary value and the generated ZKP from the client to the server. Together, the ZKP and the analysis result are 0.21 KB in size. Therefore, the average amount of data transferred per authentication process totals 2.51 MB, which remains within reasonable bounds.

Scalability Discussion. We consider three dimensions of scalability.

First, with respect to the number of data sources, Table 4 shows that increasing the number of sources from five (BrainRun) to six (BehavePassDB) raises preprocessing latency from 30 ms to 40 ms and ML-based analysis time from 2.73 s to 3.29 s. This indicates appropriate scaling, as preprocessing is parallelizable across sources (see Section 3.2) and ML-based analysis exhibits only a linear increase in latency. Moreover, our configuration supports bundling up to $\frac{\text{Slot count}}{\text{Embedding size}} = 64$ data sources into a single CKKS ciphertext, so network traffic remains largely independent of the number of sources.

Second, scalability with respect to the number of reference embeddings is favorable, as the ML model architecture used for behavioral analysis can remain unchanged, assuming sufficient model capacity. Consequently, increasing the number of reference embeddings causes only a minor increase in computa-

tional effort. In contrast, existing approaches based on distance metrics require explicit pairwise comparisons between each input and all stored references.

Third, scalability with respect to the number of users concerns the server's ability to process many authentication requests concurrently. Since authentication processes are independent of each other, HE-based neural network inference can be parallelized across users, e.g., on separate cores or machines. As a result, per-authentication latency can be maintained even as the number of users increases, provided that computational resources are scaled accordingly.

Taken together, the efficiency and scalability of PPMLAuth support its practical applicability. Existing strategies that combine HE with ML report per-authentication latencies above 6 s [30]. An exception is OPPAoM [31], which achieves lower latencies but operates on a substantially less complex behavioral analysis, which is reflected in the degraded effectiveness (see Table 1).

6 Related Work

Prior work on privacy-preserving authentication, including biometric modalities as well as behavioral approaches, can be grouped into three categories.

The first category comprises approaches based on additive HE schemes that do not support ML on encrypted data. Baig et al. [5,6] propose privacy-preserving protocols for continuous authentication using the Paillier cryptosystem in combination with oblivious transfer to disclose authentication outcomes. Govindarajan et al. [18] introduce an authentication protocol for touchscreen interactions that relies on the DGK HE scheme [14]. Shahandashti et al. [36] combine HE with order-preserving encryption to enable privacy-preserving authentication in the presence of malicious clients. Similar techniques based on the Paillier scheme have also been applied to physiological biometrics such as face and fingerprint recognition [17]. With the exception of Baig et al. [6], the approaches in this category assume an honest client or server, which severely limits their applicability in real-world scenarios. In addition, the use of HE schemes supporting only additive operations limits expressiveness, robustness against attackers, and the effective fusion of heterogeneous data sources (see Appendix B).

The second category consists of HE-based approaches that leverage modern schemes, such as CKKS, to enable ML-based analysis on encrypted data. In our prior work [30,31], we propose two strategies in which data is encrypted on the client and transmitted to the server for analysis with neural networks operating directly on ciphertexts. To reveal the analysis result to the server, a commutative re-encryption protocol is employed [7]. However, this design reveals entire CKKS ciphertexts, which endangers confidentiality by leaking excessive amounts of sensitive information. Additionally, it does not protect result integrity against malicious clients, enabling manipulation of the authentication outcome. Sumalatha et al. [41] combine CKKS with deep learning for physiological biometrics such as fingerprint recognition, but rely on client-side decryption, which introduces additional trust assumptions about the client. In contrast to PPMLAuth, these approaches further do not incorporate cross-source dependencies in their ML-

based analysis, leading to reduced authentication performance (see Section 5.3), and do not account for vulnerabilities of CKKS, allowing adversaries to potentially infer clients' secret keys.

The third category summarizes approaches that use techniques other than HE. One line of work relies on cryptographic protocols such as secure multi-party computation (MPC). For example, Domingo-Ferrer et al. [15] introduce a protocol based on private set intersection to compute similarities, while a more recent approach [9] employs MPC designs with additional helper servers for fingerprint authentication. Related approaches also include strategies based on format-preserving encryption, such as SmartCAMPP [22], which enable ML-based analysis. Another line of work leverages trusted execution environments (TEEs) to perform authentication within hardware-protected enclaves [37]. Finally, several approaches protect privacy by removing or transforming sensitive information prior to processing. Vassallo et al. [43] and Sun et al. [42] propose obfuscation techniques, while Hatin et al. [21] and Sitová et al. [38] design transformations that preserve comparability. Overall, data transformation and format-preserving techniques provide substantially weaker privacy and security guarantees [45], while MPC- and TEE-based approaches either limit analysis expressiveness, rely on strong trust assumptions, or provide limited transparency into the authentication logic, particularly for complex ML models.

7 Conclusion

We present *PPMLAuth (Privacy-Preserving and ML-based Authentication)*, a framework for privacy-preserving authentication in client–server settings based on behavioral analysis with machine learning (ML). PPMLAuth preprocesses behavioral data on the client device, encrypts it with the Cheon-Kim-Kim-Song (CKKS) homomorphic encryption scheme and sends it to the server. The server leverages the homomorphic properties to perform an ML-based analysis that fuses behavioral data sources to improve authentication performance. It derives an encrypted binary decision by scheme switching to Fast Fully Homomorphic Encryption over the Torus (TFHE). The client decrypts the result and provides a zero-knowledge proof to confirm correctness, while also preserving integrity and confidentiality. PPMLAuth addresses complexity, security and privacy issues in prior work and includes efficient key rollovers that update cryptographic material during regular operation, mitigating vulnerabilities and avoiding disruptions.

We analyzed the security, privacy, effectiveness, and efficiency characteristics of PPMLAuth based on a detailed threat model and two public datasets. We showed that ML-based behavioral analysis outperforms distance-based approaches and generalizes well, with only moderate performance loss in cross-dataset evaluations. Our framework benefits from multiple data sources, achieving effectiveness comparable to state-of-the-art approaches without privacy protection. In particular, performance loss from homomorphic encryption is minimal. Finally, efficiency analysis shows reasonable latency and traffic, with key rollovers incurring only infrequent and manageable additional costs.

Future work focuses on adopting advances in homomorphic encryption [46,47], zero-knowledge proofs [34], and ML-based behavioral analysis [1,24], which hold strong potential for further improving PPMLAuth. We additionally plan to evaluate mobile-side energy consumption, explore alternative ML model architectures, analyze numerical stability in CKKS under realistic operating conditions, and assess security aspects of the employed ML models (e.g., data poisoning).

Acknowledgements

The authors used ChatGPT5 and DeepL Write to revise individual sentences and correct grammatical errors. This work was supported by the projects KIWI (16KIS1142K) funded by the German Federal Ministry of Education and Research (BMBF), the state project bwNET2.0 funded by the Ministry of Science, Research and Arts Baden-Württemberg (MWK), and the aura.ai project co-funded by the European Union through the Interreg Upper Rhine program.

References

1. Abuhamad, M., Abusnaina, A., Nyang, D., Mohaisen, D.: Sensor-based continuous authentication of smartphones' users using behavioral biometrics: a contemporary survey. IEEE Internet of Things Journal **8**(1), 65–84 (2021)
2. Aharoni, E., Adir, A., Baruch, M., Drucker, N., Ezov, G., Farkash, A., et al.: HeLayers: A tile tensors framework for large neural networks on encrypted data. In: Proc. 23rd Privacy Enhancing Technology Symp. (PETS). pp. 325–342 (2023)
3. Al Badawi, A., Bates, J., Bergamaschi, F., Cousins, D.B., Erabelli, S., Genise, N., et al.: OpenFHE: Open-source fully homomorphic encryption library. In: Proc. 10th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography (WAHC). pp. 53–63 (2022)
4. Al Samara, M., Gilg, M., Abouaissa, A., Bennis, I., Lorenz, P.: B2CAR: Behavioural biometrics for continuous authentication with regularisation techniques. In: Proc. 21st IEEE Int. Conf. on Wireless Communications and Mobile Computing (IWCMC). pp. 324–329 (2025)
5. Baig, A.F., Eskeland, S., Yang, B.: Privacy-preserving continuous authentication using behavioral biometrics. Springer Int. Journal of Information Security **22**, 1833–1847 (2023)
6. Baig, A.F., Eskeland, S., Yang, B.: Novel and efficient privacy-preserving continuous authentication. MDPI Cryptography **8**(1), 1–14 (2024)
7. Baumstark, P., Monschein, D., Waldhorst, O.P.: Secure plaintext acquisition of homomorphically encrypted results for remote processing. In: Proc. 48th IEEE Int. Conf. on Local Computer Networks (LCN). pp. 1–4 (2023)
8. Bergamaschi, F., Costache, A., Dachman-Soled, D., Kippen, H., LaBuff, L., Tang, R.: Revisiting the security of approximate FHE with noise-flooding countermeasures. In: Proc. 28th IACR Int. Conf. on Practice and Theory of Public-Key Cryptography (PKC). pp. 102–134 (2025)
9. Blanton, M., Murphy, D.: Privacy preserving biometric authentication for fingerprints and beyond. In: Proc. 14th ACM Conf. on Data and Application Security and Privacy (CODASPY). pp. 367–378 (2024)

10. Botrel, G., Piellard, T., Housni, Y.E., Kubjas, I., Tabaie, A.: Consensys/gnark: v0.14.0 (2025), https://doi.org/10.5281/zenodo.5819104
11. Cheon, J.H., Choe, H., Passelègue, A., Stehlé, D., Suvanto, E.: Attacks against the IND-CPAD security of exact FHE schemes. In: Proc. 31st ACM SIGSAC Conf. on Computer and Communications Security (CCS). pp. 2505–2519 (2024)
12. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Proc. 23rd IACR Int. Conf. on Theory and Application of Cryptology and Information Security (ASIACRYPT). pp. 409–437 (2017)
13. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: Fast fully homomorphic encryption over the torus. Springer Journal of Cryptology **33**(1), 34–91 (2020)
14. Damgard, I., Geisler, M., Kroigard, M.: Homomorphic encryption and secure comparison. Inderscience Publishers Int. Journal of Applied Cryptography **1**(1), 22–31 (2008)
15. Domingo-Ferrer, J., Wu, Q., Blanco-Justicia, A.: Flexible and robust privacy-preserving implicit authentication. In: Proc. 30th IFIP Int. Conf. on ICT Systems Security and Privacy Protection (SEC). pp. 18–34 (2015)
16. Goldreich, O.: Foundations of cryptography: Volume 2, basic applications. Cambridge University Press, 1st edn. edn. (2009)
17. Gomez-Barrero, M., Maiorana, E., Galbally, J., Campisi, P., Fierrez, J.: Multi-biometric template protection based on homomorphic encryption. Elsevier Pattern Recognition **67**, 149–163 (2017)
18. Govindarajan, S., Gasti, P., Balagani, K.S.: Secure privacy-preserving protocols for outsourcing continuous authentication of smartphone users with touch data. In: Proc. 6th IEEE Int. Conf. on Biometrics: Theory, Applications and Systems (BTAS). pp. 1–8 (2013)
19. Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., Schofnegger, M.: POSEIDON: A new hash function for zero-knowledge proof systems. In: Proc. 30th USENIX Security Symp. pp. 519–535 (2021)
20. Groth, J.: On the size of pairing-based non-interactive arguments. In: Proc. 35th IACR Int. Conf. on the Theory and Applications of Cryptographic Techniques (EUROCRYPT). pp. 305–326 (2016)
21. Hatin, J., Cherrier, E., Schwartzmann, J.J., Rosenberger, C.: Privacy preserving transparent mobile authentication. In: Proc. 3rd Int. Conf. on Information Systems Security and Privacy (ICISSP) (2017)
22. Hernández-Álvarez, L., de Fuentes, J.M., González-Manzano, L., Hernández Encinas, L.: SmartCAMPP - smartphone-based continuous authentication leveraging motion sensors with privacy preservation. Elsevier Pattern Recognition Letters **147**, 189–196 (2021)
23. Hernández-Álvarez, L., de Fuentes, J.M., González-Manzano, L., Hernández Encinas, L.: Privacy-preserving sensor-based continuous authentication and user profiling: A review. MDPI Sensors **21**(1), 92 (2021)
24. Hu, M., Zhang, K., You, R., Tu, B.: AuthConFormer: Sensor-based continuous authentication of smartphone users using a convolutional transformer. Elsevier Computers & Security **127**, 103122 (2023)
25. Hu, M., Zhang, K., You, R., Tu, B.: Multisensor-based continuous authentication of smartphone users with two-stage feature extraction. IEEE Internet of Things Journal **10**(6), 4708–4724 (2023)
26. Levi, M., Hazan, I., Agmon, N., Eden, S.: Behavioral embedding for continuous user verification in global settings. Elsevier Computers & Security **119**, 102716 (2022)

27. Lin, C., He, J., Shen, C., Li, Q., Wang, Q.: CrossBehaAuth: Cross-scenario behavioral biometrics authentication using keystroke dynamics. IEEE Trans. on Dependable and Secure Computing **20**(3), 2314–2327 (2023)
28. Marcolla, C., Sucasas, V., Manzano, M., Bassoli, R., Fitzek, F.H.P., Aaraj, N.: Survey on fully homomorphic encryption, theory, and applications. Proc. IEEE **110**(10), 1572–1609 (2022)
29. Monschein, D., Waldhorst, O.P.: SPCAuth: Scalable and privacy-preserving continuous authentication for web applications. In: Proc. 46th IEEE Int. Conf. on Local Computer Networks (LCN). pp. 281–286 (2021)
30. Monschein, D., Waldhorst, O.P.: mPSAuth: Privacy-preserving and scalable authentication for mobile web applications (2022), `https://arxiv.org/abs/2210.04777`
31. Monschein, D., Waldhorst, O.P.: Optimizing privacy-preserving continuous authentication of mobile devices. In: Proc. 18th Int. Conf. on Network and System Security (NSS). pp. 63–81 (2024)
32. Obla, S., Gong, X., Aloufi, A., Hu, P., Takabi, D.: Effective activation functions for homomorphic evaluation of deep neural networks. IEEE Access **8**, 153098–153112 (2020)
33. Papamichail, M.D., Chatzidimitriou, K.C., Karanikiotis, T., Oikonomou, N.C.I., Symeonidis, A.L., Saripalle, S.K.: BrainRun: A behavioral biometrics dataset towards continuous implicit authentication. MDPI Data **4**(2), 1–17 (2019)
34. Sah, C.P., Kaur, M., Singh, G.: Efficiency of zero-knowledge proofs: A thorough review and analysis. In: Proc. 5th IEEE Int. Conf. on Public Key Infrastructure and its Applications (PKIA). pp. 1–7 (2024)
35. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: A unified embedding for face recognition and clustering. In: Proc. 28th IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2015)
36. Shahandashti, S.F., Safavi-Naini, R., Safa, N.A.: Reconciling user privacy and implicit authentication for mobile devices. Elsevier Computers & Security **53**, 215–233 (2015)
37. Shepherd, C., Akram, R.N., Markantonakis, K.: Towards trusted execution of multi-modal continuous authentication schemes. In: Proc. ACM Symp. on Applied Computing (SAC). pp. 1444–1451 (2017)
38. Sitová, Z., Šeděnka, J., Yang, Q., Peng, G., Zhou, G., Gasti, P., et al.: HMOG: New behavioral biometric features for continuous authentication of smartphone users. IEEE Trans. on Information Forensics and Security **11**(5), 877–892 (2016)
39. Stragapede, G., Vera-Rodriguez, R., Tolosana, R., Morales, A.: BehavePassDB: Public database for mobile behavioral biometrics and benchmark evaluation. Elsevier Pattern Recognition **134**, 109089 (2023)
40. Stylios, I., Kokolakis, S., Thanou, O., Chatzis, S.: Behavioral biometrics & continuous user authentication on mobile devices: A survey. Elsevier Information Fusion **66**, 76–99 (2021)
41. Sumalatha, U., Prakasha, K.K., Prabhu, S., Nayak, V.C.: Touch of privacy: A homomorphic encryption-powered deep learning framework for fingerprint authentication. IEEE Access **13**, 59057–59073 (2025)
42. Sun, Y., Upadhyaya, S.: Secure and privacy preserving data processing support for active authentication. Springer Information Systems Frontiers **17**(5), 1007–1015 (2015)
43. Vassallo, G., Van Hamme, T., Preuveneers, D., Joosen, W.: Privacy-preserving behavioral authentication on smartphones. In: Proc. 1st ACM Int. Workshop on Human-Centered Sensing, Networking, and Systems (HumanSys). pp. 1–6 (2017)

44. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., et al.: Attention is all you need. In: Proc. 31st Int. Conf. on Neural Information Processing Systems (NeurIPS). pp. 6000–6010 (2017)
45. Wu, Y., Weng, J., Wang, Z., Wei, K., Wen, J., Lai, J., et al.: Attacks and countermeasures on privacy-preserving biometric authentication schemes. IEEE Trans. on Dependable and Secure Computing **20**(2), 1744–1755 (2023)
46. Yang, W., Wang, S., Cui, H., Tang, Z., Li, Y.: A review of homomorphic encryption for privacy-preserving biometrics. MDPI Sensors **23**(7), 3566 (2023)
47. Yang, Y., Zhang, H., Fan, S., Lu, H., Zhang, M., Li, X.: Poseidon: Practical homomorphic encryption accelerator. In: Proc. 29th IEEE Int. Symp. on High-Performance Computer Architecture (HPCA). pp. 870–881 (2023)

A Formal Security Model

We first present an ideal functionality for authentication, then define the realized security properties, and present the proofs under the assumptions of Section 4 and the standard cryptographic setting with security parameter λ.

A.1 Definition of the Ideal Authentication Functionality $\mathcal{F}_{\mathsf{Auth}}$

Definition 6 (Ideal Authentication Functionality $\mathcal{F}_{\mathsf{Auth}}$). *For each client C, the functionality maintains a reference profile $R_C = \{P_1, \ldots, P_n\}$, a history H_C of previously accepted samples, and a counter $c \in \mathbb{N}$.*

Enrollment / Re-enrollment. *Upon input $(\mathsf{Enroll}, C, \{P_1, \ldots, P_n\})$ from an authenticated client, set $R_C := \{P_1, \ldots, P_n\}$, $H_C = \emptyset$ and $c := 0$.*

Authentication. *Upon input (Auth, C, B):*

1. *Compute $d := \mathsf{Verify}(R_C, B)$ with $d \in \{0, 1\}$, where $d = 0$ denotes acceptance.*
2. *Output d to the server and the client.*
3. *If $d = 1$ (reject), set $c := 0$, and $H_C := \emptyset$.*
4. *If $d = 0$ (accept), set $c := c + 1$ and append B to H_C.*
5. *If $c \geq R_{count}$, update $R_C := \mathsf{UpdProfile}(R_C, H_C)$, reset $c := 0$ and $H_C := \emptyset$.*

The ML-based analysis is modeled by $\mathsf{Verify}(\cdot)$; thus, the decision bit output by $\mathcal{F}_{\mathsf{Auth}}$ corresponds to the classifier output in the real protocol.

Leakage semantics. *The server learns only the decision bit d, and no additional information is revealed to the server and the client.*

In PPMLAuth, the state (R_C, B) is stored and processed with HE. The profile-update step corresponds to the logical effect of a key rollover, which couples behavioral profile adaptation with cryptographic key renewal. The timing of key rollover and re-enrollment is a protocol-level mechanism used to bound key exposure and is analyzed separately in the realization proofs.

A.2 Formal Security Properties

Definition 7 (Classifier Decision Integrity). *A protocol satisfies* Classifier Decision Integrity *if, for every PPT malicious client adversary, the decision bit accepted by the server equals d output by $\mathcal{F}_{\mathsf{Auth}}$, except with negligible probability.*

Definition 8 (Key-Exposure Boundedness). *Let $K_1, K_2, \ldots$ denote the successive key epochs of the protocol, where each epoch K_i contains all cryptographic key material active simultaneously. For an epoch K_i, let $N_{\mathsf{pairs}}(K_i)$ be the total number of ciphertext-plaintext pairs observable by the server under keys in K_i before switching to K_{i+1}. The protocol satisfies* Key-Exposure Boundedness *if there exists a bound $N_{\max}$ such that $N_{\mathsf{pairs}}(K_i) \leq N_{\max}$ for all i.*

Definition 9 (Profile Update Security). *Let T denote the number of authentication attempts within a rollover window. A protocol satisfies* Profile Update Security *if, for every PPT malicious client adversary $\mathcal{A}_{\mathsf{cli}}$, the probability it causes a profile update based on adversary-submitted history is at most $T * p^{R\mathsf{count}}$ in a rollover window, where p upper-bounds per-attempt acceptance probability.*

Inherent Confidentiality Limitations and Proofs. In the considered authentication setting, a certain minimum amount of information leakage to the server is unavoidable, regardless of the protocol used. This motivates a restricted confidentiality notion and a correspondingly adjusted ideal functionality.

Lemma 1 (Server-Side Leakage Bound). *Consider any client–server authentication protocol in which the server provides a ciphertext that is decrypted by the client to a single-bit value and returned to the server. Assume further that the client cannot independently recompute $d = \mathsf{Verify}(R_C, B)$ from its local information (because R_C, the classifier procedure, or their correct evaluation is not verifiable from the client's local information). Then a malicious server can force disclosure of at least one bit of information per authentication attempt.*

Proof. Because the client cannot recompute d from its local information, a malicious server may replace the ciphertext with an encryption of a bit $\beta \in \{0, 1\}$, which the client cannot distinguish from an encryption of the true decision bit.

Restricted Ideal Functionality for Malicious Servers. Since Lemma 1 shows that confidentiality against a malicious server cannot, in general, restrict leakage to d, we define a restricted ideal functionality $\mathcal{F}_{\mathsf{Auth}}^{\mathsf{res}}$ that reveals a single bit, which is equal to d for an honest server and adversarially chosen otherwise.

Definition 10 (Restricted Authentication Functionality). *The restricted authentication functionality $\mathcal{F}_{\mathsf{Auth}}^{\mathsf{res}}$ behaves identically to $\mathcal{F}_{\mathsf{Auth}}$, except that on each authentication attempt it outputs a bit $b \in \{0, 1\}$ to the server, where $b = d$ for an honest server and b may be chosen arbitrarily by a malicious server.*

Restricted Confidentiality Formal Definitions. We distinguish server-side and client-side restricted confidentiality to reflect the asymmetric leakage.

Definition 11 (Client-Side Restricted Confidentiality). *A protocol satisfies* client-side restricted confidentiality *if, for every PPT malicious client adversary $\mathcal{A}_{\mathsf{cli}}$, there exists a PPT simulator Sim such that the client's view in the real execution is computationally indistinguishable from its view in the ideal execution with $\mathcal{F}_{\mathsf{Auth}}$ (i.e., the client learns only the decision bit d).*

Definition 12 (Server-Side Restricted Confidentiality). *A protocol satisfies* server-side restricted confidentiality *if, for every PPT malicious server adversary $\mathcal{A}_{\mathsf{srv}}$, there exists a PPT simulator* Sim *such that the server's view in the real execution is computationally indistinguishable from its view in the ideal execution with $\mathcal{F}_{\mathsf{Auth}}^{\mathsf{res}}$.*

A.3 Realization of the Security Properties

We demonstrate that PPMLAuth realizes the security properties by establishing the corresponding lemmas for the real protocol. For correctness-style properties (e.g., integrity), we show output equivalence to the ideal functionality, while confidentiality properties are established via simulation-based arguments.

Lemma 2 (Classifier Decision Integrity from Secure Disclosure). *Assume commitments com_{sk} are computed using a collision-resistant hash function and that the zk-SNARK circuit used in PPMLAuth is sound. Then, for every PPT malicious client adversary, the probability that the server accepts a decision bit that differs from the classifier output computed by $\mathcal{F}_{\mathsf{Auth}}$ is negligible. Hence, PPMLAuth realizes Classifier Decision Integrity (Definition 7).*

Proof. The server sends a ciphertext ct_b encrypting the classifier output $d \in 0, 1$. The client returns a bit b_c with a zk-SNARK proof that $b_c = \mathsf{Dec}(ct_b, sk)$, for a secret key sk whose hash $\mathsf{Poseidon}(sk)$ equals the commitment com_{sk}.

By collision resistance of Poseidon [19], for any fixed commitment value com_{sk} there exists a unique secret key $sk^\star$ with $\mathsf{Poseidon}(sk^\star) = \mathsf{com}_{sk}$, except with negligible probability. Since the zk-SNARK circuit enforces consistency with this commitment and is sound, any accepted proof must involve $sk^\star$ and satisfy $b_c = \mathsf{Dec}(ct_b, sk^\star)$. As ct_b encrypts the classifier output, the accepted bit equals the decision computed by $\mathcal{F}_{\mathsf{Auth}}$, except with negligible probability. Therefore, PPMLAuth satisfies Classifier Decision Integrity (Definition 7).

Lemma 3 (Key-Exposure Boundedness from Rollover Strategy). *The key rollover strategy of PPMLAuth (see Section 3.5) bounds, by a constant $N_{\max}$, the number of ciphertext-plaintext pairs observable by the server in any key epoch K_i, thus satisfying Key-Exposure Boundedness (Definition 8).*

Proof. In any key epoch K_i, the server observes one ciphertext-plaintext pair per authentication attempt, namely the TFHE ciphertext encoding the classifier output and its decryption. By design, the client enforces epoch termination after at most R_{max} authentication attempts independently of server behavior, and rejects reuse of ciphertexts within the same epoch before destroying key material.

Therefore, the number of observable ciphertext–plaintext pairs per epoch is bounded by R_{max}. Setting $N_{\max} := R_{\mathsf{max}}$ yields $N_{\mathsf{pairs}}(K_i) \leq N_{\max}$ for all i, and PPMLAuth satisfies Key-Exposure Boundedness (Definition 8).

Lemma 4 (Profile Update Security from Rollover Design). *Assume that (re-)enrollment is secure against impersonation and that any PPT malicious*

client is accepted in a single authentication attempt with probability at most p. In a rollover window of length $T := R_{\max} - R_{\mathsf{start}} + 1$, the probability that a malicious client causes a profile update based on an adversary-submitted history is at most $T * p^{R_{\mathsf{count}}}$. Hence, PPMLAuth satisfies Profile Update Security (Definition 9).

Proof. In PPMLAuth, the reference profile is updated either (a) after $r = R_{count}$ consecutive successful authentications during a rollover, or (b) via authenticated (re-)enrollment. By assumption, (b) cannot be completed by a malicious client.

The rollover window is bounded by $T := R_{\max} - R_{\mathsf{start}} + 1$ authentication attempts, enforced by the server independently of client behavior. Further, a rollover-based profile update using an adversary-submitted history can occur only if the malicious client produces a consecutive block of r accepted authentication attempts within the window, each contributing an adversarial sample to the history H_C. For a fixed i, let $E_{i,k}$ be the event that the $(i + k)$-th attempt is accepted, for $k = 0, \ldots, r - 1$. By the per-attempt bound, for all k, $\Pr[E_{i,k} \mid E_{i,0} \wedge \cdots \wedge E_{i,k-1}] \leq p, \implies \Pr\left[\bigwedge_{k=0}^{r-1} E_{i,k}\right] \leq p^r$.

There are at most $T-r+1$ choices for i, so the probability that any such block occurs is at most $(T - r + 1)p^r \leq Tp^r$. Thus, the probability that a malicious client causes a profile update with an adversary-submitted history is at most $T\,p^r$, and PPMLAuth satisfies Profile Update Security (Definition 9).

Lemma 5 (Client-Side Restricted Confidentiality from Design). *For every PPT malicious client adversary $\mathcal{A}_{\mathsf{cli}}$, there exists a PPT simulator Sim such that the client's view in the real execution of PPMLAuth is computationally indistinguishable from its view in the ideal execution with $\mathcal{F}_{\mathsf{Auth}}$.*

Proof. The only server-generated value received by the client is a TFHE ciphertext encrypting the decision bit, which is indistinguishable from an encryption of any other bit prior to decryption by IND-CPA security of TFHE [13]. Therefore, the client's view in the real execution can be simulated given only the decision bit d, which is exactly the information provided to the client by $\mathcal{F}_{\mathsf{Auth}}$. Thus, Client-Side Restricted Confidentiality (Definition 11) holds.

Lemma 6 (Server-Side Restricted Confidentiality from HE). *Under the IND-CPA security of the employed HE schemes, for every PPT malicious server adversary $\mathcal{A}_{\mathsf{srv}}$, there exists a PPT simulator Sim such that the server's view in the real execution of PPMLAuth is computationally indistinguishable from its view in the ideal execution with $\mathcal{F}^{\mathsf{res}}_{\mathsf{Auth}}$.*

Proof. The server stores the reference profile R_C and history H_C encrypted under CKKS without access to the secret key. By IND-CPA security [12], its view is computationally indistinguishable for any two (R_C, H_C) of equal length.

During secure disclosure, the server receives a single bit returned by the client. By the zero-knowledge property of the zk-SNARK, the accompanying proof reveals no additional information beyond correctness of the claimed decryption. Thus, in each authentication attempt, the server learns exactly one bit and the real execution can be simulated using $\mathcal{F}^{\mathsf{res}}_{\mathsf{Auth}}$ in light of Lemma 1. Hence, Server-Side Restricted Confidentiality (Definition 12) holds.

Theorem 2 (Security of PPMLAuth). *Under the IND-CPA security of the employed HE schemes, the soundness and zero-knowledge properties of the zk-SNARK, and the enforced key rollover policy, PPMLAuth satisfies Classifier Decision Integrity, Key-Exposure Boundedness, Profile Update Security, and Client-Side and Server-Side Restricted Confidentiality as defined in Appendix A.2.*

Proof. Classifier Decision Integrity follows from Lemma 2, Key-Exposure Boundedness from Lemma 3, Profile Update Security from Lemma 4, and Restricted Confidentiality from Lemma 6 (server-side) and Lemma 5 (client-side).

B Distance-Based vs. ML-Based Approaches

Current privacy-preserving authentication systems for behavioral authentication largely rely on Euclidean, Manhattan, and Cosine distance to compare embeddings [1]. This is reasonable, as ML models used to derive embeddings are typically trained with loss functions that explicitly enforce small intra-user and large inter-user distances [35]. In conclusion, an ML model for comparing two embeddings cannot leverage more complex patterns to achieve higher effectiveness.

Nevertheless, we observe three common real-world scenarios in which significant differences between distance-based and ML-based behavioral analysis arise. To this end, we compare our neural network-based approach to analyzing embeddings with linear combinations of distances. The linear combinations were calculated as follows: $a_1 * d_1 + a_2 * d_2 + \cdots + a_n * d_n + b$, where d_i represents one of the pairwise distances between all available embeddings, and a_i as well as b are learnable parameters of the linear model. For the ML-based analysis, we concatenate all embeddings and use them as a composite input. To ensure comparability and to isolate the effect of the analysis method, both variants are trained and evaluated on unencrypted data using the procedure described in Section 5.2. For BehavePassDB, results are obtained on the validation set, as the test set is not publicly accessible and the evaluation platform restricts available metrics [39]. In this case, no hyperparameter tuning is performed to avoid bias.

B.1 Data Source Fusion

Mobile devices expose multiple sources (e.g., touch dynamics, motion sensors, and physical location), each with distinct characteristics and availability. During authentication, these signals are usually fused by applying a weighted sum to the per-source distances [1,39]. While computationally efficient, this approach fails to capture complex interdependencies between source embeddings. Accordingly, Table 5 compares the authentication effectiveness of our ML-based fusion to linear combinations of distance metrics used in existing approaches.

The ML-based classifier consistently outperforms all distance metrics on both datasets, with gains of up to 8.56 percentage points in AUC and 7.17 percentage points in EER. These improvements translate into substantially better accuracy, FAR, and FRR at fixed thresholds, indicating reliable generalization to unseen users. Overall, the results suggest that the ML-based classifier is superior due to its ability to learn richer patterns and interdependencies across data sources.

Table 5: Authentication effectiveness comparison of ML-based and distance-based classifiers when considering fusion of multiple data sources

BehavePassDB (6 data sources)

Classifier	EER%	AUC%	ACC%	FAR%	FRR%
ML-based	*25.95*	*83.67*	*75.47*	*26.99*	*25.52*
Euclidean	33.12	76.04	64.91	10.50	59.54
Manhattan	32.60	75.11	66.93	35.59	30.56
Cosine	28.04	80.52	66.54	51.33	15.54

BrainRun (5 data sources)

Classifier	EER%	AUC%	ACC%	FAR%	FRR%
ML-based	*15.65*	*92.33*	*84.54*	*11.39*	*21.57*
Euclidean	17.82	90.08	80.57	12.69	26.14
Manhattan	17.68	90.14	80.62	12.87	25.87
Cosine	18.55	89.78	79.80	12.77	27.61

B.2 Authentication Horizon

Over time, the system can accumulate reference embeddings that can be used to improve authentication effectiveness. However, this poses the challenge of effectively leveraging the additional information. Again, we compared the ML-based classifier to distance metrics using the accelerometer as a single data source while varying the number of reference embeddings. Moreover, we restrict the evaluation to BrainRun, as BehavePassDB provides insufficient sessions per user.

Table 5 shows the results, depending on the number of sessions used as a reference. We omit the results for a single reference session, as we did not observe any significant performance difference. This is expected, as the ML-based

Table 6: Comparison of ML-based and distance-based classifiers when considering multiple reference sessions for accelerometer data of the BrainRun dataset

2 Reference Embeddings

Classifier	EER%	AUC%	ACC%	FAR%	FRR%
ML-based	*27.09*	*80.35*	*71.73*	*23.79*	*32.77*
Euclidean	28.54	77.02	69.60	22.54	38.20
Manhattan	28.5	77.07	69.99	22.53	37.48
Cosine	28.96	76.49	69.61	22.31	38.44

3 Reference Embeddings

Classifier	EER%	AUC%	ACC%	FAR%	FRR%
ML-based	*25.05*	*83.67*	*75.20*	*23.92*	*29.7*
Euclidean	28.09	78.10	71.04	23.98	33.97
Manhattan	27.80	78.41	70.91	23.70	34.45
Cosine	28.74	77.31	69.67	23.86	36.83

classifier cannot exploit information beyond the embeddings being compared. With two or more reference sessions, ML-based analysis consistently outperforms distance-based methods, with the gap increasing as more reference data becomes available. In summary, ML-based analysis gains an advantage by exploiting temporal structure that linear distance aggregation cannot capture. Importantly, we observed the same trend across all other data sources.

B.3 Attack Robustness

In a client–server setting, behavioral authentication must remain *robust* against malicious clients with knowledge of system internals (e.g., ML models used). To assess the impact of the analysis method on robustness, we model attackers as neural networks (*adversarial models*), as shown in Figure 2.

The adversarial models receive a configurable fraction of the enrollment sessions as input, ranging from 0 % (no access) to 100 % (full access), representing

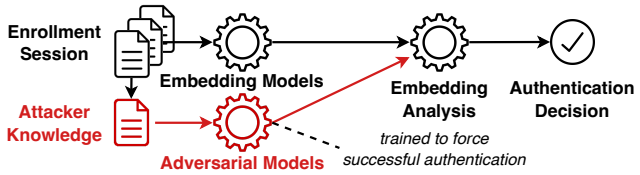

Fig. 2: Experiment setup for testing the robustness of the behavioral analysis

the attacker's knowledge of the legitimate user's behavior. They generate one adversarial embedding per data source, which is evaluated alongside the enrollment embeddings by the behavioral analysis strategy used. During training, we freeze the behavioral analysis while the adversarial models learn to generate embeddings that bypass the authentication. Furthermore, the adversarial models share the same architecture as the preprocessing models (see Section 3.2). Finally, we evaluate how often the adversarial models achieve successful authentication to quantify the effectiveness in detecting advanced attackers.

Figure 3 shows the probability of successful attacks (y-axis) as a function of the attacker's access to enrollment data (x-axis).

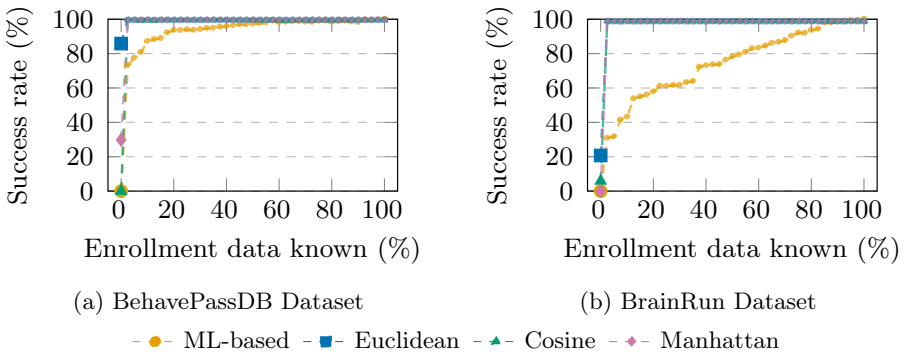

Fig. 3: Impact of enrollment data exposure on attack success probability

Across both datasets, the probability of an attacker bypassing authentication is higher for distance-based methods. In particular, distance metrics can be bypassed with non-negligible probability with no knowledge of the enrollment data. As the attacker's knowledge increases, the success probability instantly peaks at 100 % for distance-based methods, while growing significantly slower for the ML-based classifier. The dataset also influences system vulnerability, suggesting that larger datasets can enhance robustness. However, all strategies exhibit high success rates once the adversary gains access to the legitimate user's behavioral data. Consequently, broader research efforts are needed to address adversaries with behavioral knowledge and access to the underlying ML models.