LuTBA: Lightweight and Updatable Touch-Based Authentication Scheme for Mobile Devices

Jingyu Yao and Ding Wang

***Abstract*—Touch-based authentication is a promising way to enhance mobile device security in a non-intrusive manner. It can run in the background on mobile devices to provide continuous authentication. However, deploying existing solutions in practice faces two limitations. First, most of them are not robust against temporal drift in user behavior patterns. Second, the enrollment phase often requires a large number of samples for model training, which incurs additional time and resource overhead on the user side and undermines usability. In this paper, through a review of model architectures and pipelines across 44 relevant works, we reveal that these inherent limitations are rooted in the local-training paradigm. Accordingly, we propose LuTBA, a Lightweight and updatable Touch-Based Authentication scheme based on multi-modal contrastive learning. LuTBA breaks the common paradigm and identifies users by measuring the similarity between inputs and templates directly, rather than training user-specific models during the enrollment phase. We comprehensively evaluate LuTBA using two real-world datasets. Experimental results show that LuTBA achieves optimal performance with 10 enrolled samples (1/80 of the training samples of baselines), with the mean equal error rate reduced by 2.08%-24.83%. Moreover, with the dynamic update mechanism, LuTBA shows robust adaptability to behavior drift. Particularly, we build a new dataset to further evaluate LuTBA's resistance to imitation attacks. With a window-size setting of 10, LuTBA attains an average equal error rate of 8.2%. All relevant codes and datasets have been made publicly available.**

***Index Terms*—Touch-based authentication, mobile devices, contrastive learning, dynamic update.**

I. Introduction

With the popularization of smartphones, global mobile network subscriptions have surged from 3.6 billion in 2016 to 7.3 billion in 2025, and are forecast to exceed 7.9 billion by 2028 [1]. Alongside this rapid growth, smartphones have gradually become the mainstream platforms for services requiring high security such as financial transactions and private information storage. Therefore, robust access control is necessary.

Smartphones mainly employ two types of access control methods. One is knowledge-based authentication, such as personal identification numbers (PINs) [2], [3] and pattern unlocks [4], [5]. The other is biometric-based authentication, such as fingerprint recognition [6], [7] and facial recognition [8], [9]. Given the potential vulnerabilities of biometric authentication [10], knowledge-based authentication remains necessary and is still a primary authentication method. However, knowledge-based authentication is susceptible to various attacks, including online guessing attacks [10], [11], [12] and shoulder surfing attacks [13], [14]. Taking PINs as an example, Aviv et al. [13] reported that shoulder surfing attacks can achieve a success rate of 43.8% after just one observation. In addition, Wang et al. [12] demonstrated that the success rate of guessing PINs ranges from 2.9% to 5.7% within 10 attempts. These attacks continuously pose a serious threat to users' financial security and personal privacy.

To enhance smartphone security, touch-based authentication (TBA) [15], [16], [17] has gained widespread attention. Utilizing signals from the smartphone's built-in sensors (e.g., pressure sensor, gyroscope, accelerometer) during user-device interaction, TBA can capture users' unique behavioral patterns to verify their identities. Compared to explicit authentication methods [2], [3], TBA can continuously verify the identity of users in the background without interrupting them.

A. Motivations

In general, existing TBA schemes [18], [19], [20] collect their touch data when users complete designated tasks or engage with applications such as Douyin, WeChat, YouTube, PayPal, and Alipay. Upon gathering enough data, a user-specific authentication model is either trained locally or in the cloud after data upload [21]. Several studies [18], [22] have demonstrated that user behavioral patterns exhibit subtle variations across different application scenarios, and fine-grained contextual awareness can improve model performance in specific scenarios. However, some applications (e.g., PayPal and Alipay) may be accessed far less frequently and for shorter durations than services like Douyin and YouTube. Additionally, various contextual factors and users' behavioral states further increase the demand for data quantity. Under these constraints, it may be difficult to collect thousands or even tens of thousands of enrollment samples within a short period. Given that the volume of enrollment samples is critical to the efficacy of model training [22], this limitation raises concerns about the practical viability of existing schemes.

Furthermore, user behavioral patterns undergo gradual drifts over time [23], [24], [25]. Such drifts reduce the similarity between the patterns captured during usage and those

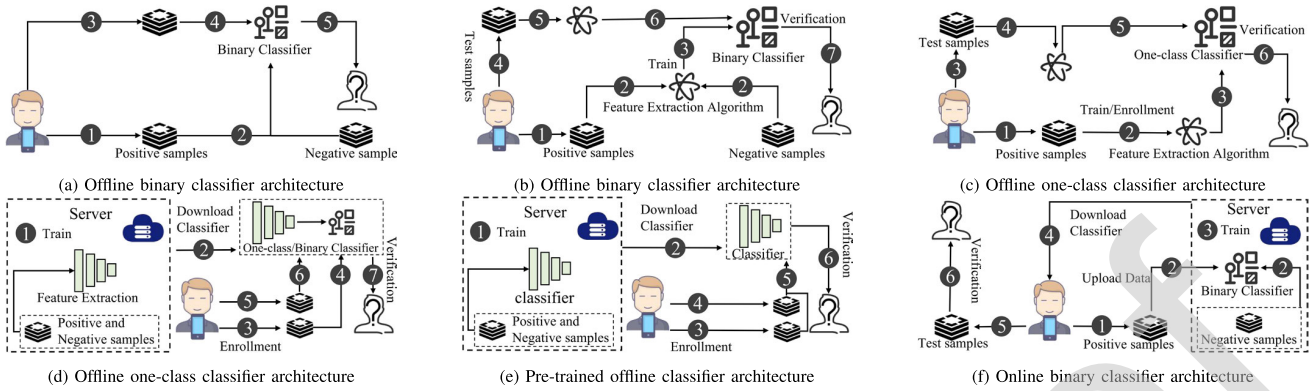
Received 23 January 2026; revised 22 August 2026; accepted 4 October 2026. This work was supported in part by Tianjin Natural Science Foundation Project under Grant 25JCJQJC00230 and in part by the National Natural Science Foundation of China under Grant 62572259. The associate editor coordinating the review of this article and approving it for publication was Prof. Muhammad Khurram Khan. *(Corresponding author: Ding Wang.)*

The authors are with the College of Cryptology and Cyber Science and Tianjin Key Laboratory of Network and Data Security Technology, Nankai University, Tianjin 300350, China (e-mail: jingyuyao@mail.nankai.edu.cn; wangding@nankai.edu.cn).

Digital Object Identifier 10.1109/TIFS.2026.3744613

1556-6021 © 2026 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

IEEE Proof



(a) Offline binary classifier architecture (b) Offline binary classifier architecture (c) Offline one-class classifier architecture

(d) Offline one-class classifier architecture (e) Pre-trained offline classifier architecture (f) Online binary classifier architecture

Fig. 1. The figure summarizes six architectures derived from existing solutions. Among them, the pre-trained offline classifier architecture shown in Fig. 1e does not require samples from registered users for model training, whereas all the other architectures rely on training samples provided by registered users.

enrolled initially, thereby increasing authentication error rates. Shen et al. [24] attempted to address this issue by retraining the authentication model with data from adjacent time windows. However, this approach has two major drawbacks. First, frequent retraining imposes additional computational overhead on the user side. Second, for infrequently used applications, the data collection becomes more challenging, ultimately undermining the effectiveness of retraining-based solutions.

B. Our Study

In this work, we conduct a comprehensive survey of TBA schemes in the literature, trying to explore the root causes of the above challenges. Through a systematic analysis, we identify six distinct architectural paradigms adopted in the literature, as illustrated in Fig. 1. The workflow of each architecture is elaborated in Section III. *We observe that most existing research endeavors resort to binary classification or one-class classification to train user-specific authentication models, a practice that inherently requires a large number of enrollment samples per target user.*

By contrast, contrastive learning, a technique that learns discriminative representations by contrasting positive and negative pairs, has been widely adopted in domains such as gait recognition [26], [27] and face recognition [8], [28]. Unlike conventional TBA models that rely on target users' own data for training, contrastive learning-based frameworks leverage auxiliary datasets from other users during pre-training, reducing the dependence on enrollment samples. Building on this technique, we propose LuTBA, a new authentication scheme that requires only a few enrollment samples per user and can dynamically update its reference template set using historical data to adapt to temporal drifts in behavioral patterns. To validate its effectiveness, we evaluate LuTBA on two large-scale real-world datasets. Furthermore, we evaluate the security of LuTBA against random-vector attack, population attack, and imitation attack, and collect a new dataset for the imitation attack evaluation.

C. Contributions

The contributions of this paper are as follows:

- We systematically categorize existing schemes into six architectures and analyze their workflows. Our analysis reveals that binary-classification and one-class-classification architectures widely adopted in prior work inherently demand substantial enrollment samples, a major factor limiting their practicability.
- We propose LuTBA, a **L**ightweight and **u**pdatable **T**ouch-**B**ased **A**uthentication scheme. Compared with diverse baseline models across different architectures, LuTBA achieves the best performance with fewer enrollment samples. Moreover, LuTBA demonstrates robustness against temporal drift in user behavioral patterns via a lightweight update mechanism.
- We construct a new public dataset for imitation attack and evaluate LuTBA's resilience against this attack type. Experimental results show that LuTBA can effectively resist imitation attacks as the authentication window increases. Furthermore, the collected dataset may serve as a useful resource for future research.

II. Related Works

In this section, we review prior works that are relevant to our study. Table I summarizes the details of 44 related works in the literature. In the following, we organize our discussion from two perspectives: classifiers and the effect of time.

A. Classifiers

In Table I, VI studies [15], [35], [36], [48], [56], [59] adopt one-class classification schemes, such as 1NN-DTW [56], OCSVM [35], [48], DeSVDD [36]. *The assumption behind them is that adversarial data or their distribution cannot be obtained in advance*. In contrast, 41 studies employ binary-classifier schemes, with the core idea that *sample data from other users can be pre-collected for model training*. Such pre-acquired third-party user data can help the model converge its decision boundary, thereby improving the accuracy of the authentication model. Studies [15], [35] have demonstrated that binary classifiers outperform one-class classifiers significantly in terms of performance.

From the perspective of model deployment overhead, 41 studies require training a dedicated model for each user, which in turn demands a large number of enrollment samples to support model training. In contrast, 3 studies [16], [56], [59] require users to provide a small number of enrollment samples

IEEE Proof

TABLE I

RELATED WORK ON TOUCH-BASED AUTHENTICATION SCHEMES

Authors	Year	Ref	Environment	Users	Signals		Feature Engineering	Classifiers	Metrics	Data	Codes
					S_1	S_2					
Kiran et al.	2025	[15]	Lab	30	●	○	Artificial	SVC-RBF,SVC-Linear, RF, XGBoost	FRR: 2.54% FAR: 82.93%	○	○
Qiao et al.	2025	[16]	-	29	●	○	Automated	Transformer + CNN	Accuracy: 97.69%-99.08%	○	○
Wang et al.	2025	[17]	Lab, Wild	-	●	○	Automated	Multi-Head Self-Attention + CNN + SVM	EER: 1.12%, 2.42%	●	○
Kudo et al.	2025	[19]	Lab	105	●	○	Artificial	SVM, RF, NN	EER: 11.01%-22.74%	●	○
Mahfouz et al.	2024	[29]	Wild	52	●	●	Artificial	RF	EER: 0.84%	○	○
Kudo et al.	2024	[30]	Lab	41, 71	●	○	Artificial	KNN, SVM, RF, MLP	EER: 8.47%	●	○
Delgado-Santos et al.	2024	[21]	Wild	464	●	●	Automated	Transformer	EER: 3.6%-11.0%	○	●
Wu et al.	2024	[25]	Lab	20	●	○	Artificial	SVM, RF	EER: 1.25%	◐	●
Mahfouz et al.	2024	[31]	Wild	60	●	○	Artificial	MLP	FRR: 33% FAR: 13% EER: 38%	◐	○
Shen et al.	2024	[20]	Wild	100	●	○	Automated	3D CNN + RNN + FC	FRR: 8.5% FAR: 7.9% EER: 8.2%	○	○
Zhao et al.	2023	[18]	Wild	100	●	●	Artificial	CNN+LSTM	FRR: 2.7% FAR: 2.4% EER: 2.5%	○	○
Shen et al.	2023	[24]	Wild	100	●	●	Artificial	GBDT+MLP	FRR: 8.77% FAR: 9.59% EER: 7.28%	○	○
Aaby et al.	2023	[32]	Lab	138	●	○	Artificial	KNN, SVM, RF, ET, GB	EER: 24.7%	◐	○
Stylios et al.	2023	[33]	Lab	39	●	○	Automated	MLP, LSTM	FRR: 0.6% FAR: 2.6% EER: 1%	○	○
Chao et al.	2023	[34]	Lab	36	●	○	Artificial	SVM, RF	EER: 7.9%-20.3%	◐	○
Ray-Dowling et al.	2022	[35]	Lab	100, 117	●	●	Artificial	Binary SVM, OC-SVM	EER: 0.2%-8.8%	◐	○
Shen et al.	2022	[36]	Lab, Wild	100	●	●	Artificial	DeSVDD	EER: 14.9%	○	○
Agrawal et al.	2022	[37]	Lab, Wild	-	●	○	Artificial	SVM, RF, MLP	FRR: 3.60% FAR: 8.00%	○	●
DeRidder et al.	2022	[38]	Lab	25	●	○	Artificial	RF, KNN	Accuracy: 93.39%-93.50%	●	○
Zaidi et al.	2022	[39]	Lab, Wild	-	●	○	Artificial	KNN, SVM, DT, NB, LR, MLP	EER: 0.53%	●	○
Georgiev et al.	2022	[40]	Wild	470	●	○	Artificial	SVM,RF,MLP,KNN	EER: 8.4%	●	●
Incel et al.	2021	[41]	Wild	45	●	●	Artificial	SVM,KNN,MLP,DT,RF,NB	FRR: 1.66% FAR: 3.37% EER: 4.03%	◐	○
Ouadjer et al.	2020	[42]	Lab	41	●	○	Artificial	XGBoost	Accuracy: 99.23%	●	○
Li et al.	2021	[43]	Wild	60	●	○	Artificial	J48 ,Naive Bayes, RBFN, BPNN, SVM	FRR: 3.12% FAR: 2.45%	○	●
Cheng et al.	2020	[44]	Wild	100	●	●	Artificial	kNN,SVM,RF,ET	EER: 3.4%	○	○
Abuhamad et al.	2020	[45]	Wild	84	●	●	Automated	LSTM	FRR: 6.67% FAR: 0.95% EER: 0.41%	○	○
Buriro et al.	2019	[46]	Wild	85	●	●	Artificial	BN,NB,SVM,KNN,J48,RF	Accuracy: 98.56%	○	○
Meng et al.	2019	[47]	Wild	50	●	○	Artificial	J48, BN, RBFN, BPNN, SVM	FRR: 3.42% FAR: 2.89%	○	●
Yang et al.	2019	[48]	Wild	45	●	○	Artificial	OCSVM, iForest	Accuracy: 95.85%	○	○
Syed et al.	2019	[49]	Lab	31	●	○	Artificial	SVM, LG, NB, RF, MLP	EER: 5.4%	◐	○
Fierrez et al.	2018	[50]	Lab, Wild	-	●	○	Artificial	SVM,GMM, SVM+GMM	EER: ∼10.0%	●	◐
Wang et al.	2017	[51]	Wild	20	●	○	Artificial	SVM, RF	AUC: 80%-96%	○	○
Palaskar et al.	2016	[52]	Lab	31	●	○	Artificial	RF, CART, NB, SVM, KNN, LP	EER: 10.4%	○	○
Mahbub et al.	2016	[53]	Wild	48	●	○	Artificial	KNN,SVM, NB, LR, RT+LR, RF, GBM	EER: 22.1%	◐	○
Shen et al.	2015	[22]	Wild	71	●	○	Artificial	KNN, SVM, MLP, RF	FRR: 0.1% FAR: 18.52% EER: 25%	○	○
Temper et al.	2015	[54]	Lab	22	●	○	Artificial	VQNN	EER: 11.5%	○	○
Antal et al.	2015	[55]	Lab	71	●	○	Artificial	KNN, RF, SVM	Accuracy: 88%	●	○
Feng et al.	2014	[56]	Wild	123	●	○	Artificial	1NN-DTW	FRR: ∼9% FAR: ∼7%	○	○
Xu et al.	2014	[57]	Lab	32	●	○	Artificial	SVM, LR, NB	FRR: ∼ 6.42% FAR: ∼ 6.42%	◐	○
Serwadda et al.	2013	[58]	Lab	190	●	○	Artificial	SVM, NB, RF, KNN, BN, MLP et al.	EER: 10.5%-17.2%	◐	○
Zhao et al.	2013	[59]	Lab	30	●	○	Automated	NC,L_1, L_2	EER: 11.28%-17.29%	○	○
Li et al.	2013	[60]	Wild	75	●	○	Artificial	SVM	Accuracy: 79.74%-95.75%	○	●
Bo et al.	2013	[61]	Wild	100	●	●	Artificial	SVM	FRR: ∼20% FAR: 9%-23%	○	○
Frank et al.	2013	[23]	Lab	41	●	○	Artificial	KNN, SVM	EER: 13%	●	○

†: S_1 denotes touch dynamics data, including touch pressure, touch area, touch coordinates, and timestamp. S_2 denotes the data captured by motion sensors, such as gyroscopes and accelerometers. ○ indicates not adopted or unavailable. ● indicates adopted or available. ◐ indicates that the paper explicitly states the data/code is available or provides an access link, but the link is invalid or the data/code requires active application for acquisition.

IEEE Proof

TABLE II
SUMMARY OF TBA ATTACK TYPES FOR MOBILE DEVICES

Type	Attack type	Capability							Description	Cases
		A1	A2	A3	A4	A5	A6	A7		
I	Zero-effort Attack	●	○	○	○	○	○	○	The adversary has no access to any information about the user and operates the mobile device with their own behavioral patterns.	[15]–[25], [29]–[61]
II	Imitation Attack	●	●	○	○	○	○	○	Adversaries may acquire partial behavioral patterns of target users via shoulder surfing, smudge attacks, etc., and try to mimic them.	[16], [25], [46], [62]
III	Population Attack	●	○	●	○	●	○	○	The attacker leverages statistical features from public datasets to generate an attack vector. The core idea is to identify the most probable sequence of feature vectors, thereby improving the success rate of the attack.	[37], [63]
IV	Robot Attack	●	○	●	○	○	○	●	The adversary trains robots using statistical data obtained from public datasets, thereby deceiving the authenticator.	[64]
V	Random-vector Attack	●	○	○	●	●	○	○	The core hypothesis is that the user's acceptance region is not compactly convergent. In other words, the acceptance regions of traditional machine learning authentication models are larger than the feature distribution regions of target users.	[37], [65]
VI	Snoop-forge-replay Attack	●	○	○	○	●	●	○	Using obtained user behavioral pattern samples, the adversary creates more forged samples and injects them into the authentication pipeline for spoofing the authenticator.	[37]
VII	Robot Mimicry Attack	●	○	○	○	○	●	●	The adversary trains robots using the acquired user behavioral pattern samples to imitate the target user's behavioral patterns, thereby deceiving the authenticator.	[37]

without the need for model training. Specifically, Feng et al. [56] and Zhao et al. [59] adopt handcrafted feature extraction with a template matching approach, rather than training authentication models. Qiao et al. [16], in contrast, construct a feature extractor using Transformer and CNN architectures, which can extract more discriminative and generalizable features from unseen user samples. To date, however, prior research lacks a systematic discussion and comparison of classifier types. To address this gap, we systematically classify the existing classifiers into six architectural categories in Section III and subsequently conduct a comparative experimental analysis of their performance. Based on the results, we argue that collecting sufficient samples for the target user during the enrollment phase to train a local model is subject to notable limitations in feasibility and resource overhead. Therefore, we suggest using the architecture illustrated in Fig. 1e, which does not depend on the target user, thereby achieving an effective trade-off between performance and efficiency.

B. Effect of Time

Several studies [23], [24], [25], [47], [50], [52], [56], [57] have demonstrated that users' behavioral patterns undergo subtle drifts over time, which consequently degrades the accuracy of authentication systems. To mitigate the adverse effects caused by such temporal behavioral drifts, existing solutions typically involve dynamically updating template libraries [56] or retraining models [24] with newly collected data. However, retraining models incurs additional resource overhead. Although the scheme proposed by Feng et al. [56] adapts to behavioral changes by updating templates, it relies on manually crafted features that are susceptible to environmental noise and variations in user posture, thereby compromising feature effectiveness. To address this issue, we leverage deep learning to extract user-independent features from large-scale real-world datasets. This approach not only effectively mitigates the impact of posture changes and environmental noise, thus enhancing feature extraction effectiveness, but also, owing to its reliance on feature-vector matching for user verification, enables straightforward adaptation to behavioral drift through template updates, eliminating the need for model retraining.

III. UNDERSTANDING TBA WORKFLOWS

In this section, we attempt to sort out and systematize the technical workflow of TBA. To the best of our knowledge, *we are the first to do this*. For this, we analyze and summarize the architectures of schemes proposed by 44 relevant works. As illustrated in Fig. 1, the architectures can be categorized into six types, among which five require users to provide samples for training of classifiers. In the following, we elaborate on each of these six architectures.

Fig. 1a presents a specific instantiation of the offline binary classifier architecture, where the classifier takes preprocessed raw data as input and performs both feature extraction and identity verification. During the interaction between a user and the smartphone, the data captured by built-in sensors of the device contains unique behavioral pattern characteristics of this user ❶. In the enrollment phase, the data is used along with pre-stored data of other users to train binary classification models, such as Random Forest and Support Vector Machine ❷. In the authentication phase, real-time behavioral pattern data collected by the smartphone ❸ is input into the trained binary classifier ❹, and the output result of the classifier is 0 or 1, where 0 represents unauthorized access and 1 represents legitimate access ❺.

Unlike the architecture illustrated in Fig. 1a, the architecture in Fig. 1b additionally incorporates a feature extraction algorithm ❷ ❺. The core function of this algorithm is to extract feature dimensions that are highly correlated with users' behavioral patterns from the raw sensor data in advance, thereby filtering out noise interference in the data and further improving the accuracy of identity authentication.

Different from the architecture in Fig. 1b, the architecture in Fig. 1c adopts a one-class classifier, with typical models including Linear Support Vector Machine and Deep One-Class Support Vector Data Description. This architecture does not require pre-storing sample data of other users locally on the smartphone. Instead, it only requires the samples

IEEE Proof

captured from the user during the enrollment phase to complete model training and deployment ❸ ❺. Its core idea lies in the fact that, in practice, it is usually impossible to obtain the attacker's sample data or data distribution in advance.

All workflow processes of the architectures in Figs. 1a-c are implemented locally, while those in Figs. 1d-f adopt the server-client architecture. In the architecture of Fig. 1d, the service provider is required to pre-train a feature extractor, and the training data is independently collected by the provider and has no relevance to service users ❶. This feature extractor is typically constructed based on deep learning techniques, such as Convolutional Neural Networks, attention mechanisms, and Transformer architectures. After users deploy the pre-trained feature extractor on local devices ❷, the built-in sensors of smartphones capture the user behavior data ❸ during the enrollment phase for the training of one-class classifiers or binary classifiers ❹. Specifically, the collected data is first mapped to feature vectors via the feature extractor, which are then fed into the classifiers for training. In the verification phase, the interaction data between the user and the smartphone captured by the sensors ❺ is also converted into feature vectors through the feature extractor, which are ultimately used for user identity authentication ❻ ❼.

Different from the architecture in Fig. 1d, the architecture in Fig. 1e does not require training a user-specific classifier on the client side. Instead, both feature extraction model and classifier are trained by the service provider (hereinafter referred to as the classifier) ❶. The core function of this type of classifier is to determine whether two samples originate from the same user, which imposes higher requirements on the generalization ability of the classifier. In the enrollment phase, the user only needs to provide a small number of template samples ❸. In the authentication phase, the template samples and the samples to be verified ❹ are jointly fed into the classifier ❺, which then outputs a binary judgment on their user consistency: a match indicates successful authentication, while a mismatch results in authentication rejection ❻.

The architecture in Fig. 1f is similar to that in Fig. 1a, with the core difference lying in the following aspects: the binary classifier in Fig. 1f is trained by the service provider ❸, and the user is required to upload personal data collected during the enrollment phase to the server ❷. The core idea is that users' smartphones have limited memory and computational power. After the classifier is trained, the user then downloads it to the local device for deployment ❹ and use ❺ ❻. However, this architecture poses a notable privacy leakage risk, as it requires uploading the user's private data to the server.

IV. Threat Model

In order to assess the security of touch-based authentication (TBA), a realistic and concrete adversary model is necessary. Defining the adversary model is the first step in conducting security evaluation. Upon a comprehensive investigation of attacks documented in the literature [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], and [61], the adversary's capabilities are summarized as follows.

- A1. $\mathcal{A}$ can gain physical access to users' devices and bypass the entry point authentication by leveraging stolen credentials (e.g., PINs) or technical approaches.
- A2. A1.] $\mathcal{A}$ can capture partial behavioral patterns of users through shoulder surfing [16], [25], [46], smudge attacks [25], and then attempt to mimic them.
- A3. $\mathcal{A}$ can analyze the behavioral distribution characteristics of the collected dataset from other users, identify key clusters via statistical learning methods, and further generate a prioritized guessing list.
- A4. $\mathcal{A}$ can generate a guessing list composed of random samples for unauthorized access attempts.
- A5. $\mathcal{A}$ can inject data into the authentication pipeline.
- A6. $\mathcal{A}$ can get the target's behavioral biometric samples.
- A7. $\mathcal{A}$ has the ability to train robots with obtained data.

The capability A1 is reasonable. Although fingerprint recognition and facial recognition are more secure authentication methods than PINs, PINs remain one of the indispensable authentication methods [10] for smartphones due to the potential failure of the former two methods in certain scenarios. However, PINs are vulnerable to various types of attacks, such as online guessing attacks [10], [11], [66], shoulder surfing attacks [13], [14], and side-channel attacks [67]. Bailey et al. [11] showed that online guessing attacks can crack 10% of PINs. Shoulder surfing attacks achieve a success rate of 43.8% with just a single observation [13]. The capability A2 is based on the fact that attackers may obtain information about users' operation patterns, such as touch trajectories and movement speeds, via shoulder surfing [16], [25], [46], camera surveillance or smudge attacks [25] on mobile phone screens.

Capability A3 is feasible, as adversaries can analyze behavioral characteristic distributions [37], [63] using public or self-collected datasets, perform clustering, and construct a prioritized guessing list. Compared with A3, Capability A4 is less complex, as it only requires adversaries to generate a guessing list composed of random samples [25], [37], [65] by leveraging prior knowledge such as the length and value range of feature vectors. Capability A5 requires adversaries to continuously obtain system access permissions and inject forged samples into the authentication interface. For Capability A6, adversaries can acquire behavioral biometric samples or their corresponding feature vectors of legitimate users through means such as social engineering [25]. Capability A7 requires adversaries to train imitators [37], such as training LEGO robots to mimic the sliding operations of legitimate users or training human imitators to generate attack vectors.

As shown in Table II, based on the adversary capabilities described above, existing studies [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64], [65] have defined seven types of attacks, namely zero-effort attack, imitation attack, population attack, robot attack, random-

IEEE Proof

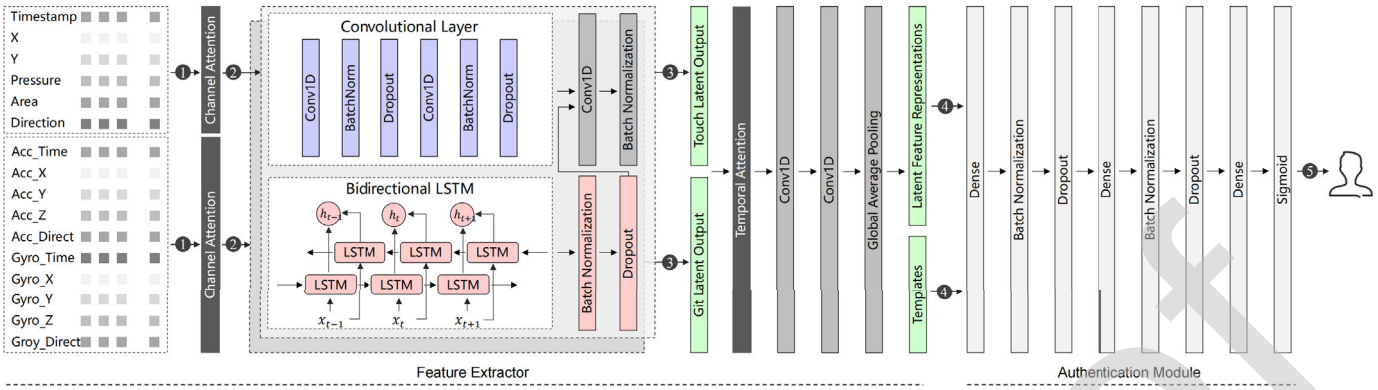


Fig. 2. The architecture of LuTBA. LuTBA consists of two main components: a feature extractor and a classifier. The feature extractor employs a 1D convolutional neural network combined with a bidirectional LSTM to map raw input data into a feature space. The classifier then takes two feature vectors as input and determines whether two samples belong to the same user.

vector attack, snoop-forge-replay attack, and robot mimicry attack. All these seven types of attacks require the adversary to be equipped with the capability A1, which is the basic assumption in the field of TBA. Specifically, population attack and random-vector attack require the adversary to inject data into the authentication pipeline. Snoop-forge-replay attack and robot mimicry attack require the adversary to obtain samples of target users. Population attack and robot attack mandate that the adversary extracts the most probable attack vector sequence from the public dataset. Robot attack and robot mimicry attack require the adversary to have the capability of training robots that are able to imitate human operations. Due to limitations in experimental conditions, in this paper we consider only zero-effort attack, population attack, random-vector attack, and imitation attack. Nevertheless, these attacks are more practical than the remaining types, as they incur lower costs and are easier to launch. For instance, an adversary may simply rely on their own behavioral patterns when using victims' phones, or may observe victims' habits and attempt to mimic them.

V. Design

LuTBA is structurally divided into two core components: a feature extractor $\mathcal{F}$ and a classifier $\mathcal{C}$. Specifically, $\mathcal{F}$ maps user behavior pattern data to the latent feature space and generates the corresponding latent feature representation, while $\mathcal{C}$ is responsible for determining whether this latent feature representation matches the template. Relevant code and parameter details have been published on GitHub.[1]

A. Feature Extractor

Fig. 2 depicts the detailed architecture of the feature extractor in LuTBA, which is designed to map motion sensor data and touch data into latent feature representations. Specifically, a channel attention mechanism ❶ is incorporated to assign distinct weights to each dimension of the raw features based on their respective importance. Formally, let $\mathbf{X} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_m\}$ represent the readings from the sensors, where m is the feature dimension. The attention mechanism can be formalized as $\mathbf{Y} = \{\vec{x}_1 \cdot w_1, \vec{x}_2 \cdot w_2, \ldots, \vec{x}_m \cdot w_m\}$, where

$$\vec{w} = \{w_1, w_2, \ldots, w_m\} = \text{ChannelAtt}(\mathbf{X}) \quad (1)$$

Then, the feature extractor first processes motion sensor data and touch data in two separate branches. Each processing branch consists of a convolutional layer and a bidirectional LSTM layer ❷. The convolutional layer is intended to extract local structural features, while the bidirectional LSTM is employed to capture long-range temporal correlations. By integrating both past (already executed touch actions) and future (subsequent touch actions) contextual information, the bidirectional LSTM enables a more comprehensive characterization of the contextual significance of individual touch actions within the entire sequence. The outputs of the two branches are concatenated, followed by a convolutional layer and batch normalization to generate latent outputs ❸. Formally, the latent output $\mathbf{L}$ can be formulated as:

$$\mathbf{L} = \text{Norm}(\text{Conv1D}(\text{ConvL}(\mathbf{Y}) \parallel \text{BiLSTM}(\mathbf{Y}))) \quad (2)$$

where ConvL denotes the convolutional layer, BiLSTM represents the bidirectional LSTM layer, Norm denotes the batch normalization layer and Conv1D represents 1D Convolutional Neural Network layer. With Equation 2, touch data is mapped to touch latent output $\mathbf{L}_t$ and motion sensor data is mapped to gait latent output $\mathbf{L}_s$. The concatenated latent outputs are then fed into a temporal attention mechanism, which calculates attention weights for each time step in the sequential data and appends these weights as a new channel to the original feature map. This mechanism quantifies the contribution of different time steps to the global feature representation, thereby enhancing the model's focus on critical time steps and suppressing the interference of irrelevant or noisy time steps. Subsequently, another convolutional layer is used to further extract cross-modal features from both gait and touch modalities. Finally, the resulting features are processed via global average pooling to yield the final latent feature representations $\vec{f}$, which can be formulated as:

$$\vec{f} = \text{GAP}(\text{Conv1D}(\text{Conv1D}(\text{TempAtt}(\mathbf{L}_t \parallel \mathbf{L}_s)))) \quad (3)$$

[1] https://github.com/Litraveler/LuTBA.git

IEEE Proof

where GAP denotes the global average pooling, TempAtt represents the temporal attention mechanism.

B. Authentication

The authentication module takes the difference vector between a feature representation to be verified and the template features $\vec{f_T}$ as input ❷. It processes the difference vector through three fully connected layers with batch normalization, ReLU activation, and dropout, and finally outputs the matching probability (ranging from 0 to 1) ❺ via Sigmoid activation. Formally, the probability p can be formulated as:

$$p = \text{Sigmoid}(\text{Dense}(\text{Dense}(\text{Dense}(\text{Norm}(\text{Drop}(\vec{f} \ominus \vec{f_T})))))) \tag{4}$$

where $\ominus$ denotes the element-wise subtraction of the two vectors, Drop denotes the dropout layer.

C. Training Strategy

We employ the triplet loss [26] to train the feature extractor. The objective of the triplet loss is to ensure that latent feature representations from the same user are positioned in close proximity within the feature space, while vectors representing samples from different users are pushed far apart from each other. The triplet loss function $\mathcal{L}_t$ is formulated as:

$$\mathcal{L}_t = \max\left(d(\mathcal{F}(a), \mathcal{F}(p)) - d(\mathcal{F}(a), \mathcal{F}(n)) + \alpha, 0\right) \tag{5}$$

where a, p, n represent the anchor sample, positive sample, and negative sample, respectively, the margin hyperparameter α enforces a minimum separation distance between positive and negative pairs, preventing the model from collapsing to trivial solutions. The function $d(\cdot, \cdot)$ computes the L2 distance of two feature representation vectors.

With a well-trained feature extractor, we use binary cross-entropy loss to train the authentication module. The loss function $\mathcal{L}_b$ is formulated as follows:

$$\mathcal{L}_b = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}) \tag{6}$$

where the label $y \in \{0, 1\}$ equals 1 when the two input samples are from the same user, and 0 otherwise. It is worth noting that the parameters of the well-trained feature extractor need to be fixed when training the authentication module.

VI. Study Design

To evaluate the effectiveness of LuTBA against imitation attacks, we conduct a controlled laboratory experiment based on the iOS application and task design developed in [40]. We reproduce the original Social Media Task and Image Gallery Task to ensure that the experimental interactions are consistent with those used in [40]. The original application is designed to collect touch interactions during common smartphone activities, including vertical scrolling and horizontal swiping. We adopt an observation-and-imitation procedure to investigate whether attackers can reproduce a registered user's behavioral patterns through direct observation.

A. Participants and Devices

A total of seven participants are recruited for our experiment, including two users and five attackers. The participants consist of four females and three males, all of whom are undergraduate or graduate students at university. Each participant receives a compensation of RMB 50 for completing this experiment. For devices, two iPhone models, an iPhone 13 Pro and an iPhone 14 Plus, are used in the experiment.

B. Application and Tasks

We use the iOS application developed in the reference study [40] for data collection. The application contains two tasks: the Social Media Task and the Image Gallery Task. Participants are asked to perform five rounds for each task.

1) Social Media Task: The Social Media Task simulates common vertical scrolling behavior when browsing social media feeds or news lists. In the original design, participants are required to scroll through a feed and identify an article or post matching a given description. In our experiment, we retain the same interaction procedure while replacing the original content with Chinese news articles. Specifically, we collect 40 news articles covering five topics from China News Service and manually construct a unique description for each article. During each round, one topic is selected, and 10 news articles randomly sampled from the entire dataset are presented to participants, including the target articles from the selected topic. Participants are required to browse the feed and identify the article corresponding to the given description.

2) Image Gallery Task: The Image Gallery Task simulates horizontal swiping behavior when browsing a sequence of images. In the original task, participants are presented with a horizontal sequence of images and asked to count occurrences of a specified target object while swiping through the gallery. We adopt the original task design without modification, including its interaction procedure and task requirements.

C. Imitation Attack Procedure

The experiment is designed to simulate an attacker who obtains information about a target user's behavioral patterns through direct observation and subsequently attempts to imitate those patterns. The experiment consists of three stages: registration, observation, and imitation.

During the registration stage, the two registered users independently complete the experimental tasks using their devices. They are instructed to perform the tasks naturally without deliberately controlling their touch behavior. The resulting interactions are treated as the registered behavioral profiles of the target users. Attackers are allowed to freely select their viewing positions around the target user while the registered user performs the tasks. Attackers can directly observe the target user's smartphone interactions and behavior patterns. In particular, they can observe visually accessible characteristics, including touch locations, scrolling or swiping speed, interaction rhythm, phone-holding posture, and the angle at which the phone is held. During the imitation stage, the five attackers are instructed to reproduce the observed behavioral patterns of the registered users as closely as possible. Each

IEEE Proof

TABLE III
STATISTICS OF DATASETS

Dataset	Sessions	Experiment Duration	Number of participants	Size
D_1	≤ 31	31 days	470	1,166,092
D_2	≤ 5	2 weeks	600	48,847
D_3	1	1 day	7	2,870

attacker completes the same experimental tasks using targets' devices and attempts to imitate their behavioral patterns. The objective is not merely to complete the tasks correctly, but to reproduce the target user's characteristic interaction patterns based on the information obtained during observation.

Each experimental session consists of 10 rounds, including five rounds of the Social Media Task and five rounds of the Image Gallery Task. The same task structure and interaction conditions are maintained for both registered users and attackers. The original study similarly used five rounds for each task to obtain touch data across the two task types.

Overall, the experimental design establishes a controlled "registration-observation-imitation" attack scenario. By keeping the application, task structure, device, and indoor experimental environment consistent, the experiment specifically investigates whether an attacker can reproduce a registered user's touch-based behavioral characteristics through direct observation and thereby challenge the security of LuTBA.

D. Datasets

As shown in Table III, we collect 2,870 touch interaction samples from the imitation attack experiment. Each participant completes 10 rounds of tasks, including five rounds of the Social Media Task and five rounds of the Image Gallery Task. The average completion time of all tasks is 613.69 seconds. This dataset has been published on IEEE DataPort.[2]

To answer the three research questions presented in Section VII, we also adopt two public multimodal datasets. Specifically, the dataset D_1 provided by Georgiev et al. [40] is collected in a longitudinal remote study with 470 participants over up to 31 days on nine types of iOS devices, containing a total of 1,166,092 swipe samples. Participants complete two daily interaction tasks simulating social media vertical scrolling and gallery horizontal browsing. The dataset D_2, namely HuMIdb [68], is a public multimodal mobile interaction dataset containing sensor recordings collected from 600 volunteers across 14 countries, with up to 5 acquisition sessions per user and over 230 distinct smartphone models involved. Data from fourteen built-in mobile sensors are synchronously captured during eight predefined human-device interaction tasks. In this work, we only adopt touch trajectory data generated from the upward and downward swipe tasks, which yields a total of 48,847 touch samples.

E. Ethical Consideration

Our research design and objectives comply with all relevant local laws, regulations, and ethical principles. Before the experiment, each participant was fully informed of the associated risks and potential benefits, and explicitly told that the collected data would be used solely for experimental research purposes. All datasets are anonymized before storage, removing any possible link to personal information. The dataset related to imitation attack has been uploaded to IEEE DataPort. All shared data remain fully anonymized to prevent any re-identification of participants.

[2]DOI: 10.21227/wnfa-v366.

VII. EXPERIMENTS

In this section, we conduct comprehensive experiments to evaluate the effectiveness of LuTBA. For each architecture, we select the state-of-the-art multi-modality models ([21], [29], [35], [45]) from existing literature (see Section III). For the offline binary classifier architecture illustrated in Fig. 1a, we employ the bidirectional long short-term memory model proposed by Abuhamad et al. [45]. Given that the offline binary classifier architecture (see Fig. 1a) shares the same model structure as the online binary classifier architecture (see Fig. 1f), we adopt the model proposed by Mahfouz et al. [29]. For the offline one-class classifier architecture presented in Fig. 1c, we utilize the one-class support vector machine model adopted by Ray-Dowling et al. [35]. For the pre-trained offline binary classifier architecture (see Fig. 1d), we select the model proposed by Delgado-Santos et al. [21]. It is worth noting that multi-modal models based on the pre-trained offline classifier architecture (see Fig. 1e) have not been reported in the literature to date. Our experiments aim to answer the following questions:

- **RQ1**: How does LuTBA perform with limited enrollment samples compared with the baseline models of the other five architectural paradigms?
- **RQ2**: What is the impact of temporal drifts in user behavioral patterns on the performance of LuTBA?
- **RQ3**: How secure is LuTBA against random-vector attack, population attack, and imitation attack?

All the experiments are conducted on a server running Microsoft Windows 11, equipped with 16 GB of memory, an Intel Core i5-14400F CPU (10 cores / 16 threads) and NVIDIA GeForce RTX 4060. We use PyTorch 2.0.1 to implement LuTBA and baseline models.

A. Experimental Setup

To enable fair comparison across different studies and ensure unbiased performance evaluation, we follow a standard set of practices proposed by Georgiev et al. [40], including:

- **Contiguous training data selection.** To mimic real-world deployment, test data are collected at a time point subsequent to the collection of training data.
- **Exclusion of attackers from the training set.** Consistent with the assumption that no information about attackers is available in advance, we exclude any attacker samples from the negative examples used for model training.
- **Window size.** To ensure comparability with related works, we only report the performance of single-swipe models in our experiments.

IEEE Proof

- **Dataset and code availability.** To guarantee the reproducibility of our experiments, we make all relevant code publicly available. In addition, we also make the imitation attack dataset publicly available.

B. Evaluation Metrics

In this paper, we use six different metrics in the evaluation: False Acceptance Rate (FAR), False Rejection Rate (FRR), Equal Error Rate (EER), Accuracy, True Positive Rate (TPR), and Area Under the Curve (AUC) of Receiver Operating Characteristic (ROC) [27]. These evaluation metrics are based on the number of correctly (TP) and incorrectly (FN) classified benign authentication events as well as the number of correctly detected (TN) and unrecognized (FP) attack attempts. More details of them are as follows:

Accuracy: Accuracy represents the proportion of correctly predicted samples to the total number of samples. Accuracy is defined as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{7}$$

FAR: FAR represents the proportion of attack attempts incorrectly classified as benign events and is defined as:

$$\text{FAR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{8}$$

FRR: FRR represents the proportion of benign events incorrectly rejected and is defined as:

$$\text{FRR} = \frac{\text{FN}}{\text{FN} + \text{TP}} \tag{9}$$

EER: EER is the error rate at the threshold where FAR and FRR are equal. It represents the trade-off between FAR and FRR across different thresholds.

AUC: AUC is the area under the ROC curve, which plots the TPR against the FAR at various classification thresholds. The AUC quantifies the overall ability of the system to discriminate between benign authentication events and attack attempts across all possible thresholds, where a higher AUC indicates better performance (1.0 represents perfect discrimination).

C. RQ1: The Effect of Sample Size

Existing studies [23], [36] have demonstrated that the size of training data has a significant impact on the performance of authentication models. In this section, we investigate the effect of training/enrollment sample sizes on authentication model performance. For dataset $\mathcal{D}_1$, the test sample size is fixed at 800 (with 400 genuine samples and 400 impostor samples). The training sample sizes for the binary classification model are set to 200, 400, and 800, respectively, while those for the one-class classification model (containing only genuine samples) are 100, 200, and 400, respectively. For dataset $\mathcal{D}_2$, the test sample size is fixed at 200. The training sample sizes for the binary classification model are 100 and 200, respectively, and those for the one-class classification model (containing only genuine samples) are 50 and 100, respectively.

For LuTBA and the Transformer model designed in [21], we partition all users into training and test sets at a ratio of 7:3. For the training set, we limit each user to 15 samples and generate the training set composed of triplets and pairs required for training. For the test set, the enrollment sample size per user of LuTBA is fixed at 10, where these 10 samples are obtained using the KMeans clustering algorithm from the `sklearn.cluster` library in Python. For the model proposed in [21], the authors attached an additional binary classifier to the pre-trained feature extractor. The feature representations generated by the extractor serve as the input of the classifier to refine the authentication model. In our experiments, we set the training sample sizes of the binary classifier to 200, 400, and 800.

Since LuTBA and M1 both adopt pre-trained feature extractors, a fair comparison between them can be conducted with identical settings. However, fair comparisons between LuTBA and the one-class as well as binary-classification models (M2-M5) under identical settings are impractical, since each user has only a limited quantity of samples. Even so, the reliability of our experimental outcomes remains unimpaired. First, we perform strict comparisons between LuTBA and M1 with unified training configurations, which provides a controlled comparison for architecture-level assessment. Second, the comparisons of LuTBA against M2 to M5 reflect model performance within real-world deployment scenarios, in which the training data supplied by end-users is inherently limited.

The experimental results are presented in Table IV. Under the condition of fixed enrollment sample size, LuTBA achieves the optimal performance. For dataset $\mathcal{D}_1$, compared with M1-M5, LuTBA improves the mean accuracy by 1.63%-16.50%, with a maximum reduction of 5.82% in accuracy standard deviation; enhances the mean AUC by 2.21%-22.14%, with the reduction range of 0.72%-9.79% in AUC standard deviation; decreases the mean Equal Error Rate (EER) by 2.24%-20.22%, with the maximum reduction of 7.62% in EER standard deviation; and lowers the mean False Negative Rate at 0.1 False Positive Rate (FNR@0.1) by 2.07%-40.83%, with the reduction range of 2.45%-8.84% in FNR@0.1 standard deviation.

For dataset $\mathcal{D}_2$, compared with M1-M5, LuTBA increases the mean accuracy by 2.62%-21.05%, with the reduction range of 0.23%-5.08% in accuracy standard deviation; raises the mean AUC by 2.98%-27.44%, with the reduction range of 5.34%-12.57% in AUC standard deviation; reduces the mean EER by 2.08%-24.83%, with the reduction range of 0.80%-6.45% in EER standard deviation; and cuts down the mean FNR@0.1 by 3.09%-52.26%, with the maximum reduction of 7.19% in FNR@0.1 standard deviation.

Given that the Shapiro-Wilk normality tests indicate non-normal distributions for the EERs of each model ($p < 0.001$), we employ one-sided (left-tailed) Mann-Whitney U tests to compare the EER of LuTBA against five baseline schemes (M1-M5), with the alternative hypothesis that LuTBA achieves statistically lower EER. Specifically, we employ the EER results obtained with the maximum training sample sizes from Table IV, which also yield the best performance across all configurations. The statistical test results yield p-values all below 0.05, with the maximum p-value being 0.042. These results demonstrate that the performance improvements of LuTBA over all baselines are statistically significant.

IEEE Proof

TABLE IV
THE PERFORMANCE OF LUTBA AND OTHER BASELINE MODELS

Source	Mark	Training/Enrollment Samples	Testing Samples	Accuracy Mean	Accuracy Std	AUC Mean	AUC Std	EER Mean	EER Std	FNR@0.1 Mean	FNR@0.1 Std
					$\mathcal{D}_1$						
Delgaso-Santos et al. [21]	M1	200	800	83.29%	6.80%	87.95%	8.36%	19.16%	8.68%	40.07%	25.79%
		400	800	83.81%	6.06%	89.08%	6.57%	18.19%	7.22%	37.10%	23.28%
		800	800	84.65%	5.90%	89.86%	6.59%	17.19%	7.08%	33.69%	23.01%
Ray-Dowling et al. [35]	M2	100	800	69.78%	10.21%	69.93%	14.36%	35.17%	13.12%	65.51%	22.99%
		200	800	72.60%	10.47%	73.24%	15.46%	31.67%	14.31%	59.29%	25.10%
		400	800	73.31%	10.33%	74.51%	14.74%	30.01%	13.25%	57.71%	25.32%
Mahfouz et al. [29]	M3	200	800	79.20%	11.61%	83.48%	12.95%	23.15%	13.40%	40.26%	26.74%
		400	800	79.84%	12.60%	83.59%	14.77%	22.45%	14.70%	39.02%	29.40%
		800	800	84.34%	11.17%	88.98%	11.70%	17.33%	12.73%	27.61%	26.14%
	M4	200	800	78.80%	10.98%	80.86%	14.52%	25.39%	14.05%	47.84%	27.51%
		400	800	80.62%	11.27%	82.64%	15.64%	23.26%	14.91%	42.88%	28.93%
		800	800	83.26%	10.03%	86.22%	13.02%	19.77%	13.24%	34.92%	27.68%
Abuhamad et al. [45]	M5	200	800	74.22%	11.03%	73.99%	15.56%	31.14%	14.26%	66.37%	25.66%
		400	800	76.46%	10.55%	77.19%	14.64%	28.41%	13.24%	61.26%	27.85%
		800	800	78.92%	9.12%	80.71%	12.15%	25.37%	11.38%	55.91%	27.08%
This work		10	800	86.28%	6.78%	92.07%	5.85%	14.95%	7.29%	25.54%	20.56%
					$\mathcal{D}_2$						
Delgaso-Santos et al. [21]	M1	100	200	71.38%	8.99%	74.93%	12.40%	31.71%	10.34%	62.04%	20.61%
		200	200	74.13%	9.00%	78.47%	11.51%	27.80%	9.81%	56.73%	19.97%
Ray-Dowling et al. [35]	M2	50	200	68.05%	10.67%	66.65%	17.65%	37.48%	14.37%	70.00%	22.86%
		100	200	68.76%	12.44%	67.50%	18.74%	36.45%	15.46%	67.87%	28.67%
Mahfouz et al. [29]	M3	100	200	82.90%	12.30%	87.32%	13.43%	18.56%	13.56%	32.82%	30.21%
		200	200	85.74%	12.57%	89.43%	12.59%	15.97%	14.53%	26.13%	29.13%
	M4	100	200	79.64%	11.52%	82.11%	14.81%	23.91%	14.32%	44.25%	30.22%
		200	200	83.45%	11.05%	86.26%	11.88%	19.71%	13.53%	35.82%	30.10%
Abuhamad et al. [45]	M5	100	200	67.31%	7.72%	64.97%	12.22%	38.72%	10.03%	75.30%	19.54%
		200	200	71.62%	10.98%	70.18%	15.81%	34.76%	14.20%	67.58%	22.58%
This work		10	200	88.36%	7.49%	92.41%	6.17%	13.89%	9.01%	23.04%	23.03%

1) Discussion: In summary, LuTBA outperforms M1-M5 on both datasets when the enrollment sample size is only 1/80 of that of M2-M5. Specifically, LuTBA does not require collecting a large number of user samples for model training. Instead, it only needs a certain number of template samples during the enrollment phase, which effectively reduces the time cost of deploying the authentication model on the user side. In comparison to M1, LuTBA can extract features with higher discriminability, leading to improved performance.

D. The Effect of Time

This section aims to investigate the impact of time on model performance. Existing studies have shown that user behavior patterns shift over time, which in turn leads to a decline in model performance.

In this section, we compare the proposed LuTBA with the two baseline models (M1 and M2) that achieved the best performance in Section VII-C. For a fair comparison, both LuTBA and the feature extractor of M1 are pre-trained under identical settings. The binary classification models of M1 [21] and M2 [29] are then fine-tuned on the same training set. Specifically, for dataset $\mathcal{D}_1$, we use 400 training samples (200 positive, 200 negative) and 800 test samples. For dataset $\mathcal{D}_2$, we use 50 training samples and 100 test samples. The initial enrollment sample size of LuTBA is set to 10.

We define a session as a block of 400 consecutive samples, and evaluate all models at intervals of 1, 3, 5, 7, and 9 sessions for $\mathcal{D}_1$. For $\mathcal{D}_2$, we limit the evaluation to intervals of 1 and 3 due to the limited data span. For LuTBA, which has a dynamically updatable template library, we simulate the temporal evolution by extracting 10 template samples from adjacent blocks using the `KMeans++` clustering algorithm, thereby enabling updates to the enrollment templates as the session progresses. In experiments, extracting 10 cluster centroids from 400 feature vectors randomly sampled in each trial exhibits an average runtime of 74.07 ms across 30 independent trials, with a standard deviation of 4.46 ms.

The experimental results are shown in Table V. For dataset $\mathcal{D}_1$, as the session interval increases, M1 and M2 both show a downward trend in the mean accuracy and mean AUC, while LuTBA shows an upward trend. In terms of mean EER and mean FNR@0.1, M1 and M2 show an upward trend, while LuTBA shows a downward trend. In terms of standard deviation (std), the std reported by M1 and M2 in the four evaluation metrics all increases with the increase of session interval, while LuTBA remains stable. Specifically, when the session interval is 9, compared with M1 and M2, the mean accuracy of LuTBA is improved by 5.39% and 7.11 % respectively, the mean AUC is improved by 5.54 % and 7.99% respectively, the mean EER is reduced by 7.45% and 6.37% respectively, and the mean FNR@0.1 is reduced by 21.16% and 17.38% respectively.

For dataset $\mathcal{D}_2$, over time, the mean accuracy and mean AUC of M2 show a downward trend, while LuTBA and M1 show an upward trend; the mean EER and mean FNR@0.1 of M2 show an upward trend, while LuTBA and M1 show a downward trend. In terms of std, the std of M2 in the four types

IEEE Proof

TABLE V
THE EFFECT OF TIME

Source	Interval	Training/Enrollment Samples	Testing Samples	Accuracy Mean	Accuracy Std	AUC Mean	AUC Std	EER Mean	EER Std	FNR@0.1 Mean	FNR@0.1 Std
					$\mathcal{D}_1$						
Delgaso-Santos et al. [21]	1	400	800	84.43%	6.53%	89.99%	6.70%	17.38%	7.65%	32.80%	22.34%
	3	400	800	83.66%	6.40%	88.76%	7.35%	18.77%	7.95%	37.64%	23.16%
	5	400	800	83.72%	6.67%	89.00%	7.21%	18.37%	8.04%	36.58%	23.61%
	7	400	800	82.77%	7.01%	88.02%	7.55%	19.55%	8.33%	40.09%	24.00%
	9	400	800	83.47%	6.76%	88.49%	7.76%	19.09%	8.55%	37.75%	24.47%
Mahfouz et al. [29]	1	400	800	84.32%	11.15%	88.98%	11.72%	17.38%	12.70%	27.60%	26.01%
	3	400	800	82.40%	11.96%	86.72%	13.04%	19.29%	13.14%	33.45%	28.81%
	5	400	800	81.90%	12.08%	85.75%	14.21%	20.14%	14.09%	35.05%	30.25%
	7	400	800	82.20%	12.18%	86.28%	14.28%	19.50%	13.85%	33.24%	29.66%
	9	400	800	81.75%	12.63%	85.95%	13.76%	19.82%	13.78%	33.97%	29.97%
This work	1	10	800	86.28%	6.78%	92.07%	5.85%	14.95%	7.29%	25.54%	20.56%
	3	10	800	85.71%	6.29%	91.60%	5.55%	15.56%	6.78%	28.35%	20.61%
	5	10	800	86.61%	6.37%	92.34%	5.59%	14.82%	6.99%	24.43%	19.41%
	7	10	800	87.45%	7.22%	92.80%	6.43%	13.96%	7.97%	22.45%	22.06%
	9	10	800	88.86%	6.54%	93.94%	4.63%	12.37%	6.72%	16.59%	15.54%
					$\mathcal{D}_2$						
Delgaso-Santos et al. [21]	1	50	100	73.67%	9.43%	76.87%	12.03%	29.20%	10.37%	57.77%	22.98%
	3	50	100	75.27%	9.20%	78.67%	11.88%	26.67%	9.14%	54.93%	27.34%
Mahfouz et al. [29]	1	50	100	85.85%	11.15%	89.92%	11.83%	15.42%	13.05%	25.33%	27.36%
	3	50	100	80.92%	12.30%	82.74%	14.66%	21.76%	14.08%	50.63%	39.39%
This work	1	10	100	89.50%	7.76%	92.78%	7.28%	13.67%	9.68%	21.15%	26.49%
	3	10	100	90.08%	5.37%	93.82%	4.16%	11.50%	6.78%	27.17%	23.71%

of indicators increases with the increase of session interval, M1 remains stable, and LuTBA decreases slightly. Specifically, when the session interval is 3, compared with M1 and M2, the mean accuracy of LuTBA is improved by 14.81% and 9.16% respectively, the mean AUC is improved by 15.15% and 11.08% respectively, the mean EER is reduced by 15.17% and 10.26% respectively, and the mean FNR@0.1 is reduced by 27.76% and 23.46% respectively.

1) Discussion: In summary, the experimental results revalidate the conclusion of existing studies [23], [24], [50]: user behavior patterns shift over time, leading to a decline in model performance. However, the dynamic update mechanism of LuTBA can effectively adapt to changes in user behavior patterns by updating the template library with samples from recent adjacent time periods. Surprisingly, the performance of LuTBA increases over time, likely due to the gradual stabilization of user behavior patterns as users become familiar with the data collection task scenario.

E. Security Evaluation

In this section, we evaluate the security of LuTBA against population attack (see Attack Type III in Table II), random-vector attack (see Attack Type V in Table II), and imitation attack (see Attack Type II in Table II). From the perspective of implementation difficulty, population attack and random-vector attack are more difficult to carry out than imitation attack, as they require the attacker to have the ability to bypass sensors and inject data into the authentication pipeline. In contrast, imitation attack only requires the attacker to observe and learn the behavioral patterns of the target user.

1) Random-Vector Attack: For the random-vector attack, we adopt the Random Input Attack proposed by Zhao et al. [65], which generates random samples with time-series characteristics to launch attacks against the authentication model. The detailed algorithm is presented in Algorithm 2. In addition, we additionally consider the Complete Random Sample Attack to assess its impact on the model (see Algorithm 1). Fig. 3a presents the results of the Temporal Random Sample Attack, and Fig. 3b reports the results of the Complete Random Sample Attack. The Equal Error Rate (EER) threshold derived from LuTBA's testing on 1600 samples is used as the evaluation benchmark. As shown in Table VI, under this threshold, both the Temporal Random Sample Attack and the Complete Random Sample Attack achieve a False Acceptance Rate (FAR) of 0 and a False Rejection Rate (FRR) of 0.5%. These results indicate that the LuTBA model can effectively resist random input attacks.

TABLE VI
THE RESULTS OF RANDOM-VECTOR ATTACK AND POPULATION ATTACK

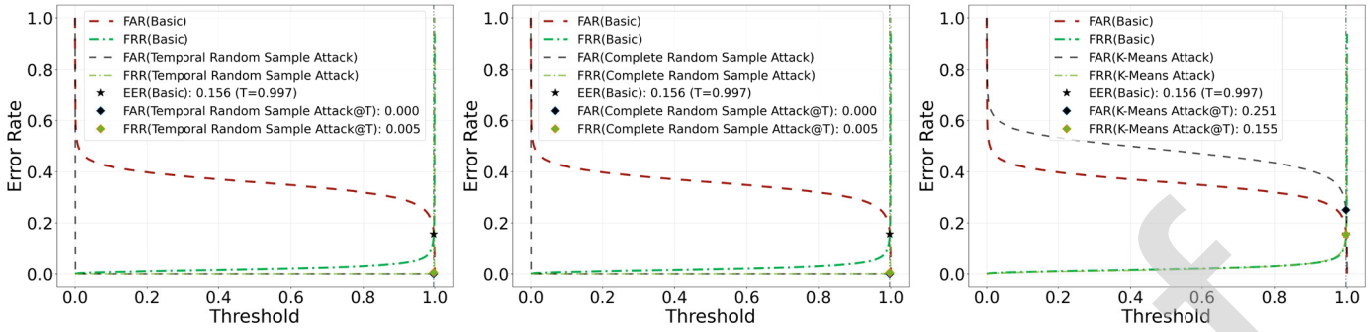
Attack	FAR	FRR
Temporal Random Sample Attack	0.000	0.005
Complete Random Sample Attack	0.000	0.005
Targeted K-means Attack	0.251	0.155

Algorithm 1 Complete Random Sample Generation

Input: The maximum sequence length $\mathcal{L}$; Feature dimension $\mathcal{D}$; Touch type $\mathcal{T}$
Output: A random sample $\mathcal{S}$.

$\mathcal{L}_v$ = random.randint(1, $\mathcal{L}$) /* $\mathcal{L}_v$ is the effective sequence length. */
for $i \in [0, \mathcal{D}]$ **do**
    $\mathcal{F}_i$ = random.uniform(0, 1, $\mathcal{L}_v$); /* $\mathcal{F}_i$ is the random sequence of the i-th feature dimension. */
S_v = np.column_stack($\mathcal{F}_0, ..., \mathcal{F}_{D-1}, \mathcal{T}$)
$\mathcal{S}$ = full(S_v, $\mathcal{L}$)
return $\mathcal{S}$

IEEE Proof



(a) Temporal Random Sample Attack (b) Complete Random Sample Attack (c) Targeted K-means Attack

Fig. 3. Figs. 3a–3b present the experimental results of the Temporal Random Sample Attack, Complete Random Sample Attack, and Targeted K-means Attack compared with the baseline model. Overall, the Temporal Random Sample Attack and Complete Random Sample Attack are ineffective, while the Targeted K-means Attack increases the False Acceptance Rate by 9.5% and the False Rejection Rate by 0.1%.

Algorithm 2 Temporal Random Sample Generation

Input: The maximum sequence length $\mathcal{L}$; Feature dimension $\mathcal{D}$; Touch type $\mathcal{T}$
Output: A random sample $\mathcal{S}$.

$\mathcal{L}_v$ = random.randint(1, $\mathcal{L}$) /* $\mathcal{L}_v$ is the effective sequence length.*/
valid_timestamps = np.linspace(0, 5, $\mathcal{L}_v$) **for** $i \in [1, \mathcal{D}]$ **do**
  $\mathcal{F}_i$ = generate_correlated_sequence($\mathcal{L}_v$, car_param); /* generate_correlated_sequence generates sequences with temporal correlation; car_param controls the strength of the correlation.*/
S_v = np.column_stack(valid_timestamps, $\mathcal{F}_1, ..., \mathcal{F}_{D-1}, \mathcal{T}$)
$\mathcal{S}$ = full(S_v, $\mathcal{L}$)
return $\mathcal{S}$

2) Discussion: Different from traditional binary classification models, LuTBA does not use target users' samples for training. Instead, it determines whether two input samples come from the same user by judging their similarity. This characteristic breaks the core logic of random input attacks: traditional binary classification models divide the feature space into an acceptance region and a rejection region, where the scope of the acceptance region is much larger than the distribution range of the target user's genuine positive samples. The empty regions in the feature space that are not covered by genuine samples will be classified as the acceptance region or rejection region by the classifier, and random input samples are likely to fall into such regions. For LuTBA, input samples are mapped to latent feature representations. Samples from the same user have close distances in the latent feature space, while samples from different users have relatively large distances. LuTBA determines the user identity by judging the similarity of latent feature representations. Experimental results demonstrate that randomly generated attack samples are difficult to approximate the feature vectors of target users.

3) Population Attack: For population attack, we employ targeted K-means++ attack proposed by Negi et al. [63]. After collecting a large volume of touch data from other users, the attacker leverages the clustering properties of the K-means++ algorithm (see Algorithm 3) to efficiently explore the sample space, and prioritizes selecting cluster centers that are far from the attempted samples as attack samples to improve the attack success rate.

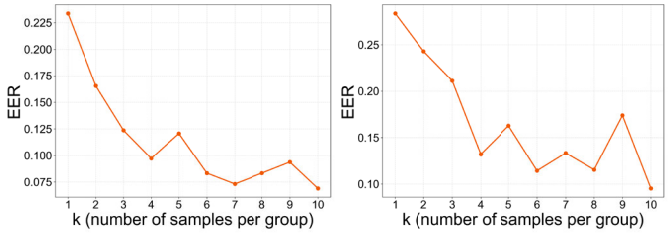
Algorithm 3 K-means++ Algorithm

Input: Data matrix $X \in \mathbb{R}^{N \times F}$ (N samples, F features); start_mean (bool, whether to use mean as first center); max_samples (int, max number of samples to use, default None)
Output: Initial cluster centers $C = \{c_1, c_2, ..., c_K\}$ (K is the number of clusters)

$n_{\text{samples}}, n_{\text{features}}$ = shape(X);
if *max_samples* $\neq$ *None* **then**
  $n_{\text{samples}} = \min(n_{\text{samples}}, \text{max_samples})$;
$x_{\text{squared_norms}}$ = row_norms(X, squared = True);
$n_{\text{local_trials}} = 2 + \lfloor \ln(n_{\text{samples}}) \rfloor$;
random_state = np.random.RandomState(SEED);
if *start_mean* = *True* **then**
  c_1 = mean(X, axis = 0);
else
  center_id = random_state.randint(n_{samples});
  if X *is sparse* **then**
    c_1 = flatten(toarray(X[center_id]));
  else
    $c_1 = X$[center_id];
$C = \{c_1\}$;
closest_dist_sq = euclidean_distances($[c_1], X, Y_{\text{norm_squared}} = x_{\text{squared_norms}}$, squared = True);
current_pot = sum(closest_dist_sq);
$K = \min(n_{\text{samples}}, 20)$;
for $c \in [2, K]$ **do**
  rand_vals = random_state.random_sample($n_{\text{local_trials}}$) $\times$ current_pot;
  candidate_ids = searchsorted(cumsum(closest_dist_sq), rand_vals);
  distance_to_candidates = euclidean_distances(X[candidate_ids], $X, Y_{\text{norm_squared}} = x_{\text{squared_norms}}$, squared = True);
  best_candidate = None;
  best_pot = None;
  best_dist_sq = None;
  for *trial* $\in [1, n_{local_trials}]$ **do**
    new_dist_sq = minimum(closest_dist_sq, distance_to_candidates[trial]);
    new_pot = sum(new_dist_sq);
    if *best_candidate* = *None* $\vee$ *new_pot* $<$ *best_pot* **then**
      best_candidate = candidate_ids[trial];
      best_pot = new_pot;
      best_dist_sq = new_dist_sq;
  if X *is sparse* **then**
    c_c = flatten(toarray(X[best_candidate]));
  else
    $c_c = X$[best_candidate];
  $C = C \cup \{c_c\}$;
  current_pot = best_pot;
  closest_dist_sq = best_dist_sq;
return C

As shown in Fig. 3c and Table VI, at the threshold corresponding to the baseline test results, the Targeted

IEEE Proof



(a) (b)

Fig. 4. Figs. 4a and 4b present the experimental results of the imitation attack under different group-sample sizes. As the number of samples per group increases from 1 to 10, the EER shows a significant overall downward trend. This indicates that although attackers can improve the attack success rate via behavior imitation, it is difficult for attackers to continuously replicate the behavioral patterns of target users.

TABLE VII
EERs Under Different Group Sizes for Two Attack Targets

Group Size	1	2	3	4	5	6	7	8	9	10
Target 1	0.234	0.166	0.124	0.097	0.121	0.083	0.073	0.083	0.094	0.069
Target 2	0.284	0.243	0.211	0.132	0.163	0.114	0.133	0.115	0.174	0.095

K-means++ Attack yields a 25.1% FAR and a 15.5% FRR. Compared with the baseline, FAR increases by 9.5%, while FRR decreases by 0.1%.

4) Discussion: The above findings indicate that clustering can effectively improve the attack success rate compared with random attack samples. Existing studies [63], [69] have shown that population attack cannot be completely defeated by a single method, due to the inherent fuzzy matching [69] and clustering [63] mechanisms in behavioral biometric authentication. In practice, however, launching such an attack is non-trivial, as it requires the adversary to first compromise the system in order to inject crafted samples into the pipeline, which may be protected by the secure communication protocols [26].

5) Imitation Attack: The objective of the imitation attack experiment is to evaluate an attacker's ability to replicate user behavioral characteristics through observation. To assess the resilience of LuTBA against the imitation attack, we divided the attack samples into equal-sized groups in chronological order, and averaged the matching scores within each group. As shown in Fig. 4, the overall trend indicates that as the window size increases, the EER for both target users decreases significantly, suggesting that multi-sample testing effectively enhances the discrimination capability against mimicry attacks. As listed in Table VII, for target user 1, as the window size increases from 1 to 10, the EER drops from 0.234 to a minimum of 0.069; for target user 2, the EER declines from an initial 0.284 to 0.095.

6) Discussion: Notably, when the window size is 1, the mean EER is 0.259, indicating that the attacker successfully captured some behavioral patterns through observation, thereby improving attack success rate. However, when the window size reaches 4 or larger, the EER drops rapidly and stabilizes below 0.15, demonstrating that enlarging the authentication window can effectively boost LuTBA's resistance to mimicry attacks.

TABLE VIII
Computational Overhead on Two Devices

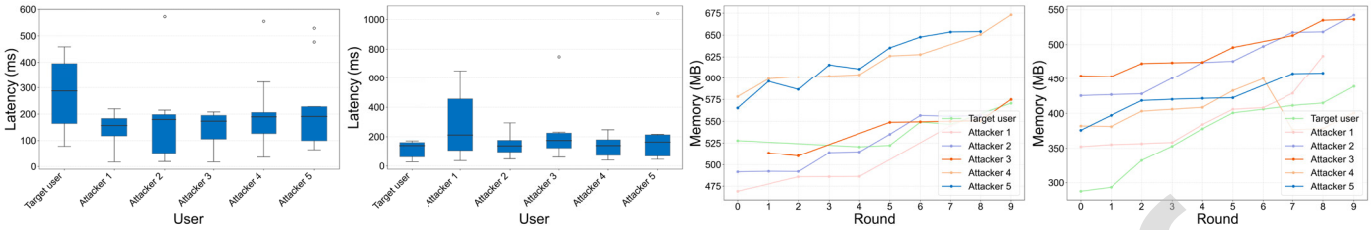
Role	Mean	Min	Max	Std
Target user 1	114.626	29.087	170.746	52.465
Attacker 1	268.579	37.834	651.062	209.985
Attacker 2	139.773	49.284	293.391	67.542
Attacker 3	220.626	62.186	748.077	194.165
Attacker 4	135.584	41.646	247.239	66.309
Attacker 5	246.169	45.931	1041.468	307.464
Target user 2	277.347	77.516	458.197	139.277
Attacker 1	141.984	17.921	219.563	63.943
Attacker 2	169.324	19.579	572.604	160.231
Attacker 3	142.596	17.971	207.630	70.237
Attacker 4	208.920	36.701	555.036	145.789
Attacker 5	226.642	63.983	528.991	158.442
Target user 1	371.575	287.422	438.766	50.178
Attacker 1	392.531	352.266	483.531	41.630
Attacker 2	475.738	425.500	542.312	40.321
Attacker 3	490.005	453.844	536.266	29.926
Attacker 4	401.948	376.625	450.031	23.155
Attacker 5	421.650	375.750	458.984	26.095
Target user 2	539.552	520.656	570.719	17.817
Attacker 1	512.402	469.094	558.094	35.852
Attacker 2	523.222	491.656	556.656	26.865
Attacker 3	542.616	510.500	575.062	21.172
Attacker 4	618.080	578.328	673.109	27.694
Attacker 5	618.368	565.312	654.078	29.895

F. On-Device Performance

In this section, we study the on-device performance of our authentication model on iOS. We use Core ML Tools to convert the trained PyTorch checkpoint into two Core ML packages (`.mlpackage`), which is a feature extractor that maps a touch sequence and a sensor sequence into a 256-dimensional embedding, and an authentication module that scores the absolute difference between the current embedding and the enrolled template. This format is compiled by Xcode into a `.mlmodelc` binary and packaged into the app bundle, which is optimized for on-device inference on Apple silicon via Core ML. We then deploy the converted models in the app and measure on-device latency, memory usage at the end of each completed task.

The upper blocks of Table VIII present the model inference latency, while the lower blocks show memory consumption. Specifically, the feature-extraction time per task round ranges from 17.92 ms to 1041.468 ms, with an overall average feature-extraction time of 191.02 ms. By dividing the total time by the sample count, we obtain an estimated per-sample inference latency of 6.65 ms. The memory overhead varies between 287.42 MB and 575.06 MB, including the overhead of the application, model, and data storage. On average, the memory consumption reaches 492.31 MB. As illustrated in Fig. 5, the overall memory overhead increases with the number of task rounds, which may stem from data accumulation during the experiment. Overall, both the memory overhead and inference latency are reasonable. According to prior works, inference latency below one second does not degrade user experience and can support online real-time services [27]. Therefore, the

IEEE Proof



(a) (b) (c) (d)

Fig. 5. Figs. 5a and 5b show the feature extraction time of LuTBA for samples obtained in each task round, while Figs. 5c and 5d illustrate the memory consumption of the experimental application. Affected by the number of samples, the feature-extraction time per task round ranges from 17.92 ms to 1041.468 ms, with an overall average feature extraction time of 191.02 ms. The memory overhead varies between 287.42 MB and 575.06 MB, including the overhead of the application, model, and data storage. On average, the memory consumption reaches 492.31 MB.

overhead measurements demonstrate that LuTBA is practical for real-world deployment.

VIII. Limitations

In real-world scenarios, the robustness of touch-based authentication systems may be influenced by three factors: the surrounding environment, user behavioral states, and application contexts.

A. Surrounding Environment

Users may interact with mobile devices across a wide range of transportation settings, including trains, high-speed rail, airplanes, cars, and subways. The ambient noise in these environments can interfere with or even overwhelm the sensor-captured behavioral signals, leading to a notable degradation in TBA system performance.

B. User Behavioral States

Mobile phone usage involves diverse postures, such as sitting, lying down, lying prone, and walking. Without comprehensive coverage of these behavioral states during training, the TBA system's performance may deteriorate sharply in unseen posture scenarios, undermining its practical reliability.

C. Application Contexts

Prior studies have shown that even though the same types of touch operations are performed across different applications, users' operating habits can vary significantly due to differences in application types and interface layouts. This cross-application variability tends to degrade authentication performance.

Current public datasets are predominantly collected from built-in mobile sensors and lack metadata on the surrounding environment, application contexts, and user behavioral states. This information gap imposes inherent limitations on improving existing TBA schemes. Without fine-grained classification and processing of these contextual factors, the authentication model's acceptance boundary becomes overly broad, which substantially increases the false acceptance rate. Furthermore, collecting large volumes of target user data locally for model training may be infeasible in certain deployment scenarios. To address these challenges, we provide the following recommendations for future research:

- **Refined annotation.** Future work is encouraged to adopt more fine-grained annotation of collected data. Such granularity facilitates scenario-specific data processing, enables more accurate extraction of user-specific behavioral patterns, and ultimately improves authentication performance.
- **Architecture selection.** The diversity of environmental, postural, and contextual conditions further complicates the collection of large-scale training data. We therefore advocate adopting a pre-trained offline classifier architecture (see Fig. 1e). Based on pre-collected data, future studies can design dedicated classifiers to actively infer the user's surrounding environment, behavioral state, and application context, enabling refined classification of enrollment samples and lightweight deployment with limited enrollment data.

IX. Conclusion

In this paper, by analyzing 44 relevant works, we identify six distinct architectural paradigms for touch-based authentication, and we systematically compare their workflows to highlight distinctions. Building upon pre-trained architectures, we propose LuTBA, a **L**ightweight and **u**pdatable **T**ouch-**B**ased **A**uthentication scheme that can better balance performance and deployment overhead. Experiments on two real-world datasets demonstrate that LuTBA not only achieves better performance but also requires fewer enrollment samples. Furthermore, the built-in lightweight update mechanism proves effective in counteracting performance decay caused by the gradual drift of users' touch behavior over time. Finally, on our self-collected imitation attack test set, although LuTBA exhibits performance degradation under the single-window setting, we find that enlarging the authentication window can effectively boost the overall authentication performance. This indicates that LuTBA is robust against imitation attacks by increasing the window size.

References

[1] Statista. (2025). *Number of Smartphone Mobile Network Subscriptions Worldwide From 2016 to 2025, With Forecasts From 2025 to 2028.* [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

IEEE Proof

[2] D. Wang, Q. Gu, X. Huang, and P. Wang, "Understanding human-chosen PINs: Characteristics, distribution and security," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Apr. 2017, pp. 372–385.

[3] R. Roberts, J. Poveda, R. Roberts, and D. Levin, "Blue is the new black (Market): Privacy leaks and re-victimization from police-auctioned cellphones," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2023, pp. 3332–3336.

[4] G. Cho, J. H. Huh, J. Cho, S. Oh, Y. Song, and H. Kim, "SysPal: System-guided pattern locks for Android," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 338–356.

[5] M. Zhou et al., "FingerPattern: Securing pattern lock via fingerprint-dependent friction sound," *IEEE Trans. Mobile Comput.*, vol. 23, no. 6, pp. 7210–7224, Jun. 2024.

[6] G. Li et al., "Fingerprinting image-to-image generative adversarial networks," in *Proc. IEEE 9th Eur. Symp. Secur. Privacy (EuroS&P)*, Jul. 2024, pp. 41–61.

[7] C. Wu, K. He, and J. Chen, "Liveness is not enough: Enhancing fingerprint authentication with behavioral biometrics to defeat puppet attacks," in *Proc. USENIX SEC*, 2020, pp. 2219–2236.

[8] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4685–4694.

[9] Z. Wu, Y. Cheng, S. Zhang, X. Ji, and W. Xu, "UniID: Spoofing face authentication system by universal identity," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2024, pp. 1–16.

[10] P. Markert, D. V. Bailey, M. Golla, M. Durmuth, and A. J. Aviv, "This PIN can be easily guessed: Analyzing the security of smartphone unlock PINs," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 286–303.

[11] D. V. Bailey, C. W. Munyendo, H. A. Dyer, M. Grant, P. Markert, and A. J. Aviv, "'Someone definitely used 000,': Strategies, performance, and user perception of novice smartphone-unlock PIN-guessers," in *Proc. Eur. Symp. Usable Secur.*, Oct. 2023, pp. 158–174.

[12] D. Wang, P. Wang, D. He, and Y. Tian, "Birthday, name and bifacial-security: Understanding passwords of Chinese web users," in *Proc. USENIX Secur.*, pp. 1537–1555.

[13] A. J. Aviv, J. T. Davin, F. Wolf, and R. Kuber, "Towards baselines for shoulder surfing on mobile authentication," in *Proc. 33rd Annu. Comput. Secur. Appl. Conf.*, Dec. 2017, pp. 486–498.

[14] A. J. Aviv, F. Wolf, and R. Kuber, "Comparing video based shoulder surfing with live simulation," in *Proc. 34th Annu. Comput. Secur. Appl. Conf.*, Dec. 2018, pp. 453–466.

[15] K. K. C., M. S. Hossain, and C. Haberfeld, "Exploring capacitive swipe gesture for user authentication using a new large dataset," *Comput. Secur.*, vol. 155, Aug. 2025, Art. no. 104475.

[16] Y. Zhai and M. Qiao, "Contrastive learning for continuous touch-based authentication," in *Proc. Int. Conf. Adv. Comput. Vis. Res. Appl. (ACVRA)*, May 2026, p. 72.

[17] Y. Wang and R. Hu, "TSCA: Enhancing smartphone security with a touchpoints-sensitive and context-aware model for touchscreen-based authentication," in *Proc. 28th Int. Conf. Comput. Supported Cooperat. Work Design (CSCWD)*, May 2025, pp. 1937–1942.

[18] C. Zhao, F. Gao, and Z. Shen, "AttAuth: An implicit authentication framework for smartphone users using multimodality data," *IEEE Internet Things J.*, vol. 11, no. 4, pp. 6928–6942, Feb. 2024.

[19] M. Kudo, T. Takahashi, I. Echizen, and H. Yamana, "ECommTouch: A benchmark dataset for touch-based continuous mobile device authentication for e-commerce," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Feb. 2025, pp. 140–147.

[20] Z. Shen, S. Li, X. Zhao, and J. Zou, "CT-auth: Capacitive touchscreen-based continuous authentication on smartphones," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 1, pp. 90–106, Jan. 2024.

[21] P. Delgado-Santos et al., "SwipeFormer: Transformers for mobile touchscreen biometrics," *Expert Syst. Appl.*, vol. 237, Mar. 2024, Art. no. 121537.

[22] C. Shen, Y. Zhang, X. Guan, and R. A. Maxion, "Performance analysis of touch-interaction behavior for active smartphone authentication," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 3, pp. 498–513, Mar. 2016.

[23] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 1, pp. 136–148, Jan. 2013.

[24] Z. Shen, S. Li, X. Zhao, and J. Zou, "IncreAuth: Incremental-learning-based behavioral biometric authentication on smartphones," *IEEE Internet Things J.*, vol. 11, no. 1, pp. 1589–1603, Jan. 2024.

[25] C. Wu et al., "It's all in the touch: Authenticating users with HOST gestures on multi-touch screen devices," *IEEE Trans. Mobile Comput.*, vol. 23, no. 10, pp. 10016–10030, Oct. 2024.

[26] H. Fereidooni et al., "AuthentiSense: A scalable behavioral biometrics authentication scheme using few-shot learning for mobile platforms," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2023, pp. 1–16.

[27] Y. Cai, Z. Zhang, and J. Gui, "Famos: Robust privacy-preserving authentication on payment apps via federated multi-modal contrastive learning," in *Proc. USENIX Secur.*, 2024, pp. 289–306.

[28] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 815–823.

[29] A. Mahfouz, H. Mostafa, T. M. Mahmoud, and A. Sharaf Eldin, "m2auth: A multimodal behavioral biometric authentication using feature-level fusion," *Neural Comput. Appl.*, vol. 36, no. 34, pp. 21781–21799, Dec. 2024.

[30] M. Kudo, T. Takahashi, and H. Yamana, "Touch-based continuous mobile device authentication using one-vs-one classification approach," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Feb. 2024, pp. 167–174.

[31] A. Mahfouz, A. Hamdy, M. A. Eldin, and T. M. Mahmoud, "b2auth: A contextual fine-grained behavioral biometric authentication framework for real-world deployment," *Pervas. Mobile Comput.*, vol. 99, Apr. 2024, Art. no. 101888.

[32] P. Aaby, M. V. Giuffrida, W. J. Buchanan, and Z. Tan, "An omnidirectional approach to touch-based continuous authentication," *Comput. Secur.*, vol. 128, May 2023, Art. no. 103146.

[33] I. Stylios, S. Chatzis, O. Thanou, and S. Kokolakis, "Continuous authentication with feature-level fusion of touch gestures and keystroke dynamics to solve security and usability issues," *Comput. Secur.*, vol. 132, Sep. 2023, Art. no. 103363.

[34] J. Chao, M. S. Hossain, and L. Lancor, "Swipe gestures for user authentication in smartphones," *J. Inf. Secur. Appl.*, vol. 74, May 2023, Art. no. 103450.

[35] A. Ray-Dowling, D. Hou, S. Schuckers, and A. Barbir, "Evaluating multi-modal mobile behavioral biometrics using public datasets," *Comput. Secur.*, vol. 121, Oct. 2022, Art. no. 102868.

[36] Z. Shen, S. Li, X. Zhao, and J. Zou, "MMAuth: A continuous authentication framework on smartphones using multiple modalities," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 1450–1465, 2022.

[37] M. Agrawal, P. Mehrotra, R. Kumar, and R. R. Shah, "GANTouch: An attack-resilient framework for touch-based continuous authentication system," *IEEE Trans. Biometrics, Behav., Identity Sci.*, vol. 4, no. 4, pp. 533–543, Oct. 2022.

[38] Z. DeRidder et al., "Continuous user authentication using machine learning and multi-finger mobile touch dynamics with a novel dataset," in *Proc. 9th Int. Conf. Soft Comput. Mach. Intell. (ISCMI)*, Nov. 2022, pp. 42–46.

[39] A. Z. Zaidi, C. Y. Chong, R. Parthiban, and A. S. Sadiq, "A framework of dynamic selection method for user classification in touch-based continuous mobile device authentication," *J. Inf. Secur. Appl.*, vol. 67, Jun. 2022, Art. no. 103217.

[40] M. Georgiev, S. Eberz, H. Turner, G. Lovisotto, and I. Martinovic, "Common evaluation pitfalls in touch-based authentication systems," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, May 2022, pp. 1049–1063.

[41] Ö. D. Incel et al., "DAKOTA: Sensor and touch screen-based continuous authentication on a mobile banking application," *IEEE Access*, vol. 9, pp. 38943–38960, 2021.

[42] Y. Ouadjer, M. Adnane, and N. Bouadjenek, "Feature importance evaluation of smartphone touch gestures for biometric authentication," in *Proc. 2nd Int. Workshop Hum.-Centric Smart Environ. Health Well-Being (IHSH)*, Feb. 2021, pp. 103–107.

[43] W. Li, W. Meng, and S. Furnell, "Exploring touch-based behavioral authentication on smartphone email applications in IoT-enabled smart cities," *Pattern Recognit. Lett.*, vol. 144, pp. 35–41, Apr. 2021.

[44] Y. Cheng et al., "Identifying child users via touchscreen interactions," *ACM Trans. Sensor Netw.*, vol. 16, no. 4, pp. 1–25, Nov. 2020.

[45] M. Abuhamad, T. Abuhmed, D. Mohaisen, and D. Nyang, "AUToSen: Deep-learning-based implicit continuous authentication using smartphone sensors," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5008–5020, Jun. 2020.

[46] A. Buriro, B. Crispo, and M. Conti, "AnswerAuth: A bimodal behavioral biometric-based user authentication scheme for smartphones," *J. Inf. Secur. Appl.*, vol. 44, pp. 89–103, Feb. 2019.

IEEE Proof

[47] W. Meng, W. Li, and L. Jiang, "SociAlauth: Designing touch behavioral smartphone user authentication based on social networking applications," in *ICT Systems Security and Privacy Protection*. Cham, Switzerland: Springer, 2019, pp. 180–193.
[48] Y. Yang, B. Guo, Z. Wang, M. Li, Z. Yu, and X. Zhou, "BehaveSense: Continuous authentication for security-sensitive mobile apps using behavioral biometrics," *Ad Hoc Netw.*, vol. 84, pp. 9–18, Mar. 2019.
[49] Z. Syed, J. Helmick, S. Banerjee, and B. Cukic, "Touch gesture-based authentication on mobile devices: The effects of user posture, device size, configuration, and inter-session variability," *J. Syst. Softw.*, vol. 149, pp. 158–173, Mar. 2019.
[50] J. Fierrez, A. Pozo, M. Martinez-Diaz, J. Galbally, and A. Morales, "Benchmarking touchscreen biometrics for mobile authentication," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 11, pp. 2720–2733, Nov. 2018.
[51] X. Wang, T. Yu, O. Mengshoel, and P. Tague, "Towards continuous and passive authentication across mobile devices: An empirical study," in *Proc. 10th ACM Conf. Secur. Privacy Wireless Mobile Netw.*, Jul. 2017, pp. 35–45.
[52] N. Palaskar, Z. Syed, S. Banerjee, and C. Tang, "Empirical techniques to detect and mitigate the effects of irrevocably evolving user profiles in touch-based authentication systems," in *Proc. IEEE 17th Int. Symp. High Assurance Syst. Eng. (HASE)*, Jan. 2016, pp. 9–16.
[53] U. Mahbub, S. Sarkar, V. M. Patel, and R. Chellappa, "Active user authentication for smartphones: A challenge data set and benchmark results," in *Proc. IEEE 8th Int. Conf. Biometrics Theory, Appl. Syst. (BTAS)*, Sep. 2016, pp. 1–8.
[54] M. Temper, S. Tjoa, and M. Kaiser, "Touch to authenticate—Continuous biometric authentication on mobile devices," in *Proc. 1st Int. Conf. Softw. Secur. Assurance (ICSSA)*, Jul. 2015, pp. 30–35.
[55] M. Antal, Z. Bokor, and L. Z. Szabó, "Information revealed from scrolling interactions on mobile devices," *Pattern Recognit. Lett.*, vol. 56, pp. 7–13, Apr. 2015.
[56] T. Feng, J. Yang, Z. Yan, E. M. Tapia, and W. Shi, "TIPS: Context-aware implicit user identification using touch screen in uncontrolled environments," in *Proc. 15th Workshop Mobile Comput. Syst. Appl.*, Feb. 2014, pp. 1–6.
[57] H. Xu, Y. Zhou, and M. R. Lyu, "Towards continuous and passive authentication via touch biometrics: An experimental study on smartphones," in *Proc. SOUPS*, 2014, pp. 187–198.
[58] A. Serwadda, V. V. Phoha, and Z. Wang, "Which verifiers work: A benchmark evaluation of touch-based authentication algorithms," in *Proc. IEEE 6th Int. Conf. Biometrics, Theory, Appl. Syst. (BTAS)*, Sep. 2013, pp. 1–8.
[59] X. Zhao, T. Feng, and W. Shi, "Continuous mobile authentication using a novel graphic touch gesture feature," in *Proc. IEEE 6th Int. Conf. Biometrics, Theory, Appl. Syst. (BTAS)*, Sep. 2013, pp. 1–6.
[60] L. Li, X. Zhao, and G. Xue, "Unobservable re-authentication for smartphones," in *Proc. NDSS*, 2013, pp. 57–59.
[61] C. Bo, L. Zhang, and X.-Y. Li, "Silentsense: Silent user identification via touch and movement behavioral biometrics," in *Proc. ACM MobiCom*, 2013, pp. 187–190.
[62] H. Khan, U. Hengartner, and D. Vogel, "Targeted mimicry attacks on touch input based implicit authentication schemes," in *Proc. 14th Annu. Int. Conf. Mobile Syst., Appl.*, Services, Jun. 2016, pp. 387–398.
[63] P. Negi, P. Sharma, V. S. Jain, and B. Bahmani, "K-means++ vs. behavioral biometrics: One loop to rule them all," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018, pp. 1–13.
[64] A. Serwadda and V. V. Phoha, "When kids' toys breach mobile phone security," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 599–610.
[65] B. Z. H. Zhao, H. J. Asghar, and M. A. Kaafar, "On the resilience of biometric authentication systems against random inputs," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2020, pp. 1–18.
[66] E. Korkes, C. W. Munyendo, A. Isaac, V. Hennemann, and A. J. Aviv, "'I'm going to try her birthday': Investigating how friends guess each Other's smartphone unlock PINs in the lab," in *Proc. Eur. Symp. Usable Secur.*, Sep. 2024, pp. 220–234.
[67] Y. Abdelrahman, M. Khamis, S. Schneegass, and F. Alt, "Stay cool! Understanding thermal attacks on mobile-based user authentication," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, May 2017, pp. 3751–3763.
[68] A. Acien, A. Morales, J. Fierrez, R. Vera-Rodriguez, and O. Delgado-Mohatar, "BeCAPTCHA: Behavioral bot detection using touchscreen and mobile sensors benchmarked on HuMIdb," *Eng. Appl. Artif. Intell.*, vol. 98, Feb. 2021, Art. no. 104058.
[69] T. Van hamme, G. Garofalo, E. A. Rúa, D. Preuveneers, and W. Joosen, "A novel evaluation framework for biometric security: Assessing guessing difficulty as a metric," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 8369–8384, 2024.



Jingyu Yao received the M.S. degree in software engineering from the School of Computer Science, Shaanxi Normal University, Xi'an, China. He is currently pursuing the Ph.D. degree with the College of Cyber Science, Nankai University, Tianjin, China. His research interests include applied cryptography, implicit authentication, and risk-based authentication.



Ding Wang received the Ph.D. degree in information security from Peking University in 2017. He was supported by the "Boya Postdoctoral Fellowship" at Peking University from 2017 to 2019. Currently, he is a Full Professor with Nankai University. As the first author (or a corresponding author), he has published more than 140 papers at venues like IEEE S&P, ACM CCS, NDSS, USENIX Security, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, and IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY. His research has been reported by over 200 media outlets, such as The Wall Street Journal, Daily Mail, Forbes, IEEE Spectrum, and Communications of the ACM; it appeared in the Elsevier 2017 "Article Selection Celebrating Computer Science Research in China," and resulted in the revision of the authentication guideline NIST SP800-63. His research interests focus on passwords, authentication, and provable security. He has been involved in the community as the PC Chair/a TPC Member for over 60 international conferences, such as USENIX Security 2027, NDSS 2023–2025, ACM CCS 2022, PETS 2022–2024, ACSAC 2020–2024, FC 2025–2027, ACM AsiaCCS 2027/2025/2022/2021, IFIP SEC 2018–2026, ICICS 2018–2026, and SPNCE 2020–2022. He has received the ACM China Outstanding Doctoral Dissertation Award, the Best Paper Award at INSCRYPT 2018, the First Prize of the Cryptologic Innovation Award of China Association for Cryptologic Research, and the First Prize of the Natural Science Award of the Ministry of Education.

IEEE Proof