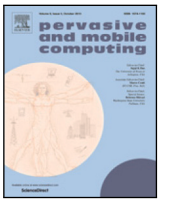
Pervasive and Mobile Computing 101 (2024) 101922

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc



MotionID: Towards practical behavioral biometrics-based implicit user authentication on smartphones

Mohsen Ali Alawami [a], Tamer Abuhmed [b], Mohammed Abuhamad [c], Hyoungshick Kim [b,*]

[a] Division of Computer Engineering, Hankuk University of Foreign Studies, Mohyeon-eup, Yongin-si, Gyeonggi-do, 17035, South Korea
[b] Sungkyunkwan University, 2066, Seobu-ro, Jangan-gu, Suwon-si, Gyeonggi-do, 16419, South Korea
[c] Loyola University Chicago, 308 Doyle Center, Lake Shore Campus 1052 W Loyola Ave., Chicago, IL, 60626, United States

ARTICLE INFO

Keywords:
User authentication
Behavioral biometrics
Anomaly detection
Touch dynamics
Deep learning
Smartphone

ABSTRACT

Traditional one-time authentication mechanisms cannot authenticate smartphone users' identities throughout the session — the concept of using behavioral-based biometrics captured by the built-in motion sensors and touch data is a candidate to solve this issue. Many studies proposed solutions for behavioral-based continuous authentication; however, they are still far from practicality and generality for real-world usage. To date, no commercially deployed implicit user authentication scheme exists because most of those solutions were designed to improve detection accuracy without addressing real-world deployment requirements. To bridge this gap, we tackle the limitations of existing schemes and reach towards developing a more practical implicit authentication scheme, dubbed MotionID, based on a one-class detector using behavioral data from motion sensors when users touch their smartphones. Compared with previous studies, our work addresses the following challenges: ① *Global mobile average* to dynamically adjust the sampling rate for sensors on any device and mitigate the impact of using sensors' fixed sampling rate; ② *Over-all-apps* to authenticate a user across all the mobile applications, not only on-specific application; ③ *Single-device-evaluation* to measure the performance with multiple users' (i.e., genuine users and imposters) data collected from the same device; ④ *Rapid authentication* to quickly identify users' identities using a few samples collected within short durations of touching (1–5 s) the device; ⑤ *Unconditional settings* to collect sensor data from real-world smartphone usage rather than a laboratory study. To show the feasibility of MotionID for those challenges, we evaluated the performance of MotionID with ten users' motion sensor data on five different smartphones under various settings. Our results show the impracticality of using a *fixed sampling rate* across devices that most previous studies have adopted. MotionID is able to authenticate users with an F1-score up to 98.5% for some devices under practical requirements and an F1-score up to roughly 90% when considering the drift concept and rapid authentication settings. Finally, we investigate time efficiency, power consumption, and memory usage considerations to examine the practicality of MotionID.

1. Introduction

Smartphones have become crucial for daily activities, including mobile banking, communications, and storing sensitive personal data. Therefore, secure and usable authentication methods are essential to prevent unauthorized access. Existing authentication

* Corresponding author.
E-mail addresses: mohsencomm@hufs.ac.kr (M.A. Alawami), tamer@skku.edu (T. Abuhmed), mabuhamad@luc.edu (M. Abuhamad), hyoung@skku.edu (H. Kim).

https://doi.org/10.1016/j.pmcj.2024.101922
Received 31 March 2023; Received in revised form 28 November 2023; Accepted 1 April 2024
Available online 3 April 2024

mechanisms on smartphones are point-of-entry methods, in which the user should authenticate only once during initial login using either knowledge-based approaches (*e.g.,* PIN, password, or pattern) [1–5] or physiological biometrics techniques (*e.g.,* face or fingerprints) [6–9]. Several previous studies have shown that PINs and lock patterns are susceptible to attacks by highly accurate camera recording and human observations (*e.g.,* shoulder surfing and smudge attacks [10–15]) that can steal a victim's credentials [16–18]. Moreover, attackers can use presentation attacks against fingerprint authentication by spoofing the system with counterfeit crafts, such as gummy fingers [9,19–22]. Most importantly, these methods fail to offer security and access control beyond the point of entry, *i.e.,* continuously authenticating the user throughout the session. In contrast, implicit continuous authentication solutions based on behavioral biometrics recorded by the built-in sensors would tackle this issue [23,24]. Many existing studies use either environmental characteristics (e.g., Wi-Fi and light [25–30]) or user's behavioral biometrics to run a background process continuously and implicitly perform a transparent authentication, *e.g.,* motion sensors [25,31–34], gait [35–37], touch dynamics [38–42], keystroke dynamics [43–48], and voice [49–51].

To our knowledge, none of the smartphone vendors or research communities have successfully deployed a practical implicit behavioral-based continuous authentication scheme, mainly due to the many issues related to smartphone deployment requirements and constraints in real-world applications. Existing behavioral-based implicit authentication studies hold some limitations and impractical assumptions that we address in this research, and we summarize them as follows.

① Existing sensory-based authentication schemes usually assume a *fixed sampling rate* for all sensors/devices when collecting sensory data. Some examples (but not limited to) are as follows. The work in [32] collected sensory data from diverse Android smartphones with 4.4 or higher versions, but they fixed the sampling rate to 50 Hz for all devices. The work in [24] conducted different types of experiments (*e.g.,* free-form usage, lab experiments, and attacker usage) and collected motion sensor datasets using different devices such as smartphones and smartwatches, but using the same sampling rate of 50 Hz. The column "Sampling rate" shown in Table 1 describes that the majority of recent behavioral-based authentication studies that conducted their experiments on various motion sensors of smartphones have used *fixed* value of sampling rate (some of the studies did not mention the used sampling rate (Unknown)). However, after we conducted extensive experimentation on various sensors and smartphones (see Table 3), we observed that sensors' sampling rates could differ significantly on different devices and across same-device sensors even when using the same sensor query APIs. Since user behaviors are directly related to the amount of sensory data collected within a specific authentication time, these differences in sampling rates are important to address for the practical evaluation of the model.

② Most implicit authentication solutions require users to operate within a specific application and perform pre-defined actions (*e.g.,* texting, tapping, swiping, and gaming) to collect motion sensor and coordinates of the screen points. For example, the work in [52] conducted behavioral-based authentication by asking different volunteers to browse a few items on a specific shopping application that they developed for collecting site browsing behaviors alongside motion sensor data on smartphones. In addition, other works have collected motion sensor data by asking participants to conduct events only on specific applications such as taps/enters 10-digits on dialing mobile application [32], authenticating gesture-typing interactions by writing specific words using smartphone's keypad [41], swiping finger movements collected in the up, down, left, and right direction on a special-developed application [39], performing swipes up and down, and taps during matching logo of cars with their names on a game application [53], and treating a memory-testing game to capture different types of gestures that describe the swiping behaviors on developed "BrainRun" game application [54]. Although the datasets of the above studies are collected under restricted conditions (*i.e.,* within special-designed mobile applications and pre-defined actions), they are no longer programmatically accessible or obtainable in newer versions of operating systems due to security restrictions. This adds additional challenges in collecting such data to effectively authenticate a user who touches multiple applications or interacts with the entire phone (*i.e.,* cross-application use). Therefore, existing solutions are not suitable for real-world usage. To tackle this issue, we propose collecting sensory data during touching/usage times when a user interacts *over-all-apps* and without any pre-defined events.

③ Previous studies evaluated authentication systems using data collected from different users on various devices (*i.e.,* cross-device-evaluation). In detail, during the enrollment phase, they train the models using data collected from a single user on a specific device (*i.e.,* the legitimate user), then they use imposters' data (*i.e.,* attackers), which is mainly collected from other devices for the testing phase [26,54–58]. We argue that this limits the practicality of such methods, as other devices' data significantly differ due to variations in sensors' quality and sampling rates. Moreover, cross-device evaluation could potentially result in an overestimation of the performance of authentication models, as it becomes unclear whether the classification performance is due to variations in user behavior or differences in the sensory data of the devices. Therefore, evaluating the performance of authentication models across devices could be misleading, as it is difficult to determine the exact cause of variations in the classification results. We propose a more realistic evaluation using legitimate and imposter users' data from the same device (*i.e., single-device-evaluation*).

④ In terms of authentication frequency, implicit authentication schemes are required to quickly detect the identity of the current user. Existing methods show the feasibility of authenticating users using data samples collected for long periods (*e.g.,* hours or days) [38,59–63]. In other words, they evaluated the models using a large amount of sensory data samples that mainly need a long time to be collected from users, which leads to delayed authentication results in practice. However, real-world scenarios show that an imposter needs just a few seconds to exploit unauthorized access to private information on the victim's smartphone. Therefore, we propose a rapid authentication feature by investigating the effectiveness of trained models when testing them using a few data samples collected during short touch durations (*e.g.,* 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, or 5 s).

⑤ Another critical shortcoming of current methods is conducting experiments in controlled settings (*i.e.,* laboratory environments) in which samples are collected from participants using pre-defined conditions (*e.g.,* sitting, standing, or walking), handling/postures (*e.g.,* on hand or on the table), or specific action/use (*e.g.,* texting, reading, or browsing). For example, three publicly available and widely cited behavioral-based authentication datasets collected from smartphones are HMOG [72],

Table 1
Summary of behavioral-based user authentication related works (Motion sensors and Touch data).

Study	Modalities	Sensors	Methods	Unconditional settings	Sampling rate (amount)	OAA	SDE	OCC	Rapid Auth.
[55]	MS	Acc, Gyr	CNN, OCSVM	✗	Fixed (100 Hz)	✗	✗	✓	✗
[31]	MS	Acc, Gyr	Ten classifiers	✗	Fixed (20 Hz)	✗	✗	✓	✗
[56]	MS	Acc, Gyr, Mag	Siamese CNN	✗	Fixed (25 & 100 Hz)	✗	✗	✓	✗
[32]	MS + To	Acc, Gyr, To	MLP	✗	Fixed (50 Hz)	✗	✗	✓	✓
[23]	MS + To	Acc, Gyr, Mag, Or, To	HMM	✗	Fixed (100 Hz)	✗	✗	✓	✗
[52]	MS + To	Acc, Gyr, To	LSTM	✗	Fixed (50 & 100 Hz)	✗	✓	✗	✗
[24]	MS	Acc, Mag, Or, To, light	KRR	both	Fixed (50 Hz)	✗	✗	✗	✗
[64]	MS	Acc, Gyr	OCSVM	✗	Fixed (100 Hz)	✗	✗	✓	✗
[59]	MS	Acc, Gyr, Grav	SVM	✗	Fixed (50 Hz)	✗	✗	✗	✗
[26]	MS + To	Acc, Gyr, Mag, To	Multilayer LSTM	✓	Fixed (32 & 64 Hz)	✓	✗	✗	✓
[57]	MS	Acc, Gyr	CNN	✗	Fixed (100 Hz)	✗	✗	✓	✓
[58]	MS + To	Acc, Gyr, Mag, grav, To	LSTM	✗	Fixed (200 Hz)	✗	✗	✗	✗
[60]	MS	Acc, Gyr, Grav	CNN + SVM	✗	Fixed (50 Hz)	✗	✗	✗	✗
[65]	MS	Acc, Gyr, Mag	Fusion	✗	Fixed (100 Hz)	✗	✗	✓	✗
[61]	MS	Acc, Gyr, Mag, Grav, Or	Siamese CNN	✗	Fixed (100 Hz)	✗	✗	✗	✗
[62]	MS	Acc, Gyr, Mag	DeepConvLSTM	✗	Fixed (100 Hz)	✗	✗	✗	✗
[38]	MS + To	Acc, Gyr, To	KNN	✗	Fixed (100 Hz)	✗	✓	✗	✗
[39]	To	Swipes	Five classifiers	✗	Unknown	✗	✗	✗	✗
[53]	MS + To	Acc, Gyr, Swipes, taps.	OCSVM	✗	Unknown	✗	✗	✓	✗
[66]	To	Swipes and taps	Random Forest	✗	Unknown	✗	✗	✓	✗
[41]	MS + To	Acc, Gyr, keyboard gestures	LR, NB, RF	✗	Fixed (50 Hz)	✗	✓	✗	✗
[63]	To	Web browsing gestures	PSO-RBFN	✗	Unknown	✗	✗	✓	✗
[67]	MS	Acc, Gyro	DWT , LSTM	✗	Fixed (200 Hz)	✗	✓	✗	✓
[68]	MS	Acc	NA	✗	Unknown	✗	✗	✗	✗
[69]	MS	Acc	SVM	✗	Fixed (80 Hz)	✗	✓	✗	✓
[70]	MS	Acc	LSTM	✗	Fixed (50 Hz)	✗	✗	✗	✗
[71]	MS	Acc, Gyr, Mag	SVM	✗	Fixed (50 Hz)	✗	✗	✓	✗
[72]	MS + To	Acc, Gyro, Mag, To	OCSVM	✗	Fixed (100 Hz)	✗	✗	✓	✗
[54]	MS + To	Acc, Gyro, Mag, To	NA	✗	Fixed (100 Hz)	✗	✗	✗	✗
[73]	To	Acc, Gyro	MLP	✗	Unknown	✗	✗	✗	✗
This work	MS + To	Acc, Elev, Grav, Gyr, Mag, To	LSTM-AE, LOF, OCSVM, IF, EE	✓	Global, dynamic	✓	✓	✓	✓

MS: Motion sensors, To: Touch data, Acc: Accelerometer, Elev: Elevation, Gyr: Gyroscope, Grav: Gravity, Mag: Magnetometer, Or: Orientation.
OAA: Over-all-apps. SDE: Single-device-evaluation. OCC: One-class classification

BrainRun [54], and touchalytics [40]. These datasets were used to evaluate the authentication performance by many studies in the literature. However, these datasets are collected in a controlled lab environment, *e.g.,* (1) for the HMOG dataset, users were asked to answer specific questions in eight sessions (4 sessions at sitting and 4 sessions at walking); (2) for the BrianRun dataset, users were asked to play specific games with different stages and scenarios to collect gestures and sensory data; (3) for the touchalytics dataset, users were asked to read three documents, answer questions after reading each document, and compare images to spot differences in pairs of similar images to collect touch data. In our work, we open up the ability to operate with unconditional settings where users freely use smartphones without any pre-determined tasks, positions, postures, or activities for authentication. We asked participants to conduct experiments and interact with smartphones as their usual daily usage.

To the best of our knowledge, none of the current literature schemes has investigated the effectiveness of authentication models while taking into account the crucial limitations mentioned above. Therefore, we are the first to address these limitations and attempt to bridge the gap towards developing an implicit continuous authentication service that meets the practical requirements necessary for real-world applications. To achieve this goal, we have identified four research questions that we aim to answer in order to support the deployment of authentication models on smartphones for real-world usage, as follows.

- **RQ1:** Does the assumption of using a fixed sampling rate over all devices and sensors hold in practice?
- **RQ2:** How much behavioral-based continuous authentication is effective when considering proposed practical advantages, such as considering only sensory data, when a user globally interacts smartphone (over-all-apps), and when using a single-device-evaluation-based approach?
- **RQ3:** Does MotionID effective when using short interaction durations for high-frequency authentication?
- **RQ4:** How effective is MotionID under certain considerations, such as time efficiency, power consumption, and memory usage?

Our work answers these questions and explores the practical deployment of behavioral biometrics towards continuous authentication. We summarize our contributions as follows.

1. We propose a behavioral-based implicit continuous authentication system, MotionID, using sensory data and considering real-world deployment requirements.
2. Unlike existing studies, MotionID presents the concept of a *global mobile average* to dynamically compute average sampling rates of sensory data on every customer-grade smartphone, which combats the variations of sampling data across sensors/devices and reduces the impact of a fixed sampling rate assumption. In addition, MotionID presents advantages for practical authentication usage *e.g., over-all-apps*, *single-device-evaluation*, *unconditional settings*, and *rapid authentication*.
3. We evaluate MotionID through multiple dimensions and experiment settings to meet real-world requirements.
4. We investigate the impact of using fixed sampling rates across sensors/devices, and demonstrate the variations observed with such assumption across various built-in sensors within a device. We propose *global mobile average* technique to overcome such variations.
5. We investigate the effects of using various/diverse devices and hyperparameters settings to build MotionID. Moreover, we analyze the performance of MotionID using multiple aspects, such as authentication frequency, data collection durations, and temporal effect on authenticators. Our results show that MotionID achieves F1-scores up to 98.5% for some devices in practical settings. We also demonstrate the practicality of MotionID in terms of response time efficiency, power consumption, and memory usage.

The remainder of this paper is structured as follows. In Section 2, we review related works. We present the design overview of MotionID in Section 3. Section 4 demonstrates the methodology details of MotionID. In Section 5, we proceed with the details of the one-class detectors used in MotionID. Section 6 shows experiment settings and results from various evaluation approaches. We finally provide our conclusion in Section 7.

2. Related work

To avoid damaging smartphones or revealing private information by imposters, continuous authentication (CA) of a user using the device is required to support the user's privacy and security. Existing studies on user authentication are classified into categories such as classical-based methods (including PIN, passwords, patterns, and fingerprint) and biometrics behavioral-based methods where a background process continuously captures the user's behaviors (e.g., motion sensors, touchscreen gestures, keystrokes dynamics, gait, light, and voice) on a device to perform implicit authentications [29,74–79].

Since this work focuses on behavioral-based continuous authentication using motion sensors and touchscreen timestamps, we explored existing related works and provided a summary and a comparison between them, including ours, in terms of proposed real-world requirements, as shown in Table 1. We highlight the most recent and related works as follows. Yantao et al. [55] proposed a two-stream CNN-based continuous authentication, named SCANet, from frequency and time domain data and selected the top 25 features to train the OCSVM classifier. SCANet has been applied to two datasets and experimentally achieved 90.04% accuracy when a user authenticated within 3s. Similar work has been done using smartphone sensors, Shen et al. [31] extracted features in three domains (time, frequency, and wavelet) and provided empirical evaluation and sensor-behavior analysis using a dataset containing five actions and smartphone placements. Their experiment results show EER values range from 2.21% to 28.22% based on the kind of action and placement. Centeno et al. [56] exploited the Siamese-CNN network to learn user behaviors from data of three sensors (Acc, Gyr, Mag) when the volunteers conducted three tasks (reading, writing, map navigation) in 24 sessions on smartphones and achieved accuracy up to 97.8% using on-class SVM.

Buriro et al. [32] proposed an authentication scheme, named DialerAuth, using both motion and touch data when a user dials with the 10-digit numbers and hand's movements. DialerAuth studied the invisible timings and the phone micro-movements and achieved a TAR of 85.77% using an unsupervised MLP classifier. Shen et al. [23] presented a performance analysis for authentication to investigate the applicability and reliability of motion sensors under various operational scenarios. The evaluation was done using HMM one-class classifier on a large dataset and achieved an average EER of 4.74%. Amini et al. [52] addressed the issue of remaining a user logged in for hours or weeks when doing online actions and provided a re-authentication system, DeepAuth, using LSTM to secure users on a mobile application. DeepAuth can re-authenticate a user with 96.70% accuracy within only 20 s. Lee et al. [24] proposed a context-based continuous authentication scheme by leveraging sensors of multiple devices and several machine learning algorithms and showed results of an average accuracy of 98.1%. Yantao et al. [64] presented an authentication system, SensorAuth, to understand user behaviors using motion sensors and five augmentation approaches (permutation, sampling, scaling, cropping, and jittering) in order to add more samples to the training data. SensorAuth extracted time and frequency features from the augmented dataset and applied OCSVM to authenticate users with an average EER of 4.66% within 5 s.

Zhu et al. [59] present an authentication system, RiskCog, using motion sensors that is flexible to the device placement, does not require user movements, and provides offline verification with a very short latency. Abuhamad et al. [26] presented an unconditional implicit authentication system, AutoSen, using LSTM deep learning with motion sensors. AutoSen was implemented when users freely used their devices for several days and the binary classifiers achieved authentication performance F1 of around 98% using half-second and one-second of data samples.

Li et al. [57] presented an authentication system, DeFFusion, using the accelerometer and gyroscope sensors on smartphones and exploited CNN for deep feature fusion. DeFFusion was evaluated using a one-class SVM classifier with several impacts, such as training data size, time window, time efficiency, and unseen users, and achieved an average EER of 1.00% within 5 s. Stragapede et al. [58] conducted a comparative analysis of multimodal traits of behavioral biometrics while users doing activities

on smartphones such a scrolling, drawing numbers, typing, and tapping on the screen. Their experiments were evaluated on a public dataset, HuMiDb[1] using LSTM networks, and the fusion of modalities showed EER results ranging from 4% to 9% in a 3-second duration. Zhu et al. [60] proposed a hybrid deep-learning model by fusion of a variational mode decomposition (VMD), Tri-training semi-supervised methods, and combined CNN-OCSVM network for effective user authentication, which showed an authentication accuracy of 95.01%. Mingming et al. [65] proposed an authentication system by combining two stages of feature construction: manual statistical features and a three-channel matrix fed into the deep learning model. The system evaluated using two public datasets (HMOG[2] and BrainRun[3]) and four one-class classifiers.

Dybczak et al. [61] proposed a continuous authentication system to record sensory data of hand movement when using the smartphone using Siamese neural networks and their results of experiments on five smartphones showed an average accuracy of 85.5%. Mekruksavanich et al. [62] introduced a framework of continuous authentication, DeepAuthen, to identify physical activity patterns of users measured from three motion sensors and the fed to deep learning network named DeepConvLSTM. They evaluated the DeepAuthen system using three public datasets and achieved an average EER of 8%. Zhang et al. [38] proposed an authentication system, TouchID, using touch gestures and motion sensors to study the finger's movements when performing touch gestures under four different typologies. Ali et al. [39] presented a behavior-based authentication system using swipes' data, extracted optimal features, and then evaluated models using five machine learning classifiers and archived average F1 scores ranging from 87.8% to 95.70%. Garbuz et al. [53] analyzed touch interactions of users on smartphones such as vertical swipes up and down and taps alongside sensory data of three sensors where one-class SVM was used to evaluate the system. Syed et al. [66] demonstrated the impact of configurations such as device size and user's posture on the performance of touch-based authentication schemes. The author released a new dataset collected on four devices using three different postures from 31 subjects each of them has conducted at least 8 sessions. Smith-Creasey et al. [41] proposed an authentication system based on gesture-typing at a word-independent level. They collected a dataset in sitting, standing, and walking situations and evaluated the system using three classifiers to achieve an average EER of up to 3.58%. Lastly, Meng et al. [63] considered authenticating users in a specific application, such as a web browser, and proposed a TouchWB system with 21 gesture-based features from web browsing. Huh et al. [80] addressed one of the real-world deployment constraints which is ensuring low bit error rates over time when using keystroke dynamics for continuous authentication schemes on smartphones. In detail, they focused on developing two feature engineering techniques from sensory data (accelerometer and gyroscope) and their correlations with keystroke dynamics of users' behaviors to improve authentication accuracy.

Lastly, some studies have exploited location information to develop secure authentication schemes on smartphones. Alawami et al. [30] has proposed a fine-grained indoor unlocking system on the smartphone by exploiting Wi-Fi and light signatures seen at each user's location. By doing so, a user can build his trusted location's profile, train a user-specific model, and automatically get authenticated to lock/unlock her smartphone whenever resides within the trusted location. Cho et al. [81] focused on using location information to explore usability and security requirements for designing location-based authentication. In detail, they conducted an interview study with 18 participants to understand users' perceptions and expectations of location-based authentication. Based on design requirements identified from the study, they developed an Android application for screen unlock of smartphones using Wi-Fi RSSI readings and achieved low error rates (FAR and FRR) that meet real-world deployments.

Our work contributes to the literature as follows. (1) Existing studies did not give a comprehensive view of the main requirements and key constraints that are required for developing a practical authentication system for real-world usage. We comprehensively address and analyze the evaluation of an authentication system considering these key limitations as they are crucial to understand the system's applicability and security for real-world applications. (2) Most of the existing work focused on improving authentication accuracy and neglected the other requirements and aspects, leaving a gap between current research and practical deployment needs in authentication security. We extensively analyze how much every requirement can affect the accuracy of authentication and investigate usability measurements (*e.g.,* authentication time, power consumption, and memory usage) for deploying a realistic authentication system. Note that since there is no directly related work conducted as MotionID (*i.e.,* with the same practical requirements that we addressed in this work), we could not directly compare MotionID with previous studies. However, we list most related studies with comparable items in Table 1 to show the differences and advantages of our work compared to others. Additionally, in this section, we report the authentication performance (*i.e.,* accuracy, F1-score, or EER) for each previous study for fair "*indirect*" comparison.

3. MotionID system design

This section introduces an overview of the MotionID system, its architecture, and design challenges.

3.1. Overview

We propose MotionID, a practical implicit continuous authentication system using behavioral biometrics on smartphones. MotionID aims to address limitations in the existing authentication works and move towards practical schemes. Unlike existing

[1] https://github.com/BiDAlab/HuMIdb
[2] https://hmog-dataset.github.io/hmog/
[3] https://www.mdpi.com/2306-5729/4/2/60

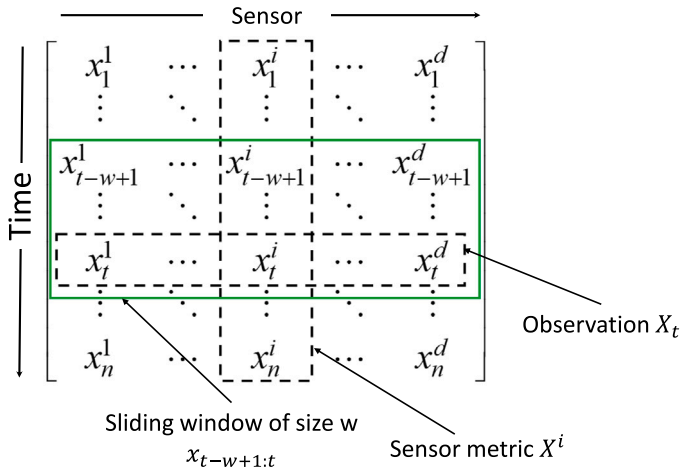


Fig. 1. Data formulation of time-series sensor data.

works, MotionID collects multivariate time-series readings (MTS) from built-in sensors in *unconditional* and *freely* manner when a user globally touches the entire phone environment *(over-all-apps)*, not only on a specific application. In other words, we rely on data collected when a user interacts with the smartphone that includes: (1) *screen touches timestamps (Ts) i.e.,* timings data when users globally touch any application during smartphone usage, (2) *sensory data i.e.,* accelerometer, elevation, gravity, gyroscope, and magnetometer, during touching periods. Table 2 shows and explains the list of all symbols used in the work.

Since modern smartphones are equipped with several motion sensors that have a fast response and can generate huge discrete data (*i.e.,* numbers) in a very short time (*e.g.,* milliseconds) and at various coordinates (*e.g.,* x, y, and z), we can simulate user-touching behaviors on smartphones using discrete data collected from motion sensors. Although we consider five motion sensors for collecting motion data from a user while touching a smartphone, the formulations of these sensory data are different. Also, the number of sensors available out of these five sensors changes from one smartphone to another (*i.e.,* some phones provide all five sensors while other phones provide only some of them — this depends on the phone grade/brand and vendor). In general, we describe the data dimensions provided by each sensor of the five motion sensors we used in this work as follows. Specifically, each accelerometer reading is a 3-D vector $Ac \in \Re^3$ of x, y, and z values that represent the phone coordinates. The elevation sensor provides a 1-D vector $El \in \Re^1$ of x values that represent the altitude of an object above a fixed level. The gravity sensor provides a 3-D vector $Gr \in \Re^3$ of x, y, and z values that measure the direction and intensity of gravity. The gyroscope sensor provides a vector $Gy \in \Re^3$ for angular rotation in radians per second along the axes. The magnetometer sensor provides a 3-D vector $Mg \in \Re^3$ of x, y, and z values that are used to report the magnetic field in microtesla.

We collect a five-sensors data set (AcElGrGyMg) alongside touching timestamps *(Ts)* to capture the behavioral patterns of users on smartphones. During each active/touching period on the screen, our Android collector application simultaneously collects sensory data from all motion sensors, as well as touch timestamps, and then concatenates them horizontally. The total dimension of the data collected on a specific smartphone (*i.e.,* columns) is fixed to the number of detected sensors and their dimensions on each smartphone. For example, if a smartphone is equipped with all five mentioned sensors ($k = 5$), then the total horizontal dimension (*i.e.,* columns) is $d_{total} = 13$ (where $d_{Ac} = 3$, $d_{El} = 1$, $d_{Gr} = 3$, $d_{Gy} = 3$, $d_{Mg} = 3$). Sometimes, we found that some smartphones (*e.g.,* Samsung Wide4 and LG LM-Q510N) do not have an Elevation sensor, and hence the total horizontal dimension of data provided by motion sensors is $k = 12$, and so on. Note that, the vertical dimension of sensory data (*i.e.,* number of rows) collected from a smartphone is a time-varying factor, which means that it increases as touching time increases. Based on this data formulation description, our data collector application frequently creates a matrix of sensory time-series data whenever a user interacts with and touches the screen of a smartphone. These data matrices have dimensions as follows: the number of columns is d representing the total dimension of the k detected motion sensors on a smartphone. However, the number of rows represents the number of data samples collected from the k detected motion sensors during the time consumed for touching.

After describing data matrices and their dimensions, we now expand the description of data formulation of the time-series sensors. As mentioned above, when a user touches the smartphone's screen for a period of time (*e.g.,* one touch period T_p is equal for a minute), the data collector application that we developed will generate a matrix of time-series sensory data collected from all motion sensors (AcElGrGyMg) detected on that smartphone. This process will be repeated as long as the user keeps touching the screen and then all matrices will be vertically concatenated to create one time-series dataframe collected during a longer time interval $T = (t_0, t_1, t_2, \ldots, t_n)$. Fig. 1 shows the data formulation of time-series sensory data that have two coordinates: (1) Time-based dimension that represents the number of rows of sensory data collected during several touch periods. For example, a user touches a smartphone's screen for 10 consecutive touch periods, each of which has a time length of about a minute, and collects 1000 rows of sensory readings. Thus, the total number of rows for that data matrix will be 10000 samples. (2) Sensor-based dimension that represents the value d, which is the number of detected sensors k multiplied by the number of the values of each sensor. If all five

Table 2
List of symbols used in the paper and their interpretations.

Symbol	Interpretation
w	Sliding window size that we used for smoothing process (removing outliers and cleaning raw sensory data).
X_t	One row of readings from all targeted sensors on a smartphone at a specific time t. (see Fig. 1).
X^i	One column of readings from only one specific sensor (i), but over all time stamps. (see Fig. 1).
AcElGrGyMg	Ac: Accelerometer, El: Elevation, Gr: Gravity, Gy: Gyroscope, Mg: Manometer
Ψ	Number of users (or devices) that were used to create the dataset.
Φ	Number of data extents DEs = $(DE_1, DE_2, \ldots, DE_n)$ (i.e., Data Pieces from all sensors) that collected over several intervals $T = (t_0, t_1, t_2, \ldots, t_n)$ from the same user (see Section 4.2).
k	Number of sensors considered per each device.
d	Number of dimensions for all detected motion sensors on a smartphone (*e.g.,* d of the five AcElGrGyMg sensors = 13)
ρ	Number of touch periods $(Tp_1, Tp_2, \ldots, Tp_\rho)$ occur during each data extent (DE). (see Fig. 3).
Ts	The exact timestamp value that is detected when a user touches the smartphone's screen. The timestamps are collected alongside sensor readings.
Tp	The touch period represents the timestamps and sensory readings while a user keeps touching the phone for a period (*e.g.,* a minute) continuously. (see Fig. 3).
df	DataFrame represents the matrix form of sensory data collected for a certain touch period from a specific sensor.
L	Size (i.e., number of rows) of each dataframe (df) of each sensor. (see Fig. 3).
DB	DataBlock represents dataframes collected from a single sensor for consecutive touch periods (see Fig. 3).
DE	DataExtent represents dataframes collected from k sensors for consecutive touch periods (see Fig. 3 and Fig. 4).
DS	DataSegment represents all DEs pieces of sensory data that were collected during several time intervals of touching the phone, but only for one user and on one smartphone.
Q	Indicates to Global Mobile Average (see Section 4.2).

motion sensors are detected on a smartphone (*i.e.,* $k = 5$), then d =13 (*i.e.,* number of columns). By looking again at Fig. 1, we denote each row of the sensory data matrix as $X_t = x_t^1, x_t^i, \ldots, x_t^d$ represents observation from all k sensors at specific time t, while each column $X^i = x_1^i, x_t^i, \ldots, x_n^i$ represents one sensor temporal readings from all n touch periods over time. Then, each single time-series sensor value is represented as $x_t^i \in \Re^{nxd}$ that collected from ith sensor at time instant t, where d as the total dimension of collected data on a specific smartphone. For instance, $d = 13$ when using k=5 sensors (AcElGrGyMg), given the touch timestamps $(Ts) \in \Re^1$, accelerometer reading $Ac \in \Re^3$, elevation reading $El \in \Re^1$, gravity reading $Gr \in \Re^3$, gyroscope reading $Gy \in \Re^3$, and magnetometer reading $Mg \in \Re^3$. Finally, this matrix formulation of raw time-series sensory data includes readings of different ranges depending on each sensor's quality. Therefore, we applied a moving average-based smoothing technique with various values of window size w to eliminate outliers and clean the raw data (see Section 4.4).

3.2. MotionID: Architecture

As an authentication scheme, the MotionID system architecture consists of two phases: enrollment/training and authentication/testing. In Fig. 2, MotionID presents the series of four major components for authenticating the user using behavioral biometrics from sensory data on smartphones as follows.

Data collector application. We developed the MotionID data-collector application that should be installed on users' smartphones. The application implicitly and continuously runs in the background, collecting data from five (AcElGrGyMg) sensors and touching timestamps *Ts*. Since MotionID is designed using client–server architecture, the collected data files automatically forward to the user's directory on the server using an SFTP (secure file transfer protocol) connection. The application is valid on smartphones of API 23 (Android 6) and higher.

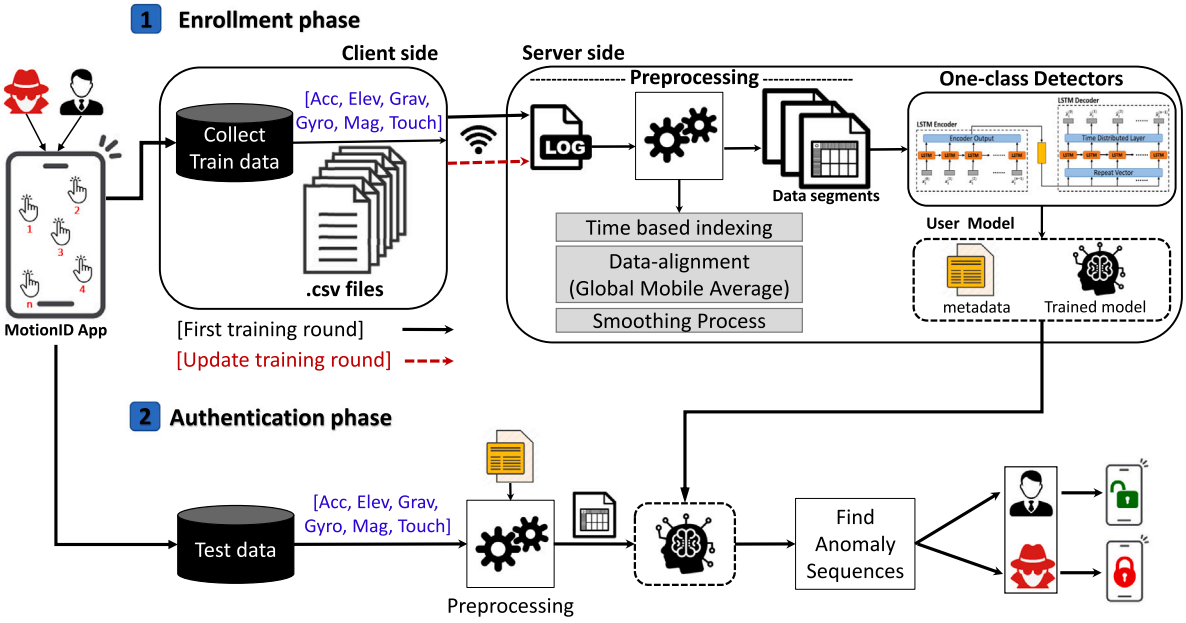


Fig. 2. Architecture of the proposed MotionID system.

Preprocessing. As shown in Fig. 2, preprocessing includes steps of creating log-files, time-based indexing, data-alignment, and smoothing process so that the prepossessed data segment of a user is suitable to feed to the MotionID models for training or testing. We provide a detailed explanation of these steps in Section 4.1.

Enrollment/training phase. After preprocessing, we apply anomaly detection methods to build a legitimate user model. Note that the binary classification approaches used in the literature are not practically valid as it is difficult to obtain imposter data while building models for real-world usage. Therefore, the legitimate user's data is the only available data for building one-class models, a.k.a. anomaly detectors, to detect attackers whose samples are deviated. In this work, we decided to select five common one-class detectors, which are deep learning LSTM-based autoencoders (LSTM-AE), One-class SVMs (OCSVM), Local Outlier Factor (LOF), Isolation Forest (IF) and Elliptic Envelope (EE) – more details about them are presented in Section 5. These one-class classification methods are more realistic than binary to support our main goal of going towards developing a practical authentication system, which continuously detects abnormal observations collected from sensors while accessing smartphones and predicts them as deviated data from the usual and expected behaviors.

Authentication/testing phase. In this phase, only short durations (*i.e.,* 1 to 5 s) of test samples are captured by the MotionID collector application. The data must be captured using the same sampling rate determined by *global mobile average (Q)* technique in the enrollment phase (*i.e.,* training phase) for every device independently. Then, preprocessing and data segment generation steps are applied the same way as in the enrollment phase to feed the trained model. The five anomaly detectors calculate the anomalous scores and distinguish whether test samples belong to the legitimate user's template created in the enrollment phase or not.

We explain the scenario of the MotionID system for authenticating a user when interacting on smartphones as follows. After bypassing the primary security method (e.g., PIN, Lock pattern, or Fingerprint), the smartphone is unlocked and a user (normal or attacker) can freely explore it. However, when touching events occur, MotionID automatically starts collecting time-series of sensory data and touch timestamps *(Ts)* that reflect behavioral biometrics in an unconditional manner – i.e., when the user globally interacts on any application over the entire smartphone *(over-all-apps)* and the collection process occurs implicitly in a background service. On the client side, the data collection process occurs on smartphones only once by the normal user to train authentication models. Then, MotionID sends collected data files to the trusted third-party server (the server is under our control) via a secure FTP connection. On the server side, MotionID automatically creates a unique directory for each user based on the user's device ID to store the collected files. After that, several steps run on the server, such as preprocessing, data segment generation, and creating one-class models with metadata for authentication. The trained model is then implemented with the device owner profile for later prediction. In the testing stage, MotionID collects behavioral data in real-time on smartphones for a few seconds when a user touches a smartphone for authentication. In practice, whenever the user keeps touching the smartphone, MotionID continuously collects data implicitly in the background, runs the trained model, finds anomalies, and gives real-time predictions about whether the current user is legitimate to lock/unlock the smartphone accordingly.

3.3. Design challenges

To develop a practical behavioral-based authentication scheme, we briefly summarize challenges that we faced during the implementation of the work as follows: (1) *Retraining models*. To address the problem of behavior changes of legitimate users over

time and mitigate its negative influence on the authentication performance, it is necessary to occasionally collect more data files and store them in the user's directory on the server to retrain models. Thus, to ensure retraining models only on the *newly-collected* data, we created a *log-file* that logs *time-date* of new data files and ignores data files of previous training time. ② *No touch attributes*. During developing the collector application of MotionID, we found that touch data such as gestures, swipes, taps, and x-y coordinates are no longer available and restricted on newer Android versions when globally touching any application in the phone because of security reasons. This serious restriction could make previous touch-based authentication works not valid in the future. Therefore, we investigated that only touch timestamps and sensory data are still available and can be acquired on Android devices. ③ *Single-device-evaluation*. As we mentioned that collecting data from both owner and attacker users on the same device is a more realistic approach to evaluating authentication models for real-world applications. However, it is hard to ask owners to let other users use their devices because of privacy concerns. Thus, we used general five devices from our laboratory and asked ten users to use them for experiments. ④ *Number of sensors*. Since MotionID collects data from five sensors, we found that not all smartphones are equipped with all sensors. For example, Galaxy S8 and Galaxy Note5 devices are equipped with five sensors, however, the Galaxy Wide4 device is equipped with only four sensors (has no Elevation sensor). To be generalized over all devices, MotionID first detects number of available sensors on a device and generates data segments to train models accordingly. ⑤ *Different sensors sampling rates*. From experiments, we found that sensors generate data with sampling rates that differ from one sensor to another on the same device as well as from one device to another. Since the model is fed with data collected from multiple sensors, data segments used to train models should have the same dimensions over all sensors. To solve this alignment issue, we use "Global mobile average (Q)" technique, in which all dataframes are either under- or over-sampled to the Q value, to generate final data segments with consistent dimensions suitable to use as input to the model for training or testing (see Section 4.2).

4. MotionID: Methodology

This section describes the details of the MotionID methods shown in Fig. 2, including preprocessing steps, Global Mobile Average technique, and generating data segments for every user to build his/her template profile when using smartphones.

4.1. Preprocessing

Raw sensory data should be processed to effectively reflect smartphone user behaviors.

Log-files. A *log-file* is responsible for selecting only new data files that are collected recently for training/re-training the model and skipping the old data files that exist in the user's directory on the server — this is important to consider the impact of the user's behaviors changes on smartphones over time. In other words, since user behaviors change over time, authentication models should occasionally be retrained and updated. Each file was named as follows: *"deviceID_SensorType_date_time.csv"* so that we know to which device and sensor the file belongs as well as the exact date/time of collection. Whenever the authentication performance decreases, there is a need to collect new behavioral data from the user to retrain models and improve accuracy. To do so, MotionID collects new sensory data files, stores them in the user's directory on the server, and updates the log file by inserting the dates and times of new sensory data files. At training/retraining, MotionID automatically accesses the log file, reads all files' dates and times, and extracts only recently collected files based on their dates and times records. By doing this, we guarantee to train/retrain the user model using only new data files, which indeed reflect the recent user's behaviors.

Time-based indexing. This step aims to extract sensory data that only occur during touching periods and filters out sensory data that occur during other times. In other words, sensors on smartphones are registered once to listen to all events of the user regardless of whether he is touching the smartphone's screen or not. In android programming, *SensorManger* class lets developers access the device's sensors and register listeners for generating time series data using the *OnSensorChange* method continuously. However, for the purpose of authentication, we focus on collecting sensory data during touching times only *i.e.,* when normal users (or imposters) interact (touching, swiping, taping, browsing, *etc.*.) with smartphones. We ignore sensory data that occur during other non-touch times and unwanted situations, such as smartphones being idle inside pockets or on the table. To do this, we implemented time-based indexing, in which we recorded touching timestamps and compared them with timestamps collected with data from five sensors. Then, MotionID automatically selects the first and last touch timestamps of every touch period (*i.e.,* one second) and extracts corresponding dataframes from all sensors that happened within the same two timestamps. By concatenating all extracted sensory dataframes that occur during (ρ) touch periods, we can create a piece of data (Data Block DB) from each sensor.

4.2. Global mobile average technique

Until now, we explained how MotionID creates data blocks (DB) for each of the five sensors by extracting and concatenating data frames that occur during touch periods only. We justify touch periods as small durations (*e.g.,* a few seconds) that a user needs for touching (typing, tapping, or swiping) the screen. For example, during writing on screen, users usually touch the screen for a few seconds then remove their fingers, think, and re-writing again, and so on. Our collector application only collects motion sensor data whenever the screen is touched. Therefore, whenever the screen is touched for a few seconds (called one touch period), we precisely collect sensory data as one data frame. The process will be repeated as long as the user keeps touching the screen, building data frames of sensory data exactly corresponding to touch periods. However, we found that the lengths of extracted data frames are different for each touch period, even when generated by the same sensor on the same device. Fig. 3 shows that for each sensor belongs to the range of sensors ($Sensor_1$, $Sensor_2$, …, $Sensor_k$) on a device, touch periods (Tp_1, Tp_2, …, Tp_ρ) generate blocks of

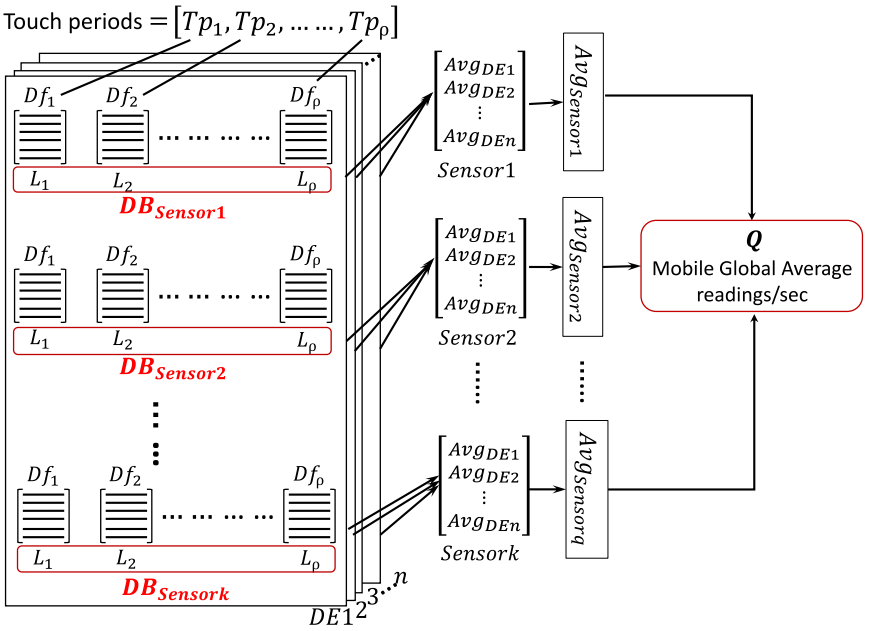


Fig. 3. Computing global mobile average (Q).

dataframes (DB_{Sensor_1}, DB_{Sensor_2}, …, DB_{Sensor_k}) with various lengths of dataframes ($L_1, L_2, \ldots, L_\rho$). Empirically, we found that the variations in readings/second happen due to different sampling rates that are directly related to the behaviors of a user who is using a smartphone and the sensors' capabilities, which are different from one device to another. In detail, the *OnSensorChange* method is continuously called whenever there is a new sensor event/change occurs. So, during a specific touch period (*i.e.,* few seconds), if the behavior of the user (typing and touching rhythm) on the smartphone's screen is fast, this leads to a lot of changes and frequent calls for the *OnSensorChange* method, generate more sensor events, and hence increase the length of sensory dataframes and vice versa. This reflects the real-world usage where some users can interact (typing, browsing, tapping, *etc.*.) on the screen faster and generate larger data frames of sensory data than the slower users even during the same touch period. Also, fixing the sampling rate to a static value (*e.g.,* 100 Hz) could not be applicable for some smartphones that do not provide this rate of readings generating. Therefore, our innovation is to NOT use the same sampling rate of sensors and the same amount of data samples for all users during training authentication models. *Therefore, experimentally, we declare that the assumption presented in the state-of-the-art works of using a fixed sampling rate for generating data from different sensors on different devices and users is impractical.*

Thus, achieving our research goal of developing a practical authentication system, we present a novel Global Mobile Average technique in which, for each training round, MotionID dynamically calculates the average amount of generated readings from all selected sensors per device independently. In other words, the sampling rate (readings/second) used to generate data frames of sensory data is a dynamic parameter that depends on several factors such as the sensor's quality, the smartphone's type, and the user's touch behavior. For example, if we want to train a model, we collect sensory data of the owner user who continuously touches the smartphone for time interval T (*e.g., n* minutes). Note that, since sensors generate large amounts of data during a small amount of time and to avoid losing sensory data (.csv) files when transferring to the server due to network communication failures, we innovated dividing data into pieces called data extents (DEs). As shown in Fig. 3, each data extent contains five types of sensory data collected during time t_i (*e.g.,* one minute), which belongs to $T = (t_0, t_1, t_2, \ldots, t_n)$, where T means the total time of n data extents. We justify a data block (DB) as part of (DE) which indicates all dataframes that are extracted during ρ touch periods from only one sensor; however, a data extent (DE) indicates a collection of data blocks from k sensors on a device during time t_i. Therefore, we calculate global average readings by averaging the lengths of all ρ touch periods for k sensors over n data extents. In detail, for each sensor data block, ($DB_{Sensor1}$, $DB_{Sensor2}$,…, $DB_{Sensorn}$), we average all dataframes' lengths ($L_1, L_2, \ldots, L_\rho$) across n data extents ($DE_1, DE_2, \ldots, DE_n$) to get the vector ($Avg_{DE1}, Avg_{DE2}, \ldots, Avg_{DEn}$). By repeating the process for each sensor, we get five vectors of average values for the five sensors for all data extents. We average each vector to get five averaged values ($Avg_{Sensor1}$, $Avg_{Sensor2}$,…, $Avg_{Sensorn}$) that are used to compute the final value of the global mobile average (Q). Note that we provide a meta-data file with the trained model that contains information such as *device-ID, number of sensors, and global mobile average (Q)* value for every smartphone independently. The meta-data is also used when testing the model to authenticate legitimate and imposter users. *In this way, we guarantee the applicability and generality properties of a practical authentication system that is resilient for diverse sensors' capabilities, devices' models, and users' behaviors*.

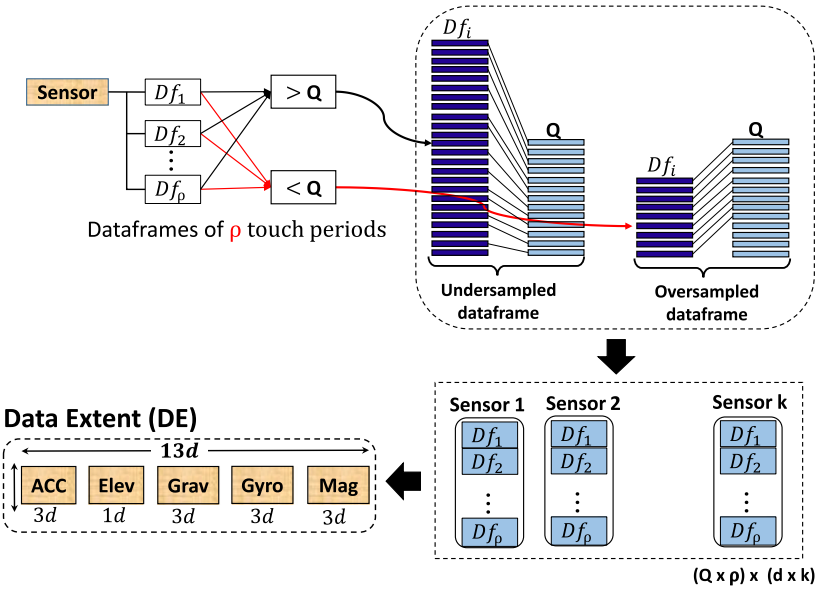


Fig. 4. Data extent preparation steps.

4.3. Data segment generation

After computing the global mobile average (Q), we use the Q value to perform a data alignment process on the raw dataset for generating data segments of a user to be suitable to feed to the models for training or testing. As explained in the previous section, the data extent (DE) is a piece of sensory data collected from all selected sensors during time interval t where $t \leq T$. However, we consider each data extent as a matrix that should have the same dimensions: length (number of dataframes × samples per dataframe) and width (number of columns per sensor × number of sensors). While the number of dataframes is constant and equals the number of touch periods that a user conducted, the number of samples per dataframe varies and depends on the sensor's rate and user behaviors. Therefore, we need to perform an alignment process for all dataframes that come from all sensors to generate matrix-based data extent. The key innovation is to use the global mobile average (Q) to perform the alignment in the matrix length dimension. In addition, the number of columns per sensor is as follows: Accelerometer $3d$, elevation $1d$, gravity $3d$, gyroscope $3d$, and magnetometer $3d$. Note that, not all smartphones have the five sensors, so MotionID can automatically pre-determine the number of sensors available on a smartphone from the beginning to set up the width dimension of the data extent (DE). For example, the Galaxy S8 smartphone provides all five sensors (*i.e.,* a total width of 13d), while Galaxy Wide4 provides only 12d (no elevation sensor embedded). Fig. 4 shows the process of how we used the estimated Q value to generate data extent that aligned for all touch periods (length) and sensors (width) by implementing oversampling and undersampling methods. In detail, for every dataframe of a specific sensor, we compare its length with the estimated Q value. If it is larger than Q, we apply the undersampling process to downsize its length to Q by randomly removing some samples. Also, if the dataframe's length is less than Q, we apply the oversampling process to increase its length to Q. This step guarantees that all output dataframes have equal lengths to Q. After vertically concatenating all dataframes and horizontally for all provided sensors, we get the final size of the data extent as $(Q \times \rho) \times (d \times k)$, collected during time interval t (*e.g.,* a minute). Then, we use the concept of the data segment (DS) representing the whole sensory data collected from each user and required for a training round. In detail, to create the final data segment (DS) of sensory samples, we vertically concatenate all (n) data extents for each user collected during time interval $T = (t_0, t_1, t_2, \ldots, t_n)$. Finally, we collect all data segments for all users (one data segment for each of them on his/her device independently) to train their authentication models on the server, as shown in Fig. 2.

Description of algorithm 1: Here, we explain the detailed steps of two parts. (1) Estimating the global mobile average (Q). (2) Creating the final data segment (DS) of the interacting user (owner or attacker) during the time interval T on a smartphone. As explained a data segment (DS) indicates a complete dataset that is collected during time interval T, where $T = (t_0, t_1, t_2, \ldots., t_n)$, contains n data extents and used for training authentication models. **As input,** for each user, we used all raw sensory data (.csv) files collected during the time interval T and stored in the user's directory on the server, touch timestamps, and device-ID. **As output,** we get the final data segment of dimensions $DS = [(\Phi \times \rho \times \mathbf{Q}) \times (d \times k)]$ that was created based on the estimated value of the global mobile average (Q). **In the beginning, Lines 1-4** show the acronyms used in the algorithm. During the whole process, we iterate over each user, each data extent, each sensor in a device, and each touch period per sensor, respectively. In **Lines 6-20**, we handle the global mobile average (Q) process to calculate the average of readings/second for each user on a specific device. In **Lines 22-34**, we provide steps regarding data alignment to generate the final data segment that contains sensory data gathered from all data extents during tough periods. During **Lines 10-11**, using the start and end timestamps of each touch period, we go to

Algorithm 1: Mobile Global Average (**Q**) and Data segment (**DS**) generation algorithm.

Input : Sensory data, Touch timestamps, User (Device ID).
Output: Data segment for each user: DS = $[(\Phi \times \rho \times \mathbf{Q}) \times (q \times k)]$

1 Ψ: number of users (devices).
2 Φ: number of data extents **DEs** for a user.
3 k: number of sensors per device.
4 ρ: number of touch periods.
5 ——— Mobile Global Average (**Q**) ———
6 **foreach** *mth user in* Ψ **do**
7 **foreach** *nth DE in* (Φ) **do**
8 **foreach** *ith sensor in* k **do**
9 **foreach** *jth touch period in* ρ **do**
10 - Get (start,end) timestamps;
11 - Get the sensory dataframe (df_{ij}) occur within (start,end) timestamps;
12 **end**
13 - Get array of all touches' dataframes → $df_i = [df_{i1}, df_{i2}, ..., df_{i\rho}]$;
14 - Compute dataframes' lengths → Average them $[AvgLeng_{(DE_n,i)}]$;
15 **end**
16 - Append $[AvgLeng_{(DE_n,Acc)}, AvgLeng_{(DE_n,Elev)}, AvgLeng_{(DE_n,Grav)}, AvgLeng_{(DE_n,Gyro)}, AvgLeng_{(DE_n,Mag)}]$;
17 **end**
18 - Get overall $[AvgLeng_{(Acc)}, AvgLeng_{(Elev)}, AvgLeng_{(Grav)}, AvgLeng_{(Gyro)}, AvgLeng_{(Mag)}]$;
19 - Get global mobile average $[Q(readings/sec)]$ for the device;
20 **end**
21 ——— Data segment (**DS**) generation ———
22 **foreach** *mth user in* Ψ **do**
23 **foreach** *nth DE in* (Φ) **do**
24 **foreach** *ith sensor in* k **do**
25 - Take each dataframe of $[df_{i1}, df_{i2}, ..., df_{i\rho}]$;
26 - Apply undersampling_oversampling method;
27 - Get $DB_i[(\rho \times Q), d]$;
28 **end**
29 - Concatenate horizontally all sensors' data blocks (DBs);
30 - Create one data extent $DE_n[(\rho \times Q), (d \times k)]$;
31 **end**
32 - Append vertically all data extents (DEs);
33 - Get final data segment (DS) of a user for one training round $DS = [(\Phi \times \rho \times Q), (d \times k)]$;
34 **end**

every sensor's data file collected within the data extent's time (t) and extract the corresponding dataframes that appear with given timestamps. By finishing all touch periods (ρ) for a given ith sensor, we get all its dataframes $df_i = [df_{i1}, df_{i2}, \dots, df_{i\rho}]$. Then, we compute all dataframes' lengths and average them to get $[AvgLeng_{(DE_n,i)}]$ for the ith sensor on the nth data extent. By repeating the process for other sensors on advice, we get averages of five sensors' dataframes for the nth data extent as in **Line 16**. Continuing, in **Lines 18**, considering that there are several data extents collected during times $t_0, t_1, t_2, \dots, t_n$ of user interaction on a smartphone, this step aims to the overall average for each sensor on all data extents $[AvgLeng_{(Acc)}, AvgLeng_{(Elev)}, AvgLeng_{(Grav)}, AvgLeng_{(Gyro)}, AvgLeng_{(Mag)}]$. Following, **Line 19** estimates global readings per second (Q) for a user on a device which practically reflects the data generation rate based on the user's behavior and device capability during one training round. In the second part, we handle *data alignment* of the five sensors across all data extents to create the final data segment package of a user when interacting on a smartphone during time interval T. Therefore, in **Lines 24-28**, we loop over every sensor and take each of its dataframes that corresponding sensory data within touch periods and apply oversampling or undersampling method depending on how much the length of the dataframe compared with the estimated (Q) value. This creates a data block (DB) for each sensor of size $[(Q \times \rho), d]$. After that, **Line 29-30** represents the horizontal concatenating of all data blocks of the sensors in order to create one data extent (DE). Finally, **Line 32-33** explain that by vertically concatenating all created data extents, we can generate one piece of the data segment (DS) that reflects the user's data through the whole usage time determined for one training round.

4.4. Smoothing and normalizing

Since we have created a complete user data segment for training and testing models, and to develop stable models, the dataset needs smoothing and normalization before the training process. We justify these steps as follows. The first is to smooth the dataset

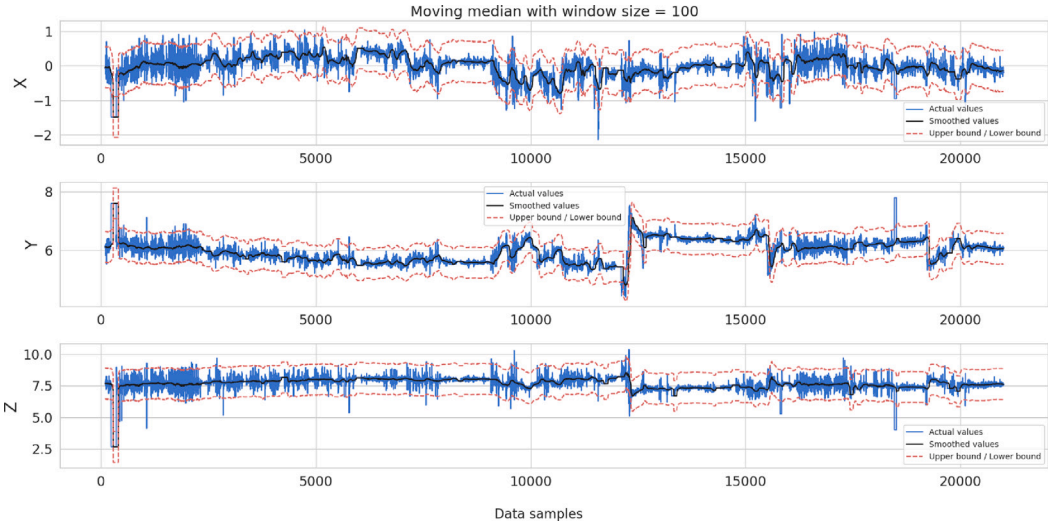


Fig. 5. An example of how to smooth the signal of an accelerometer sensor (x, y, and z) using the moving average window technique.

by removing outliers and cleaning noisy values that occur during unwanted events when touching a smartphone. In fact, the raw collected sensor samples cannot be applied to train models because they are not pure and include outliers and noisy readings occur due to instability of the user's hand and body movements (*e.g.,* during walking) while touching the smartphone's screen. Since our approach considered practical unconditional settings when collecting the dataset, getting noisy sensory readings in the dataset is heavily expected. To overcome this, we applied the common moving average (sliding window) as a smoothing technique and tested its accuracy over four different window sizes (100, 500, 1000, and 2000) and using two smoothing methods (Mean and Median) on five different smartphones using LSTM-based bidirectional autoencoders. Fig. 5 shows an example of a data smoothing process applied to accelerometer sensor readings acquired from a Galaxy S8 smartphone. In this example, we used a window size of 100 and a Median-based smoothing method to eliminate the outliers (spikes shown in blue color) and smooth all readings (shown in black color). From our experiments, we found that median-based smoothing provides higher F1 scores for four devices out of the five than the mean-based method, while the best window size is 100 for two devices, 2000 for the other two devices, and 1000 for one device. This is theoretically expected because the median has good performance at smoothing some specific types of noise and does not create unrealistic time-series values. Regarding window size, we checked 100 and 2000 lengths and found that choosing 2000 is less effective since it can get more irrelevant information and result in a loss of time-series resolution. Thus, we empirically selected the median-based smoothing method and a window size of 100 for the rest of the experiments in the work. The second process is normalizing the time-series data. We Justify that this process is needed because our work involves collecting readings from multiple sensors with various ranges of values. This step is important and common for the preparation of the input dataset for machine learning and deep learning models. To do this, we perform a linear transformation on the original data. We scale different ranges of data from sensors to within the range (0, 1) using min–max normalization for a faster computing process.

5. One-class detectors

Typically, our task is to exploit behavioral patterns extracted from the sensors of smartphones to authenticate a user. To capture these behavioral patterns and make practical classification, one-class detectors based on deep learning and machine learning mechanisms are the best candidates for this task. MotionID utilizes five common one-class detectors for smartphone time-series data authentication tasks.

LSTM-based AutoEncoder detector. For practical authentication, we utilize anomaly detection approaches on the time-series data instead of binary classification used in many state-of-the-art studies, because imposters' data are not available when training models for creating user (owner) template profiles in real-world usage. Anomaly detection means identifying/finding data points or events that deviate from the usual patterns. We choose Autoencoders as one-class unsupervised detectors to learn efficient representations of unlabeled data. Autoencoders contain two main parts: an encoder that maps the input sequences to a code and a decoder that reconstructs the output sequences from the code. Fig. 6 shows the architecture of Autoencoder used in the work, which learns the time-series data representation of its inputs (encoding), typically for dimensionality reduction, and regenerates the inputs from the encoding (decoding). In addition, we use Long Short-Term Memory (LSTM) networks with the autoencoder since LSTM is the most popular version of recurrent neural networks in which layers are made of memory cells, which makes it easier to remember time-series data of user's behaviors in memory. Specifically, to train models using LSTM-Autoencoders, we divided the dataset of each legitimate user into sequences where each sequence contains time-steps (TS = 1, 5, 10, or 20) of data points (rows) and has all sensors' dimensions (*i.e.,* columns). Splitting time series into sequences is important to reshape the input data

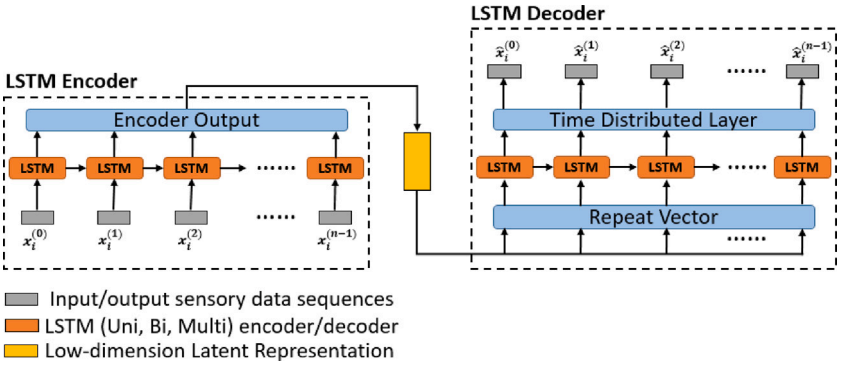


Fig. 6. LSTM Auto-encoder architecture.

into (samples, time-steps, and features) to be suitable for LSTM networks. After that, Autoencoder takes every sequence as input $(x_i^{(0)}, x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(d-1)})$ and outputs the $(\hat{x}_i^{(0)}, \hat{x}_i^{(1)}, \hat{x}_i^{(2)}, \ldots, \hat{x}_i^{(d-1)})$ sequence of the same shape. Following, we compute mean absolute errors (MAE) for every sequence to determine the legitimate user's training threshold values. Fig. 7 shows the reconstruction errors matrix and the three cases of computing thresholds. Case 1, named Thr_{R-C}, we compute the average of max_1 and max_2, where the value of max_1 indicates the maximum of all error averages calculated in a column-wise way (*i.e.,* Axis = 0) and the value of max_2 indicates the maximum of all error averages calculated in a row-wise way (*i.e.,* Axis = 1). Case 2, named Thr_C, indicates only the maximum of all error averages calculated in a column-wise way. Case 3, named Thr_R, indicates only the maximum of all error averages calculated in a row-wise way. In addition, during our experiments, we investigate the performance of the authentication task using three different LSTM-based architectures: uni-layer simple LSTM, bidirectional LSTM, and multilayer LSTM. In Section 6.3, we provide experimental details about threshold cases, the performance of three LSTM architectures (uni, Bi, and Multi), and the tuning task of model hyperparameters (*e.g.,* number of time-steps, epochs, batch size, neurons size, Dropout, Learning rate).

Machine-learning-based detectors. Several machine-learning-based techniques are used for one-class classification that involves fitting a model on the "normal" data and predicting whether new data is normal or an outlier/anomaly. In our work, we selected four common ML algorithms for performing a one-class detection as follows. *One-class SVM (OCSVM)*, *Local Outlier Factor (LOF)*, *Isolation Forest (IF)*, and *Elliptic Envelope (EE)*. Similarly, we provide experimental details about finding optimal hyperparameters for the four machine learning algorithms in Section 6.3.

6. Evaluation

In this section, we explain the experimental setup and demonstrate the evaluation results of all experiments.

6.1. Experimental setup

Here, we explain the experimental setup, including the MotionID Android application, datasets, and used devices. Also, we demonstrate the evaluation metrics and approaches used for all experiments.

6.1.1. MotionID Android application

As the primary aim of our work is to focus on Android-based smartphones, we developed a prototype data-collection application on the Android platform using JAVA programming that transparently operates in the background to collect sensory data with timestamps during touch periods. We exploited multiple sensors: accelerometer, elevation, gravity, gyroscope, magnetometer, and screen touch. Specifically, the application hooks on to the *onSensorChanged* method from *SensorManager* and *SensorEventListener* to listen to built-in $Accelerometer$, $Elevation$, $Gravity$, $Gyroscope$, $Magnetometer$, and $touch$ sensors for collecting the sensory measurements in $3d$, $1d$, $3d$, $3d$, $3d$ dimensions respectively. MotionID application is valid for Android smartphones that start from API 23 and higher. Legitimate users must first install the MotionID application on their smartphones and accept the "Privacy and Term Conditions".[4] We added this to the application to get the user's agreement before data collection. Furthermore, we designed a "Question Survey" on the application's first page to collect user characteristics and demography. Note that, these two tasks (*i.e.,* Privacy and Term Conditions and Question Survey) will be conducted only one time at the installation of the MotionID application before collecting the dataset. For every smartphone, we store the user's information and survey answers securely in the user's directory on the server — it means that no need to repeat the tasks even when repeat collecting datasets for the next training/retraining rounds. Each sensor's data is saved in a separate file named with the device-ID, sensor type, date, and time. Then, MotionID connects to the server and sends dataset files using a secure FTP protocol connection (SFTP) from the internal memory of

[4] https://github.com/Mohsen-Ali-Alawami/Privacy-Terms-and-Conditions.git

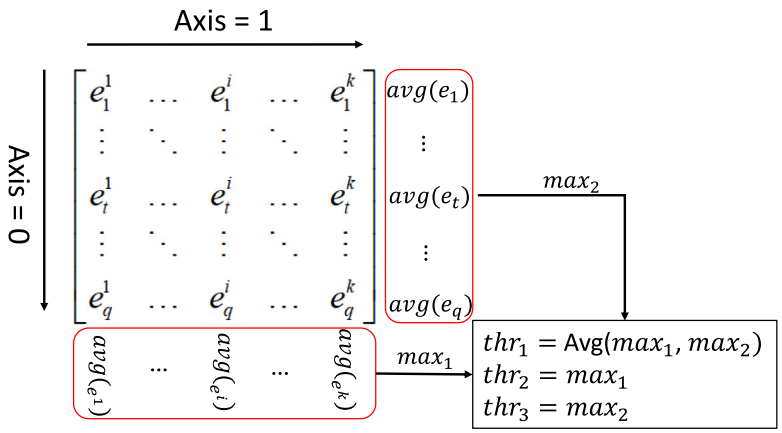


Fig. 7. Calculating threshold values for LSTM-AE.

the phone to the user's directory named by the device-ID on the server. In addition, we developed the feature of "automatic-lock" using *DevicePolicyManager* class[5] to immediately lock the device when an attacker is detected during the authentication time.

6.1.2. Devices and datasets

For our experiments, we selected five different devices for evaluating the performance of our system as follows. (1) LG LM-Q510N with Android version 9 and API 28. (2) Samsung Note5 with Android version 6 and API 23. (3) Samsung Note10 with Android version 12 and API 31. (4) Razer Phone2 with Android version 9 and API 28. (5) Samsung Wide4 with Android version 9 and API 28. It is worth noting that (1) we intentionally selected these five devices of different brands (e.g., Samsung, LG, and Razer) with various android versions (ranging from Android V6 API 23 to Android V12 API 31); (2) due to the practical requirements that we considered such as using *single-device-evaluation* (*i.e.,* both legitimate and imposter users should access the same device over all applications) – many participants refused those other users (imposters) access their own devices due to privacy concerns. Therefore, it is challenging to recruit a large number of devices and users for data collection under the privacy concerns of the participant, as well as it is hard for us to buy a large number of devices. We think that five devices are sufficient to show the feasibility of the system — many studies exploited one or two devices for their experiments.

Regarding datasets, we collected three different types of datasets, named Dataset-A, Dataset-B, and Dataset-C, as follows. *Dataset-A*: We collected this dataset from eight different smartphones (only owner users) – the aforementioned five devices, Samsung S8 with Android version 9 and API 28, LG LM-V409N with Android version 9 and API 28, and Xiaomi MI MIX2 with Android version 9 and API 28. This dataset aims to investigate the first research question *RQ1* by exploring sampling rates of the five selected sensors (i.e., Accelerometer, Elevation, Gravity, Gyroscope, and Magnetometer) during long-term data collection over several days as well as checking smartphone-to-server connection issues and storage sizes on the server. *Dataset-B:* We collected this dataset using the aforementioned five devices for the purpose of conducting experiments for models' optimal hyperparameters settings of the five selected one-class detectors, tuning best window sizes, and smoothing methods. During the collection process of this dataset, we ask ten users (6 males and 4 females) to participate in the five smartphones (*i.e.,* a legitimate user and an imposter for every device). Their ages range between 25 to 50 years, all of them have experience using smartphones, have a high educational level (bachelor and higher), and are informed about the study's purpose before doing the experiments. *Dataset-C:* To consider the concept of data drifts due to changes in the user's behaviors over time, we deliberately collected this dataset in the same manner as Dataset-B but after several weeks. This dataset was used for the rest of the evaluation experiments of the work to address the remaining research questions (i.e., *RQ2*, *RQ3*, and *RQ4*). We received permission from our university's Institutional Review Board (IRB) to collect datasets. We also informed the participants about the experimental purpose and type of collected data. In addition, we declare that during dataset collection, participants were freely using their smartphones for data collection without any specific instructions from the authors regarding specific postures, positions, or environments/conditions — to keep our goal of conducting experiments as in real life where every user has to use his/her device freely and unconditionally.

6.1.3. Evaluation metrics and approaches

During the enrollment phase, only normal data collected from the legitimate user was used for creating models (one class). However, to compute the evaluation metrics of the performance of MotionID, we need other users' data (*e.g.,* imposters). Therefore, we used the following metrics to evaluate the effectiveness of MotionID: *True negative (TN):* The data samples from legitimate users (i.e., owners of smartphones) are correctly identified. *False positive (FP):* The system incorrectly rejects the data samples from legitimate users, and hence the smartphone is wrongly locked. *True positive (TP):* The actual data samples of imposter users are

[5] https://developer.android.com/reference/android/app/admin/DevicePolicyManager

Table 3
Sensor sampling rates (Hz) of the smartphones.

Sensor\Device	Samsung S8 (API 28)	Samsung Wide4 (API 28)	Xiaomi MI MIX2 (API 28)	LG LM-V409N (API 28)	LG LM-Q510N (API 28)	Razer Phone2 (API 28)	Samsung Note5 (API 23)	Samsung Note10 (API 31)
	Readings/second (when using 100 Hz)							
Acc	100	100	100	100	100	100	100	**105**
Elev	**10**	None	**25**	**30**	None	**25**	**25**	**25**
Gravity	100	100	100	100	100	100	100	**105**
Gyro	100	100	100	100	**200**	100	100	**105**
Mag	100	100	**50**	100	**50**	100	100	100

correctly identified anomalies. *False negative (FN):* The data samples from imposters are incorrectly identified as legitimate user data and hence accepted by the system. We considered *Precision*, *Recall (Sensitivity)*, and *F1-score* as metrics to evaluate the performance of MotionID. *Precision* is defined as $\frac{TP}{TP+FP}$, which measures the ratio of correctly predicted positive time series samples to the total predicted positive samples. *Recall (Sensitivity)* is defined as $\frac{TP}{TP+FN}$, which measures the ratio of correctly predicted positive data samples to all data samples in the actual class. The formula for the F1 score is F1 = 2 × (precision × recall) / (precision + recall), which is a weighted average metric emphasizing the model's performance regarding false positives and false negatives.

Our evaluation approaches to assess the performance of MotionID were based on several settings as follows. First, users interact with smartphones freely in unconditional circumstances in which we collected sensory data while they are touching (*e.g.,* tapping, swiping, browsing, texting, *etc.*.) during walking, standing, and sitting. Second, data is collected when users use the entire phone to freely access any application (*over-all-apps*) not specified on a pre-determined application. Third, both legitimate and malicious users are asked to use the same smartphone, and their data was collected in the same manner on the same device (*single-device-evalution*) to simulate realistic scenarios when attackers access the owner's device itself. Considering the above settings, we further conducted several evaluations to address our research questions using five smartphones with different brands, different short-durations of touching data, five one-class detectors, time complexity, energy consumption, and memory usage.

6.2. Experiment 1: Investigating ***RQ1***

Here, we investigate whether the assumption of setting a fixed sampling rate for all devices and sensors in a device is valid or not. Based on our experiments, we carefully explored the collected datasets from diverse smartphones and found that assuming the generation of data samples from sensors at a fixed rate (as in existing studies), is not practical, especially for developing a general authentication system that can be deployed on divers smartphones with different types and quality of sensors. To clarify this, we collected *Dataset-A* from eight smartphones of different brands and android versions. In the coding of the application, we deliberately fixed the sampling period to 100 Hz (*i.e.,* the sampling period of 10 ms) for all five sensors. We installed the MotionID application on the eight devices and asked owner users to use the phone with normal touching behaviors as usual. We vary the data collection duration ranging from a few minutes to a few days of the application usage. For example, the user on the Samsung S8 device collected data for 11 days from five sensors — we checked his directory on the server and found that the number of sensory data files is 245 and the total average data size of 4.28 GB. Also, the user on the Samsung Wide4 device collected data for 10 days from four sensors (the elevation sensor is not available) – his directory on the server has 348 data files and a total average data size of 6.08 GB. Note that, the number of files and the total size of data per user is different based on how much a user touches and interacts with the smartphone daily. The user on Samsung Note5 was asked to collect the dataset for around 10 min only and a total average data size of 29.6 MB. We do the same thing for the other smartphones, and all users' data files are received correctly to their directories on the server. By doing these experiments, we declare that we neither find any client–server connection issues nor missing data files during transmission from the smartphones to the storage directories of the users on the server. We manually analyzed all collected data files for the eight smartphones and calculated the provided sampling rate (readings/second) and the number of sensors available for each device.

Table 3 shows the results that provide the tested devices' details, available sensors, and corresponding sampling rates. In detail, we found that some devices have no elevation sensor, such as the Samsung Wide4 and LG LM-Q510N devices; other devices provide low elevation sampling rates ranging from 10 to 25 readings/second. In addition, the gyroscope of the LM-Q510N device provides 200 readings/second. Also, the magnetometer sensor, which reads the surrounding electromagnetic field strength, appears to change based on the device type; *e.g.,* Xiaomi MI MIX2 and LG LM-Q510N devices have different rates. Finally, Samsung Note10 shows a slightly higher sampling rate than the 100 Hz for the accelerometer, gravity, and gyroscope sensors. It is worth noting that the assumption of a fixed sampling rate used by an authentication system is not realistic for generalizing over different types of devices and sensors — some sensors may not catch the pre-defined sampling rate, and others may generate higher data rates, and this also differs from device to another. These discrepancies in sampling rates arise problems for authentication such as the ambiguity about the needed amounts of sensory samples, the actual time for defining user identity, and the non-alignment of training datasets for building models. To tackle this issue for a practical solution, we proposed *Global mobile average* in which, we calculate the average of readings/second across available sensors on the device regardless of the pre-specified sampling rate and use this value for generating data segments for training model of each device independently. Another observation that we found the sampling rate differs from one user to another, even on the same device — this is because the sampling rate is directly related to user behaviors when using a smartphone, as explained in Section 4.2.

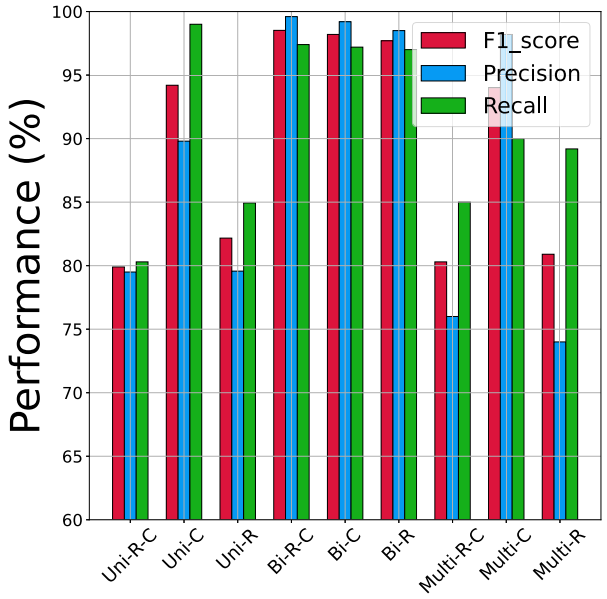


Fig. 8. Performance of various networks and threshold cases.

Table 4
Performance of best models' hyperparameters across different devices and threshold cases.

Device	Case	Time series size	Epoch	Batch size	Neurons size	Drop-out	Learning rate	F1-score	Precision	Recall
Samsung Wide4	Bi-R-C	20	100	256	512	0.3	1e−5	**98.52**	99.61	97.46
	Bi-C	10	50	32	512	0.2	1e−5	98.18	99.18	97.21
	Bi-R	10	100	128	512	0.3	1e−5	97.74	98.53	96.97
Samsung Note5	Bi-R-C	20	100	32	512	0.2	1e−4	87.67	85.75	90.09
	Bi-C	20	50	128	512	0.2	1e−4	89.21	89.75	88.68
	Bi-R	20	50	64	512	0.2	1e−4	**91.30**	93.39	89.30
Samsung Note10	Bi-R-C	20	10	32	512	0.3	1e−5	92.39	85.86	99.99
	Bi-C	20	10	32	512	0.2	1e−5	**92.75**	86.49	99.99
	Bi-R	20	100	64	512	0.2	1e−5	92.62	86.26	99.99
LG LM-Q510N	Bi-R-C	10	50	256	256	0.2	1e−3	**93.16**	96.20	90.31
	Bi-C	10	50	256	128	0.2	1e−3	90.40	83.90	98.01
	Bi-R	5	100	256	128	0.2	1e−3	90.82	86.21	95.95
Razer Phone2	Bi-R-C	5	100	64	128	0.3	1e−3	74.21	59.12	99.62
	Bi-C	5	100	64	128	0.3	1e−3	74.03	58.83	99.84
	Bi-R	10	10	64	256	0.3	1e−3	**74.41**	59.26	99.99

6.3. Experiment 2: Effects of models' hyperparameters settings

In this section, we demonstrate experiments conducted to choose the best optimal hyperparameters of one-class detectors for authentication.

Choosing best LSTM-network type. As explained in Section 5, the LSTM-based Autoencoders (LSTM-AE) were designed with three network architectures (Uni-LSTM, Bi-LSTM, Multi-LSTM) and the model's thresholds are calculated with three cases (Thr_{R-C}, Thr_C, Thr_R). Therefore, we conducted experiments for the nine different cases (three LSTM network architectures multiplied by three threshold cases) using LSTM-AE detector and *Dataset-B*. We computed F1, Precision, and Recall for each of the nine cases as shown in Fig. 8. During each case, we train the models using only the legitimate user dataset (one-class detection), while we tested them using unseen datasets collected from both legitimate and attacker users. For example, when conducting experiments using Bi-R-C (*i.e.,* Autoencoders with bidirectional LSTM networks and threshold case 1, column-wise and row-wise), we trained the model using the owner's dataset for each device by entering sequences of time-series sensory data and calculated the average model's threshold value over all sequences. After that, during the testing, we input every sensory time-series sequence from both owner and attacker users to the model. Then, the model generates the output sequence and estimates the reconstruction error by Bi-R-C way. If the error is higher than the threshold, that means the observation of the sequence is highly deviated from the owner's normal behavior and classified as an attacker sample. If the error is below the threshold, the sequence is labeled as normal. Note that, we iterate time-series sequences with different sizes of data samples such as 5, 10, and 20. By doing so for all data samples, we

Table 5
Optimal parameters of anomaly detectors.

Detector	Optimal parameters
LSTM-AE	TS = 20, Epoch = 100, Batch size=256, Neurons = 512, Drop_out = 0.3, learning_rate=1e-5
OC-SVM	*kernel = 'rbf', gamma =1, nu= 0.1*
LOF	*n_neighbors = 1000, Novelty = True*
IF	*n_estimators = 50*
EE	*contamination = 0.1 , assume_centered = False*

computed the final F1-score, recall, and precision metrics to classify owners and attackers on each device independently. Our final results show that the performance of all nine cases ranges from around 80% to more than 98% of the F1-score. We found that Autoencoders with bidirectional LSTM networks provide the best performance and achieve F1 scores of 98.52%, 98.2%, and 97.7% with the three threshold cases (Thr_{R-C}, Thr_C, Thr_R), respectively. As a conclusion of this experiment, we selected the architecture of Autoencoders with Bi-LSTM for the rest of the work. This is expected because bidirectional LSTM networks run time-series inputs in two ways, one from past to future in the forward direction and one from future to past in the backward direction to better understand the context of the information.

Choosing optimal hyperparameters. Based on the results of previous experiments, we used bidirectional network architecture with three threshold cases (*i.e.,* Bi-R-C, Bi-C, Bi-R) to conduct a grid search for six hyperparameters, as follows: Time-steps (1, 5, 10, 20), Number of epochs (10, 30, 50,100), batch size (32, 64, 128, 256, 512), number of neurons (64, 128, 256, 512), dropout (0.2, 0.3), and learning rate (1e-2, 1e-3, 1e-4, 1e-5). We implemented experiments on five devices: Samsung Wide4, Samsung Note5, Samsung Note10, LG LMQ510, and Razer Phone2. For every device, we input the user's data segment (owner and attacker) to the Anomaly Autoencoder Bi-LSTM network, where the model was trained and tested using one value of the six hyperparameters for each iteration. We also repeated the evaluation using each of the three threshold cases. In total, for each device, we run the experiments for 7680 iterations because we have 2560 (combinations of six hyperparameters) × 3 (threshold cases). We selected the best six hyperparameters that give the highest F1 score at every threshold case. Table 4 shows the results of finding the best hyperparameters for the MotionID models across five devices and three threshold cases. For example, for Samsung Wide4, the MotionID autoencoder model using Bi-LSTM and threshold case (Bi-R-C) achieves the best performance of F1 score 98.5% using when Time-steps of 20, epochs of 100, Batch size of 256, Neurons of 512, dropout of 0.3, and learning rate of 1e-5. However, the six hyperparameters changed when the model was evaluated using other threshold cases (i.e., Bi-C and Bi-R). By looking at the results of the other devices, we can see variations in the best values of hyperparameters which are expected because devices' data change based on sensors quality and behaviors of users on each device. To find common hyperparameters that can be generalized across devices, we counted the most repeated hyperparameters that showed the best performance over the five devices. In other words, we compared F1 values and selected Bi-LSTM with the (Bi-R-C) threshold case, Time-steps of 20 (repeated on 10/15), Epoch of 100 (repeated on 7/15), Batch size of 256 or 32 (repeated on 5/15), Neurons size of 512 (repeated on 10/15), Dropout of 0.2 (repeated on 11/15), and Learning rate of 1e-5 (repeated on 5/15). These are the optimal values of hyperparameters that we use for the rest of the work. The summary of optimal LSTM-AE hyperparameters that we use for the rest of the work is listed in Table 5. Similarly, we followed the same experiment manner of the grid search to find optimal hyperparameters for the other four machine-learning one-class detectors: OCSVM, LOF, IF, and EE. We train each one-class detector using various values of the following parameters: kernel, gamma, and nu for OCSVM; number of neighbors and novelty for LOF; number of estimators for IF; contamination and assume-centered for EE. We apply the grid search and evaluate performance to get the optimal parameter settings for each one-class classification algorithm. The optimal parameter settings of each one-class detector are listed in Table 5.

*6.4. Experiment 3: Investigating **RQ2***

In this section, we evaluate the effectiveness of the MotionID system under the practical requirements that we proposed, such as considering dynamic sensory data, users freely using the entire phone *over-all-apps* (not only on a specific app), unconditional activities, and using the single-device-evaluation approach. The purpose of this experiment is to take into consideration the effect of data drift and changes that occur in users' behaviors as well as evaluate the MotionID system on the five one-class detectors using their optimal hyperparameters obtained from previous experiments, shown in Table 5. So, we collected new data *Dataset-C* from the same participants, but after several weeks gap from collecting *Dataset-B*. For each device, we split the legitimate user data into 50% for model training and the remaining 50% used as unseen normal samples for the evaluation. We also used 50% of the imposter collected data after interacting on the same device to implement practical evaluation with balanced datasets for real-world situations. We emphasize that the dataset used for this evaluation was collected while the users were freely doing their activities, such as walking, standing, or sitting, without any constraints. In addition, the users were not asked to write pre-determined texts during touching smartphones or accessing a specific application.

This reflects the practical unconditional usage of smartphones and the authentication model implicitly and continuously works to detect malicious events. We run an experiment for each one-class detector using its optimal hyperparameters and report the performance results in Table 6. The results show each detector's F1-score and training time in seconds (in brackets) for each smartphone. We also provided the overall median F1 score for each device across all detectors and the overall median for all devices across each detector.

Table 6
Median F1 scores when considering practical requirements and drift concept on different devices and one-class detectors.

Device\Detector	F1-score (Train time in seconds)					
	LSTM-AE	OCSVM	LOF	IF	EE	Median (Detector)
LG LM-Q510N	87.47 (377.9)	88.82 (2.2)	88.32 (3.9)	89.11 (0.7)	**89.76 (5.8)**	**88.82**
Samsung Note5	85.55 (302.8)	73.57 (1.4)	74.62 (4.0)	80.01 (0.07)	82.83 (3.8)	80.01
Samsung Note10	68.89 (381.0)	68.94 (2.9)	68.94 (6.4)	68.94 (1.5)	68.94 (4.8)	68.94
Razer Phone2	81.89 (326.7)	82.20 (1.9)	81.90 (4.6)	83.43 (0.07)	81.90 (4.5)	81.90
Samsung Wide4	82.61 (357.9)	83.12 (2.7)	76.95 (5.9)	80.98 (0.07)	82.97 (6.09)	82.61
Median (Device)	**82.62 (349.26)**	82.20 (2.2)	76.95 (4.96)	80.98 (0.48)	**82.83 (4.99)**	

Table 7
Median F1-scores and standard deviations (std) on different short touching durations.

Detector	Device	Median F1-score (std)								
		1.0 s	1.5 s	2.0 s	2.5 s	3.0 s	3.5 s	4.0 s	4.5 s	5.0 s
All detectors	LG LM-Q510N Q_N = 101, Q_A = 95	(132 drt)	(88 drt)	(66 drt)	(53 drt)	(44 drt)	(38 drt)	(33 drt)	(29 drt)	(26 drt)
		79.01 (18.35)	79.11 (18.34)	79.11 (18.26)	79.15 (17.96)	79.14 (17.80)	79.12 (17.62)	79.15 (17.28)	79.26 (17.39)	79.43 (17.05)
	Samsung Note5 Q_N = 66, Q_A = 73	(183 drt)	(122 drt)	(91 drt)	(73 drt)	(61 drt)	(52 drt)	(45 drt)	(40 drt)	(36 drt)
		74.67 (13.14)	79.02 (12.98)	80.99 (12.78)	81.21 (12.75)	81.32 (12.66)	81.32 (12.49)	81.57 (12.28)	80.36 (11.92)	80.77 (12.28)
	Samsung Note10 Q_N = 68, Q_A = 69	(239 drt)	(157 drt)	(118 drt)	(95 drt)	(79 drt)	(67 drt)	(59 drt)	(52 drt)	(47 drt)
		69.24 (8.00)	69.25 (7.96)	69.41 (7.97)	69.51 (8.01)	69.47 (7.94)	69.50 (7.66)	69.48 (7.75)	69.48 (7.79)	69.49 (7.59)
	Razer Phone2 Q_N = 74, Q_A = 69	(172 drt)	(114 drt)	(86 drt)	(68 drt)	(56 drt)	(48 drt)	(42 drt)	(37 drt)	(34 drt)
		64.76 (5.08)	64.77 (4.99)	64.94 (5.04)	65.03 (4.48)	65.00 (4.97)	65.06 (4.02)	65.10 (4.17)	65.07 (3.942)	65.04 (4.70)
	Samsung Wide4 Q_N = 78, Q_A = 81	(186 drt)	(123 drt)	(92 drt)	(73 drt)	(61 drt)	(52 drt)	(46 drt)	(41 drt)	(36 drt)
		78.91 (19.18)	78.86 (18.68)	78.78 (18.31)	78.60 (17.94)	78.62 (17.96)	78.62 (17.62)	78.46 (17.60)	78.69 (17.59)	78.07 (16.95)

By analyzing the results, we can see that the LG LM-Q510N shows the highest F1-scores (ranging from above 87% to roughly 90%), while Samsung Note10 shows the lowest F1-scores (around 69%), using all five one-class detectors. Following, the Samsung Note5 device provides the second-best performance (ranging from above 73.57% to 85.55%), the Razer Phone2 device provides the third-best performance (ranging from above 81% to 83.43%), and the Samsung Wide4 device provides the fourth-best performance (ranging from above 76.95% to 83.12%). Then, we computed the median performance of each one-class detector over all devices and found that the detectors LSTM-AE and EE are the best, with an F1 score of 82.62% and 82.83%, respectively. Similarly, we computed the median performance of each smartphone over the five one-class detectors and found that the LG LM-Q510N shows the best F1 score with 88.82%. We observed that the drift concept of the data (collected after several weeks of behavior changes) affects the performance to slightly dropped compared with previous results. We believe that the drop can be compensated by retraining models and hyperparameters occasionally. Furthermore, we computed the consumed time for the training model for each one-class detector using the dataset collected only from legitimate users on their devices. We found that training LSTM-AE deep-learning models takes the longest time (ranging from 5 to 6 min) compared to other machine-learning detectors. This is expected since bidirectional LSTM networks have to learn the patterns deeply in forward and backward directions through all hidden layers and neurons. However, the isolation forest (IF) algorithm is the fastest one taking less than 1.5 s, while other ML algorithms take time ranging from one to less than 7 s to train models on the same amount of dataset.

*6.5. Experiment 4: Investigating **RQ3***

Here, we evaluate the effectiveness of the MotionID under the rapid authentication feature, in which test data is collected within short interaction durations (drt). It is important to check the accuracy of the model when authenticating a user (legitimate or imposter) using a few samples of sensory data collected within a short time of touching (*e.g.,* 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, and 5 s) to quickly detect the identity and take a decision (lock/unlock) smartphone accordingly. This kind of practical evaluation of models' performance for rapid authentication is required for real-world scenarios since an attacker often needs a few seconds to explore the victim's device, steal his private information, or make harmful threats. To do this, we first trained five one-class detectors with only the legitimate users' dataset in the same manner as the previous section, then we split test data into short-duration pieces of data samples based on the estimated value of *global mobile average (Q)* for the two users (legitimate and imposter) on each of the five smartphones. From our experiments, the estimated Q values are different for each user, even when using the same device, because the overall readings per second of the five sensors are directly related to the user's behaviors of how he/she touches and holds the

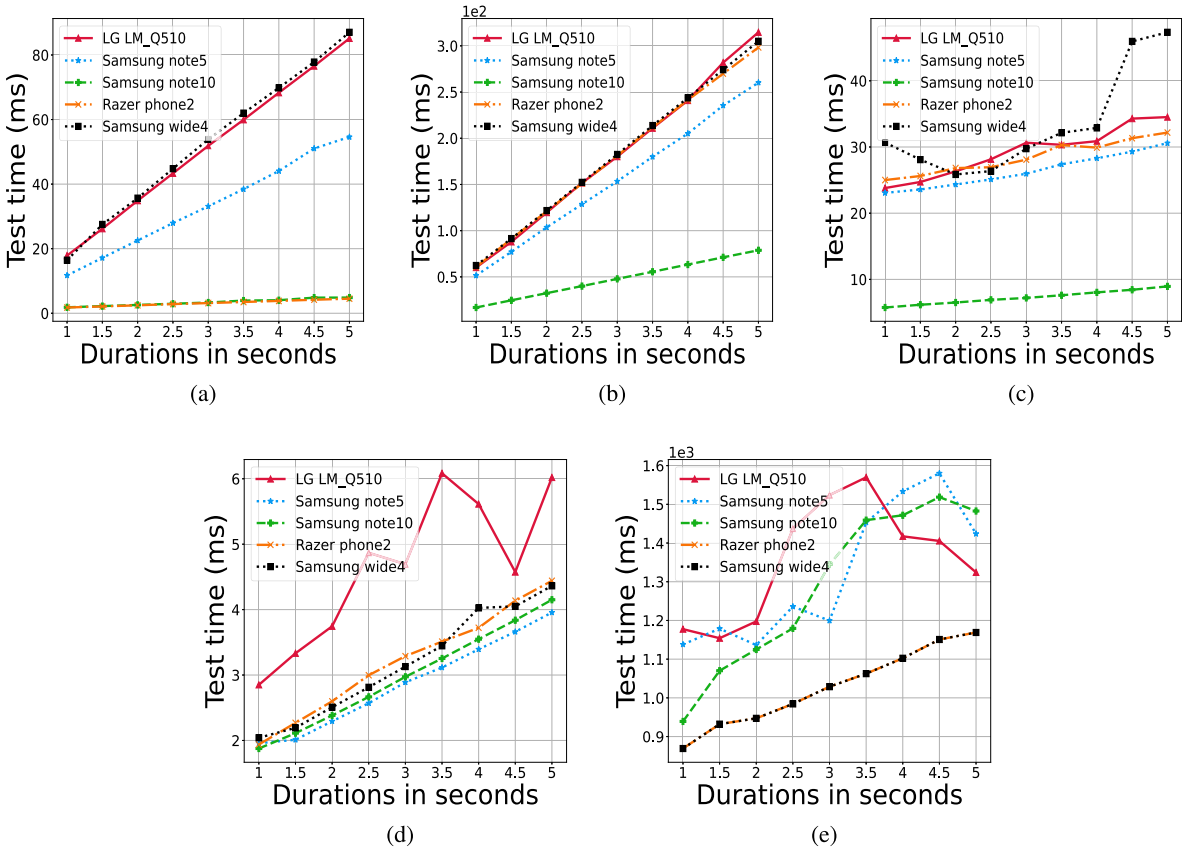


Fig. 9. Test time complexity of different detectors: (a) OCSVM (b) LOF (c) IF (d) EE (e) LSTM-AE.

smartphone. We estimated samples per second of normal user (Q_N) and of attacker user (Q_A) on the five devices as follows: [(LG LM-Q510N, $Q_N = 101$, $Q_A = 95$), (Samsung Note5, $Q_N = 66$, $Q_A = 73$), (Samsung Note10, $Q_N = 68$, $Q_A = 69$), (Razer Phone2, $Q_N = 74$, $Q_A = 69$), (Samsung Wide4, $Q_N = 78$, $Q_A = 81$)]. After that, we calculated the number of short durations (drt) for each user's test data.

Table 7 reported the median F1-score results of the experiment when five smartphones were evaluated using five one-class detectors. The number of short durations (drt), F1 scores, and standard deviation values are estimated through the 9 short duration periods (*i.e.,* 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5). The number of short durations decreases as the duration of touching smartphones increases. For example, legitimate and imposter users have been evaluated using data from 132 short durations, each of them including sensory data collected in one second of touching on the LG LM-Q510N device, and have achieved a median F1-score of 79.01% over five anomaly detectors. However, the performance was slightly increased to 79.43% when evaluated using data from 26 short durations of 5 s. We noticed that the performance of authentication on most devices increases (slightly) as the touching period increases from one to five seconds as the model gets more data for understanding the user's behavior and then produces good decisions. We also estimated the standard deviation (std) of short durations to check how dispersed the performance is in relation to the mean. For example, we evaluated the model for 132 short durations of a one-second touch period and got a median performance of 79.01% and 18.35 std. A low standard deviation indicates that the performance of short durations is clustered around the mean; the opposite is true for high standard deviations. In the end, we conclude that although the reported results showed a reasonable level of performance of the models, it shows the feasibility of developing rapid and practical authentication systems using a small number of data samples collected within a few seconds.

6.6. Experiment 5: Investigating ***RQ4***

During the experiments of this section, we investigate the effectiveness of the MotionID system in terms of time efficiency, power consumption, and memory usage.

Time complexity. Regarding detector complexity, we estimated the authentication time for the five smartphones using one-class detectors when varying short touch durations from one second to five seconds and reported the results in Fig. 9. In other words, we calculated how much time a trained model takes to give a prediction and detect the user's identity from the time he/she finishes

Table 8
Evaluation of power consumption when MotionID is continuously sensing.

Phone	OS (API version))	Battery level		Time		Period	Power consumed by MotionID app
		Before	After	Start	Stop		
Samsung S8	Android 9 (API 28)	100%, 3000 mAh	51%, 1530 mAh	3:36 PM	9:01 PM	5h 25 m	5.73% (84.23 mAh)
Samsung Note5	Android 6 (API 23)	100%, 3000 mAh	66%, 1980 mAh	3:49 PM	10:54 PM	7h 05 m	8.70% (88.74 mAh)

Table 9
Memory size of MotionID models using different detectors and devices.

Memory usage (Model size) Device	LSTM-AE	OCSVM	LOF	IF	EE
LG LMQ510	3.024 MB	146 KB	2.8 MB	421 KB	138 KB
Samsung Note5	3.028 MB	142 KB	2.6 MB	441 KB	126 KB
Samsung Note10	3.028 MB	189 KB	3.6 MB	430 KB	167 KB
Razer Phone2	3.028 MB	149 KB	2.77 MB	361 KB	132 KB
Samsung Wide4	3.024 MB	169 KB	3.34 MB	367 KB	160 KB

touching the smartphone for a specific short duration. Our results show that as durations in seconds increase, the models take longer to give predictions in milliseconds for most smartphones for all anomaly detectors — except the LM-Q510N device shows some fluctuations when increasing durations time towards 5 s. In addition, we observed that the amount of time complexity varies from one detector to another for the five devices as follows. The best three detectors with less time complexity are the EE detector, which shows the fastest testing time with less than 6 ms, the IF detector with less than 50 ms, and the OCSVM detector with less than 90 ms. However, the LOF and LSTM-AE detectors show more testing time complexity, reaching 300 ms and 1600 ms, respectively — because LOF needs to measure the local deviation of a given sample with respect to its 1000 neighbors, and LSTM-AE needs to deeply access past and future data samples within network neurons in bi-directional learning.

Power consumption. MotionID system involves continuously and implicitly sensing sensors using the data-collector application on smartphones while a user is in the session. To avoid unnecessary sensing when the device is off-session, we designed MotionID to be triggered for sensing only when the action of "user is present" is received using *BroadcastReceiver* interface, which indicates that the device is unlocked and the user is in the session. However, a user may stay logged in to the session for a long time – *e.g.,* when browsing or watching movies for a couple of hours. Therefore, we use Android Battery Historian[6] to profile MotionID's power consumption for continuous sensor sensing. To show examples of power consumption on smartphones, we selected two smartphones of different models. In detail, we charged the batteries of the Galaxy S8 and Galaxy Note5 devices to their full capacity of 3000 mAh and kept all other smartphones' services running normally while the MotionID application was continuously collecting data from all sensors for the time lengths of 5 h 25 m and 7 h 05 m, respectively. We then collected battery logs from smartphones and visualized battery historian reports using the Docker toolbox, which displayed the battery level when the phones were discharging. Table 8 shows that the power estimates consumed by MotionID are only 5.73% (84.23 mAh) and 8.70% (88.74 mAh) for Galaxy S8 and Galaxy Note5, respectively. We conclude that MotionID is acceptable for daily authentication, especially with the help of the locking/unlocking triggering methods, which provide activity detection to avoid unnecessary scanning of sensors and save battery.

Memory usage. To develop a practical authentication system that holds trained deep-learning and machine-learning models for being useful in the real world, it is necessary to ensure the accessibility of MotionID system on smartphones. Specifically, when deploying authentication models, it is important to monitor the memory usage of the size of the models on the device. Therefore, we calculated the sizes of trained models and reported the results in Table 9. The models' sizes vary from hundreds of KBs to a few MBs based on the type of the used classifier. In detail, OCSVM, IF, and EE detectors show small sizes of models while LOF and LSTM-AE detectors show less than 4 MB of model sizes. Therefore, the average memory usage of the MotionID application when deploying on a smartphone is almost very little when compared to other applications. This emphasizes the usability of the MotionID system for real-world deployment on smartphones.

6.7. Discussion

1. **Number of sensors**: Since our work depends on data collected from five sensors (Accelerometer, Elevation, Gyroscope, Gravity, and Magnetometer) to increase the diversity of time-series readings that perfectly reflect user's behaviors on smartphones, we also clarify the feature importance of each sensor in case of selecting some of them for the work. In practice, we implemented feature importance evaluation on legitimate users' datasets on the five smartphones using three different classifiers (Decision tree, Random Forest, and XGB). The feature index from 0 to 12 indicates sensor vectors (Acc: x,y,z; Elev: x; Grav: x,y,z; Gyro: x,y,z; Mag: x,y,z). Fig. 10 illustrates values of each sensor vector importance on five smartphones using Random Forest classifier. It shows that no specific sensor has superiority over other sensors when we look across all devices.

[6] https://github.com/google/battery-historian

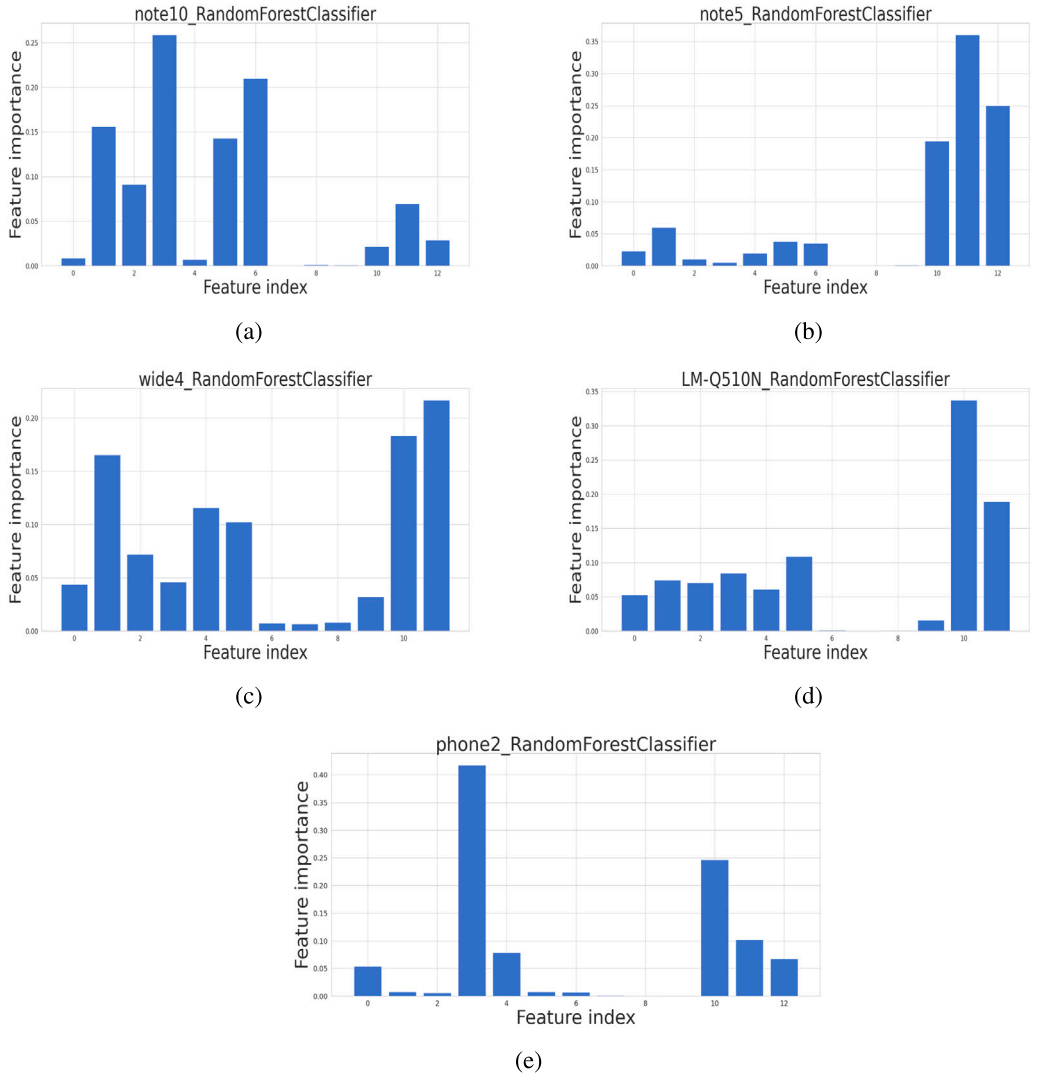


Fig. 10. Illustration of feature importance of sensory data on five devices using Random Forest classifier.

For example, for Samsung wide4 and note5 devices, Magnetometer and Accelerometer show higher importance values than other sensors. However, for other devices, this is changed where elevation and gravity sensors show higher importance values. To clarify this more, we also run experiments on two additional classifiers (Decision tree and XGB) and reported the results in Table 10, bold sensors are the most important. They show another feature importance order of sensors, which differs from one device to another and from one classifier to another. Therefore, since experiments show no clear answer about which sensor is much important, we could not select some of them and decided to use data from the five sensors (or four sensors as in some devices) as they are all useful for capturing user behaviors for authentication.

2. **Authentication operation design**: In this work, we utilized client–server implementation to transfer the location data between the client (smartphone) and the server. In other words, data processing, creating data segments, and training models of MotionID are performed for each user independently on the server. However, we did not directly perform these components of MotionID on smartphones because of the following reasons. (1) This stage of the work aimed to first examine the feasibility of the proposed objectives such as developing an authentication system with practical requirements and needs. (2) As known that training neural networks is a hard and time-consuming task. MotionID models are based on several LSTM layers and machine learning classifiers. In our understanding, small SoC architecture, a small-size GPU, and medium-sized DRAM of smartphones are reasons to prevent training neural networks from being deployed locally on the device. Nowadays, there are many AI-backed apps on phones (*e.g.,* Google Assistant and Apple Siri) that use client–server design. They upload all heavy

Table 10
Analysis of feature importance on sensory data on five devices using three different classifiers.

Feature Importance (Bold is most important)			
Device	Decision Tree Classifier	RandomForestClassifier	XGBClassifier
Samsung Wide4	**Acc, Mag,** Gyro, Grav	**Mag, Acc, Grav,** Elev, Gyro	**Mag, Acc, Elev,** Gyro,Grav
Samsung Note5	**Mag,** Acc, Elev, Grav, Gyro	**Mag, Acc,** Grav, Elev, Gyro	**Mag,** Acc, Elev, Grav, Gyro
Samsung Note10	**Elev, Acc, Mag,** Gyro, Grav	**Acc, Grav,** Mag, Elev, Gyro	**Elev, Acc, Mag,** Gyro, Grav
Razer Phone2	**Elev,** Acc, Mag, Gyro, Grav	**Elev, Mag, Acc,** Grav, Gyro	**Elev,** Acc, Mag, Gyro, Grav
LG LMQ510N	**Mag, Grav,** Acc, Gyro	**Mag, Acc, Grav,** Gyro	**Mag, Acc,** Grav, Gyro

AI tasks to a data center, send all users inputs into the data center, processed by the servers, and the output will return back to the user. Therefore, at this time, we utilized client–server design implementation, but we load trained models from the server into devices for on-device authentication. We leave on-smartphone training design for future work.

3. **Large-scale data collection**: Unlike existing works that collect data of owner and attacker users from different devices to train authentication models, we believe this is impractical as we do not know whether the classification results are based on variations in user behaviors or variability of sensory data acquired from various devices. Realistically, an attacker who wants to steal victims' information needs to interact with the same device that the victim uses. Therefore, we should evaluate the authentication system using data collected from different users on the same device. One of the challenges that we faced in this work is recruiting many users for collecting data over a large number of devices. Actually, since one of the work objectives is to let owner and attacker users use the same smartphone and collect their sensory data while they are opening any app, browsing, and doing any action on the device; this raised privacy concerns for the participants because they do not want strangers to explore their private devices. We then recruit ten users on five independent smartphones from various brands to collect datasets. We believe that the amount of used devices and users is sufficient to show the feasibility of the work. We will find a way to recruit more users in the future version of this work.

7. Conclusion

Recently and because of the affordability and availability of smartphones, security solutions such as continuous authentication have received significant attention to protect our sensitive and personal information that is increasingly loaded overtime on smartphones. The packaged sensors of smartphones play a notable role in differentiating the user's behavioral attributes (the way a user interacts with the phone) to continuously and implicitly authenticate his/her identity. However, existing authentication studies have serious limitations, preventing them from deploying an authentication solution on devices due to many practical requirements and constraints in real-world applications. In this work, we explored real-world requirements to bridge the gap towards developing a practical implicit authentication solution, MotionID, based on behavioral-based biometrics for deployment. MotionID presents the concept of dynamic sampling rates using *global mobile average (Q)* to independently authenticate a user based on his/her behaviors and the quality of the device's sensors. In addition, our system presents new advantages that support developing realistic authentication schemes for real-world usage: *unconditional settings*, *over-all-apps*, *single-device-evaluation*, *rapid authentication*, and *one-class detectors*. We conducted extensive experiments to evaluate MotionID on real-world datasets collected from five smartphones under various experimental conditions. In future work, we plan to extend the work by conducting additional experiments on a large number of devices, improving overall performance through exploiting more features and developing on-device models to run real authentications on smartphones.

CRediT authorship contribution statement

Mohsen Ali Alawami: Conceptualization, Data curation, Investigation, Methodology, Writing – original draft, Writing – review & editing, Formal analysis, Software, Validation. **Tamer Abuhmed:** Conceptualization, Methodology, Writing – original draft, Supervision. **Mohammed Abuhamad:** Investigation, Methodology, Supervision. **Hyoungshick Kim:** Conceptualization, Funding acquisition, Investigation, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgment

We thank the anonymous reviewers for their constructive comments. Hyoungshick Kim is the corresponding author. This work was supported by the Korea Internet & Security Agency (KISA) grant (No. 1781000003), and the Institute of Information & communications Technology Planning & Evaluation (IITP) grants (No. 2018-0-00532, Development of High-Assurance (>=EAL6) Secure Microkernel (50%), 2022-0-01199, and RS-2023-00229400) funded by the Korean government. The leading author of this work conducted this study at Sungkyunkwan University.

References

[1] Sebastian Uellenbeck, Markus Dürmuth, Christopher Wolf, Thorsten Holz, Quantifying the security of graphical passwords: The case of android unlock patterns, in: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, 2013, pp. 161–172.
[2] Abdulaziz Alzubaidi, Jugal Kalita, Authentication of smartphone users using behavioral biometrics, IEEE Commun. Surv. Tutor. 18 (3) (2016) 1998–2026.
[3] Victor Chang, Muthu Ramachandran, Towards achieving data security with the cloud computing adoption framework, IEEE Trans. Serv. Comput. 9 (1) (2015) 138–151.
[4] Jinho Seol, Seongwook Jin, Daewoo Lee, Jaehyuk Huh, Seungryoul Maeng, A trusted IaaS environment with hardware security module, IEEE Trans. Serv. Comput. 9 (3) (2016) 343–356.
[5] Geumhwan Cho, Jun Ho Huh, Junsung Cho, Seongyeol Oh, Youngbae Song, Hyoungshick Kim, Syspal: System-guided pattern locks for android, in: 2017 IEEE Symposium on Security and Privacy, SP, IEEE, 2017, pp. 338–356.
[6] Report: 1.25 billion fingerprint-enabled smartphones shipped in 2020, 2020, https://www.digitimes.com/news/a20180110PD211.html.
[7] Kai Cao, Anil K. Jain, Learning fingerprint reconstruction: From minutiae to image, IEEE Trans. Inf. Forensics Secur. 10 (1) (2014) 104–117.
[8] Ctirad Sousedik, Christoph Busch, Presentation attack detection methods for fingerprint recognition systems: a survey, Iet Biom. 3 (4) (2014) 219–233.
[9] Tarang Chugh, Kai Cao, Anil K. Jain, Fingerprint spoof buster: Use of minutiae-centered patches, IEEE Trans. Inf. Forensics Secur. 13 (9) (2018) 2190–2202.
[10] Furkan Tari, A. Ant Ozok, Stephen H. Holden, A comparison of perceived and real shoulder-surfing risks between alphanumeric and graphical passwords, in: Proceedings of the Second Symposium on Usable Privacy and Security, 2006, pp. 56–66.
[11] Adam J. Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, Jonathan M. Smith, Smudge attacks on smartphone touch screens, in: 4th USENIX Workshop on Offensive Technologies, WOOT 10, 2010.
[12] Seunghun Cha, Sungsu Kwag, Hyoungshick Kim, Jun Ho Huh, Boosting the guessing attack performance on android lock patterns with smudge attacks, in: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, 2017, pp. 313–326.
[13] Florian Schaub, Ruben Deyhle, Michael Weber, Password entry usability and shoulder surfing susceptibility on different smartphone platforms, in: Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia, 2012, pp. 1–10.
[14] Gloria Dhandapani, Jamie Ferguson, Euan Freeman, HapticLock: Eyes-free authentication for mobile devices, in: Proceedings of the 2021 International Conference on Multimodal Interaction, ICMI '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 195–202.
[15] So Higashikawa, Tomoaki Kosugi, Shogo Kitajima, Masahiro Mambo, Shoulder-surfing resistant authentication using pass pattern of pattern lock, IEICE Trans. Inf. Syst. 101 (1) (2018) 45–52.
[16] Davide Balzarotti, Marco Cova, Giovanni Vigna, Clearshot: Eavesdropping on keyboard input from video, in: 2008 IEEE Symposium on Security and Privacy, Sp 2008, IEEE, 2008, pp. 170–183.
[17] Diksha Shukla, Rajesh Kumar, Abdul Serwadda, Vir V. Phoha, Beware, your hands reveal your secrets!, in: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, 2014, pp. 904–917.
[18] Guixin Ye, Zhanyong Tang, Dingyi Fang, Xiaojiang Chen, Kwang In Kim, Ben Taylor, Zheng Wang, Cracking android pattern lock in five attempts, in: Proceedings of the 2017 Network and Distributed System Security Symposium 2017, NDSS 17, Internet Society, 2017.
[19] I Standard, Information Technology—Biometric Presentation Attack Detection—Part 1: Framework, ISO, Geneva, Switzerland, 2016.
[20] Rodrigo Frassetto Nogueira, Roberto de Alencar Lotufo, Rubens Campos Machado, Fingerprint liveness detection using convolutional neural networks, IEEE Trans. Inf. Forensics Secur. 11 (6) (2016) 1206–1213.
[21] Ajita Rattani, Arun Ross, Automatic adaptation of fingerprint liveness detector to new spoof materials, in: IEEE International Joint Conference on Biometrics, IEEE, 2014, pp. 1–8.
[22] Hoyeon Lee, Seungyeon Kim, Taekyoung Kwon, Here is your fingerprint! actual risk versus user perception of latent fingerprints and smudges remaining on smartphones, in: Proceedings of the 33rd Annual Computer Security Applications Conference, 2017, pp. 512–527.
[23] Chao Shen, Yuanxun Li, Yufei Chen, Xiaohong Guan, Roy A. Maxion, Performance analysis of multi-motion sensor behavior for active smartphone authentication, IEEE Trans. Inf. Forensics Secur. 13 (1) (2017) 48–62.
[24] Wei-Han Lee, Ruby B. Lee, Implicit smartphone user authentication with sensors and contextual machine learning, in: 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN, IEEE, 2017, pp. 297–308.
[25] Mohsen Ali, Tamer ElBatt, Moustafa Youssef, SenseIO: Realistic ubiquitous indoor outdoor detection system using smartphones, IEEE Sens. J. 18 (9) (2018) 3684–3693.
[26] Mohammed Abuhamad, Tamer Abuhmed, David Mohaisen, DaeHun Nyang, Autosen: Deep-learning-based implicit continuous authentication using smartphone sensors, IEEE Internet Things J. 7 (6) (2020) 5008–5020.
[27] Mohsen A. Alawami, Hyoungshick Kim, LocAuth: A fine-grained indoor location-based authentication system using wireless networks characteristics, Comput. Secur. 89 (2020) 101683.
[28] Mohsen A. Alawami, William Aiken, Hyoungshick Kim, The light will be with you. Always–A novel continuous mobile authentication with the light sensor (poster), in: Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services, 2019, pp. 560–561.
[29] Mohammed Abuhamad, Ahmed Abusnaina, DaeHun Nyang, David Mohaisen, Sensor-based continuous authentication of smartphones' users using behavioral biometrics: A contemporary survey, IEEE Internet Things J. 8 (1) (2020) 65–84.
[30] Mohsen Ali Alawami, Aishwarya Ram Vinay, Hyoungshick Kim, LocID: A secure and usable location-based smartphone unlocking scheme using Wi-Fi signals and light intensity, IEEE Internet Things J. 9 (23) (2022) 24357–24372.
[31] Chao Shen, Yufei Chen, Xiaohong Guan, Performance evaluation of implicit smartphones authentication via sensor-behavior analysis, Inform. Sci. 430 (2018) 538–553.
[32] Attaullah Buriro, Bruno Crispo, Sandeep Gupta, Filippo Del Frari, Dialerauth: A motion-assisted touch-based smartphone user authentication scheme, in: Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy, 2018, pp. 267–276.
[33] Cong Wu, Kun He, Jing Chen, Ziming Zhao, Ruiying Du, Liveness is not enough: Enhancing fingerprint authentication with behavioral biometrics to defeat puppet attacks, in: 29th USENIX Security Symposium, USENIX Security 20, 2020, pp. 2219–2236.
[34] Wei-Han Lee, Ruby B. Lee, Multi-sensor authentication to improve smartphone security, in: 2015 International Conference on Information Systems Security and Privacy, ICISSP, IEEE, 2015, pp. 1–11.

[35] Pablo Fernandez-Lopez, Judith Liu-Jimenez, Carlos Sanchez-Redondo, Raul Sanchez-Reillo, Gait recognition using smartphone, in: 2016 IEEE International Carnahan Conference on Security Technology, ICCST, IEEE, 2016, pp. 1–7.
[36] Robertas Damaševičius, Rytis Maskeliūnas, Algimantas Venčkauskas, Marcin Woźniak, Smartphone user identity verification using gait characteristics, Symmetry 8 (10) (2016) 100.
[37] Ashika Ramesh Kothamachu, Basabi Chakraborty, Real time gait based person authentication using deep hybrid network, in: 2021 IEEE 4th International Conference on Knowledge Innovation and Invention, ICKII, IEEE, 2021, pp. 155–159.
[38] Xinchen Zhang, Yafeng Yin, Lei Xie, Hao Zhang, Zefan Ge, Sanglu Lu, Touchid: User authentication on mobile devices via inertial-touch gesture analysis, Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 4 (4) (2020) 1–29.
[39] Adnan Bin Amanat Ali, Vasaki Ponnusamy, Anbuselvan Sangodiah, Roobaea Alroobaea, N.Z. Jhanjhi, Uttam Ghosh, Mehedi Masud, Smartphone security using swipe behavior-based authentication, Intell. Autom. Soft Comput. 29 (2) (2021) 571–585.
[40] Mario Frank, Ralf Biedert, Eugene Ma, Ivan Martinovic, Dawn Song, Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication, IEEE Trans. Inf. Forensics Secur. 8 (1) (2012) 136–148.
[41] Max Smith-Creasey, Muttukrishnan Rajarajan, A novel word-independent gesture-typing continuous authentication scheme for mobile devices, Comput. Secur. 83 (2019) 140–150.
[42] Amith K. Belman, Vir V. Phoha, DoubleType: Authentication using relationship between typing behavior on multiple devices, in: 2020 International Conference on Artificial Intelligence and Signal Processing, AISP, IEEE, 2020, pp. 1–6.
[43] Soumik Mondal, Patrick Bours, Person identification by keystroke dynamics using pairwise user coupling, IEEE Trans. Inf. Forensics Secur. 12 (6) (2017) 1319–1329.
[44] Tanapat Anusas-Amornkul, Strengthening password authentication using keystroke dynamics and smartphone sensors, in: Proceedings of the 9th International Conference on Information Communication and Management, 2019, pp. 70–74.
[45] Maciej Szymkowski, Patryk Milewski, Khalid Saeed, Fingerprint and keystroke dynamics fusion in multimodal biometrics system, in: Advanced Computing and Systems for Security, Springer, 2021, pp. 67–82.
[46] Lerina Aversano, Mario Luca Bernardi, Marta Cimitile, Riccardo Pecori, Continuous authentication using deep neural networks ensemble on keystroke dynamics, PeerJ Comput. Sci. (2021).
[47] Md Liakat Ali, John V. Monaco, Charles C. Tappert, Meikang Qiu, Keystroke biometric systems for user authentication, J. Signal Process. Syst. 86 (2) (2017) 175–190.
[48] Chen Song, Aosen Wang, Kui Ren, Wenyao Xu, Eyeveri: A secure and usable approach for smartphone user authentication, in: IEEE INFOCOM 2016-the 35th Annual IEEE International Conference on Computer Communications, IEEE, 2016, pp. 1–9.
[49] R.C. Johnson, Walter J. Scheirer, Terrance E. Boult, Secure voice-based authentication for mobile devices: vaulted voice verification, in: Biometric and Surveillance Technology for Human and Activity Identification X, vol. 8712, International Society for Optics and Photonics, 2013, p. 87120P.
[50] Juan Manuel Espín López, Alberto Huertas Celdrán, Javier G. Marín-Blázquez, Francisco Esquembre, Gregorio Martínez Pérez, S3: An AI-enabled user continuous authentication for smartphones based on sensors, statistics and speaker information, Sensors 21 (11) (2021) 3765.
[51] Mikhail I. Gofman, Sinjini Mitra, Nicholas Smith, Hidden markov models for feature-level fusion of biometrics on mobile devices, in: 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications, AICCSA, IEEE, 2016, pp. 1–2.
[52] Sara Amini, Vahid Noroozi, Amit Pande, Satyajit Gupte, Philip S. Yu, Chris Kanich, Deepauth: A framework for continuous user re-authentication in mobile apps, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018, pp. 2027–2035.
[53] Alina Garbuz, Anna Epishkina, Konstantin Kogos, Continuous authentication of smartphone users via swipes and taps analysis, in: 2019 European Intelligence and Security Informatics Conference, EISIC, IEEE, 2019, pp. 48–53.
[54] Michail D. Papamichail, Kyriakos C. Chatzidimitriou, Thomas Karanikiotis, Napoleon-Christos I. Oikonomou, Andreas L. Symeonidis, Sashi K. Saripalle, Brainrun: A behavioral biometrics dataset towards continuous implicit authentication, Data 4 (2) (2019) 60.
[55] Yantao Li, Hailong Hu, Zhangqian Zhu, Gang Zhou, SCANet: sensor-based continuous authentication with two-stream convolutional neural networks, ACM Trans. Sensor Netw. 16 (3) (2020) 1–27.
[56] Mario Parreño Centeno, Yu Guan, Aad van Moorsel, Mobile based continuous authentication using deep features, in: Proceedings of the 2nd International Workshop on Embedded and Mobile Deep Learning, 2018, pp. 19–24.
[57] Yantao Li, Peng Tao, Shaojiang Deng, Gang Zhou, DeFFusion: CNN-based continuous authentication using deep feature fusion, ACM Trans. Sensor Netw. 18 (2) (2021) 1–20.
[58] Giuseppe Stragapede, Ruben Vera-Rodriguez, Ruben Tolosana, Aythami Morales, Alejandro Acien, Gaël Le Lan, Mobile behavioral biometrics for passive authentication, Pattern Recognit. Lett. 157 (2022) 35–41.
[59] Tiantian Zhu, Zhengyang Qu, Haitao Xu, Jingsi Zhang, Zhengyue Shao, Yan Chen, Sandeep Prabhakar, Jianfeng Yang, RiskCog: Unobtrusive real-time user authentication on mobile devices in the wild, IEEE Trans. Mob. Comput. 19 (2) (2019) 466–483.
[60] Tiantian Zhu, Zhengqiu Weng, Guolang Chen, Lei Fu, A hybrid deep learning system for real-world mobile user authentication using motion sensors, Sensors 20 (14) (2020) 3876.
[61] Jakub Dybczak, Piotr Nawrocki, Continuous authentication on mobile devices using behavioral biometrics, in: 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing, CCGrid, IEEE, 2022, pp. 1028–1035.
[62] Sakorn Mekruksavanich, Anuchit Jitpattanakul, Deep learning approaches for continuous authentication based on activity patterns using mobile sensing, Sensors 21 (22) (2021) 7519.
[63] Weizhi Meng, Yu Wang, Duncan S Wong, Sheng Wen, Yang Xiang, TouchWB: Touch behavioral user authentication based on web browsing on smartphones, J. Netw. Comput. Appl. 117 (2018) 1–9.
[64] Yantao Li, Hailong Hu, Gang Zhou, Using data augmentation in continuous authentication on smartphones, IEEE Internet Things J. 6 (1) (2018) 628–640.
[65] Mingming Hu, Kun Zhang, Ruibang You, Bibo Tu, Multi-sensor-based continuous authentication of smartphone users with two-stage feature extraction, IEEE Internet Things J. (2022).
[66] Zahid Syed, Jordan Helmick, Sean Banerjee, Bojan Cukic, Touch gesture-based authentication on mobile devices: The effects of user posture, device size, configuration, and inter-session variability, J. Syst. Softw. 149 (2019) 158–173.
[67] Jinpei Yan, Yong Qi, Qifan Rao, Saiyu Qi, Towards a user-friendly and secure hand shaking authentication for smartphones, in: 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering, TrustCom/BigDataSE, IEEE, 2018, pp. 1170–1179.
[68] Asadullah Laghari, Zulfiqar Ali Memon, et al., Biometric authentication technique using smartphone sensor, in: 2016 13th International Bhurban Conference on Applied Sciences and Technology, IBCAST, IEEE, 2016, pp. 381–384.
[69] H. Feng, S. You, M. Wei, et al., MGRA: Motion gesture recognition via accelerometer, Sensors 16 (4) (2016) 1–25.
[70] Guangyuan Hu, Zecheng He, Ruby Lee, Smartphone impostor detection with built-in sensors and deep learning, 2020, arXiv preprint arXiv:2002.03914.
[71] Muhammad Ehatisham-ul Haq, Muhammad Awais Azam, Usman Naeem, Yasar Amin, Jonathan Loo, Continuous authentication of smartphone users based on activity pattern recognition using passive mobile sensing, J. Netw. Comput. Appl. 109 (2018) 24–35.
[72] Zdeňka Sitová, Jaroslav Šeděnka, Qing Yang, Ge Peng, Gang Zhou, Paolo Gasti, Kiran S. Balagani, HMOG: New behavioral biometric features for continuous authentication of smartphone users, IEEE Trans. Inf. Forensics Secur. 11 (5) (2015) 877–892.

[73] Maximilian Haring, Delphine Reinhardt, Yvonne Omlor, Pick me up and i will tell you who you are: Analyzing pick-up motions to authenticate users, in: 2018 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops, IEEE, 2018, pp. 472–475.
[74] Praveen Kumar Rayani, Suvamoy Changder, Continuous user authentication on smartphone via behavioral biometrics: a survey, Multimedia Tools Appl. (2022) 1–35.
[75] Ioannis Stylios, Spyros Kokolakis, Olga Thanou, Sotirios Chatzis, Behavioral biometrics & continuous user authentication on mobile devices: A survey, Inf. Fusion 66 (2021) 76–99.
[76] Chen Wang, Yan Wang, Yingying Chen, Hongbo Liu, Jian Liu, User authentication on mobile devices: Approaches, threats and trends, Comput. Netw. 170 (2020) 107118.
[77] S. Ayeswarya, Jasmine Norman, A survey on different continuous authentication systems, Int. J. Biom. 11 (1) (2019) 67–99.
[78] Mohsen Ali Alawami, William Aiken, Hyoungshick Kim, LightLock: User identification system using light intensity readings on smartphones, IEEE Sens. J. 20 (5) (2020) 2710–2721.
[79] Ahmed Mahfouz, Tarek M. Mahmoud, Ahmed Sharaf Eldin, A survey on behavioral biometric authentication on smartphones, J. Inf. Secur. Appl. 37 (2017) 28–37.
[80] Jun Ho Huh, Sungsu Kwag, Iljoo Kim, Alexandr Popov, Younghan Park, Geumhwan Cho, Juwon Lee, Hyoungshick Kim, Choong-Hoon Lee, On the long-term effects of continuous keystroke authentication: Keeping user frustration low through behavior adaptation, Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 7 (2) (2023) 1–32.
[81] Geumhwan Cho, Sungsu Kwag, Jun Ho Huh, Bedeuro Kim, Choong-Hoon Lee, Hyoungshick Kim, Towards usable and secure location-based smartphone authentication, in: Seventeenth Symposium on Usable Privacy and Security, SOUPS 2021, 2021, pp. 1–16.