IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 8, 2021, accepted February 24, 2021, date of publication March 2, 2021, date of current version March 15, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3063424*

# DAKOTA: Sensor and Touch Screen-Based Continuous Authentication on a Mobile Banking Application

**ÖZLEM DURMAZ INCEL[1], SEÇIL GÜNAY[1], YASEMIN AKAN[1,2], YUNUS BARLAS[2], OKAN ENGIN BASAR[1,2], GÜLFEM ISIKLAR ALPTEKIN[1], AND MUSTAFA ISBILEN[2]**

[1]Department of Computer Engineering, Galatasaray University, 34349 Istanbul, Turkey
[2]Yapı Kredi Teknoloji Istanbul, 34469 Istanbul, Turkey

Corresponding author: Özlem Durmaz Incel (odincel@gsu.edu.tr)

This work was supported by Tubitak under Grant 5170078.

**ABSTRACT** Authenticating a user in the right way is essential to IT systems, where the risks are becoming more and more complex. Especially in the mobile world, banking applications are among the most delicate systems requiring strict rules and regulations. Existing approaches often require point-of-entry authentication accompanied by a one-time password as a second-factor authentication. However, this requires active participation of the user and there is continuous authentication during a session. In this paper, we investigate whether it is possible to continuously authenticate users via behavioral biometrics with a certain performance on a mobile banking application. A currently used mobile banking application in Turkey is chosen as the case, and we developed a continuous authentication scheme, named DAKOTA, on top of this application. The DAKOTA system records data from the touch screen and the motion sensors on the phone to monitor and model the user's behavioral patterns. Forty-five participants completed the predefined banking transactions. This data is used to train seven different classification algorithms. The results reveal that binary-SVM with RBF kernel reaches the lowest error scores, 3.5% equal error rate (EER). Using the end-to-end DAKOTA system, we investigate the performance in real-time, both in terms of authentication accuracy and resource usage. We show that it does not bring extra overhead in terms of power and memory usage compared to the original banking application and we can achieve a 90% true positive recognition rate, on average.

**INDEX TERMS** Behavioral biometrics, continuous authentication, mobile applications, mobile sensing, sensor-based authentication, smartphone authentication.

## I. INTRODUCTION

Authenticating a user on a mobile phone is an essential operation since users store critical information, such as personal photos, contact details, call histories, private messages, login details and application data. Personal Identification Number (PIN), passwords, graphical patterns, physical biometrics (i.e., fingerprints) are the examples of active or explicit authentication. Active authentication approaches are often used when users launch the device and require the user's active participation. Users usually use simple, easily guessed PINs and passwords, making it very easy for the imposters to access the device contents [1]. In the case of graphical patterns, users often choose to use the same pattern and leave oily residues on the screen [2], and one can easily repeat the secret pattern.

Mobile banking applications are examples of applications where authentication is of critical importance and strict rules and regulations exist for providing it. Usually, a banking application requires a password at login; however, there are also examples of banking applications, which use physical biometrics (i.e., iris or face recognition) as the authentication mechanism. These approaches often require one-time or point-of-entry authentication. To complete several critical transactions (i.e., money transfer), a one-time password (OTP) is often required as a second-factor authentication. However, the user is still not continuously authenticated during a session and these approaches are limited in offering

The associate editor coordinating the review of this manuscript and approving it for publication was Vincenzo Conti.

implicit and transparent authentication [3]. If the user forgets to logout and the phone is captured by someone else, attackers may get access to his/her account and may even add their fingerprint or face ID to the system settings and access the application.

Considering the limitations of active or explicit authentication schemes, continuous authentication, also known as implicit, passive, or progressive authentication is emerging as an alternative or complementary solution [1], [3]–[5]. It is based on tracking the user's interaction patterns with a device and provides a continuous service. Continuous authentication uses behavioral biometrics, which refer to a specific behavioral pattern to a user [6]. In a continuous authentication scheme, the authentication process happens in real-time during the entire interaction. By reducing the requirement of explicit authentication, a more usable experience is provided to the users [7].

This paper proposes a behavioral biometrics-based continuous authentication system named DAKOTA, which is mainly designed for a mobile banking application. The application is from a local bank in Turkey and 6.4 million users actively use it. The DAKOTA[1] system is developed as a research project and the aim is to provide a continuous authentication scheme as an additional or complementary solution, similar to OTP. The introduced research question in this paper can be summarized as: 'Can we continuously and efficiently authenticate the users using behavioral biometrics, so that we can remove OTP requirements in certain transactions with low risk?'

We formulate the authentication process as a classification problem, either the subject is the real user or not, i.e., the imposter. For this purpose, the currently used mobile banking application is modified to log data from the touch screen and the motion sensors on the phone to monitor and model the user's behavioral patterns. The logger component collects touch data, such as *x*-*y* coordinates on the screen, finger pressure, and sensor data from accelerometer, gyroscope, and magnetometer to capture the user's hand movements. We collected a dataset from 45 participants, considering different usage scenarios, including different types of transactions, and postures while sitting, standing, and keeping the phone on a table. We designed several experiments to evaluate the impact of different classifiers, feature transformation and feature selection methods, sensor type, and posture on the authentication performance. According to the results of the experiments, binary-SVM (support vector machine) with RBF kernel is observed to reach the highest true positive recognition rate (TPR), 99%, and the lowest error scores, 3.5% equal error rate (EER).

We also implemented the end-to-end DAKOTA system, where the trained models are stored on a server and the data collected during a transaction is transferred to the server for authentication based on the user's biometric model. We tested the performance both in terms of authentication accuracy and resource usage. We initially obtained true recognition rates in terms of authentication accuracy, ranging between 40% and 80%, with the SVM classifier. We observe that, statistically, there are differences between the old training data and new test data due to changes in user's biometric metrics, as reported in the literature [8]–[10]. As a solution, we updated the models by adding new data and could achieve true recognition rates ranging between 64% and 98%. These results are obtained when each scroll is tested individually. However, during a session/transaction, there are multiple scroll events and some of them might be mis-classified and reporting the result of each scroll to the user is not practical. Hence, we also report each session's performance by using a threshold value for the minority class and show that we can achieve higher accuracy and lower error rates. In terms of resource consumption, we observed that the DAKOTA system increases power consumption by 27% compared with the original application; however, since the mobile banking application is not continuously used, this does not significantly affect the battery life and user experience. Power consumption and memory usage did not change significantly. Contributions and highlights of the paper are summarized as follows:

- Instead of using a dummy application, we modified a mobile banking application, which is currently used by many users, so that user behaviors are captured directly on the real application. During data collection, the most used transactions on the application are performed by the users.
- We tested the performance with a large set of parameters, including nine different classifiers, a large set of features, two different sampling algorithms, five different usage scenarios, and three different postures.
- Most of the related studies focus on building the biometrics models; however, the proposed systems do not investigate the resource consumption on mobile devices. We presented an end-to-end system and tested its performance, both in terms of authentication and resource consumption.
- The tests on the end-to-end system show that there can be changes in biometric features, which might reduce the predictive performance. In order to address this issue, we focused on how to update the models. Moreover, we discussed the performance with session-based authentication compared to authentication for each scroll.

This article also extends a preliminary study [11], in which a smaller version of the dataset (only 20 users) was utilized, and only one-class SVM algorithm was used in the analysis, with fewer experiments and a less detailed discussion. The rest of the paper is organized as follows: In Section II, we present and compare the related studies. Section III includes the details of the dataset along with the processing of the data. In Section IV, we introduce our methodology, particularly the details of the applied models, while in Section V, we present the performance analysis of the classifiers and the

[1] The acronym is in Turkish, meaning "authentication based on behavioral patterns."

results of real-time authentication. In Section VI, we discuss the limitations and possible extension of the study, and in Section VII, we draw the conclusions.

## II. RELATED WORK

Authentication is defined as the process or action of verifying the identity of a user or process. The authentication approaches for computer systems are categorized into three main classes: (1) knowledge-based authentication (e.g., password, PIN), (2) possession-based authentication (e.g., memory card, smart card tokens), and (3) biometric-based authentication (physiological or behavioral) [6]. The knowledge-based authentication, the most commonly used one on mobile devices, is based on unique and private information, which is expected to be known only by the user. The object-based/possesion-based authentication is based on possession of a specific physical object. Behavioral biometrics are based on users' learned movements, such as hand-writing, keystroke, touch screen, gait, signature, voice, and behavior profiling [4]. All these approaches enable analyzing users' interactions with the device and building a model, which decides to authenticate the current user.

In related literature, there are four survey papers [1], [3]–[5] that investigate the use of biometrics for continuous authentication on smartphones. The research in [5] has examined AI-based continuous authentication solutions proposed for IoT applications while the survey in [3] provides a detailed analysis of the approaches that mainly utilize embedded sensors in smartphones. The authors emphasized that sensors, such as camera, microphone, can be used to collect physical data, while components such as accelerometers, gyroscopes, touch screens can be used to collect behavioral biometric data, such as walking, screen touch gestures, and hand gestures [1]. In another review paper [4], the literature studies were examined in terms of the type and size of the data collected, classifiers used in identification, and results obtained. We should note that these papers also investigate the use of physical biometrics for authentication. In this paper, we specifically focus on studies using behavioral biometrics. In Table 1,[2] we present a summary of the comparison with the related studies. We elaborate on the comparisons at the end of this section. We should note that our aim is not to provide a detailed survey, and we cannot cover all the related studies in this paper, and there are several survey papers on the topic, as mentioned. We included example studies that are highly cited and mentioned in the survey papers and that are more recently published in the last 3-4 years.

### A. TOUCH-BASED AUTHENTICATION

Touch screens are used as input mediums on a great majority of smartphones. By applying classification algorithms on the data collected from users' interactions with the touch screen (micro-movements, pressure, finger movements, etc.), it is possible to recognize authorized users. Several related works examine password patterns [12], tapping behavior [13], or touch gestures [14] to create a model to decide whether the corresponding user is an authorized one or not.

An authentication scheme employing password, keystroke, and swipe dynamics together for touch screen mobile devices is proposed in [15]. Hence, the biometric-based authentication attributes are examined together with the knowledge-based authentication attributes. For sensitive data, in [16], an accurate and efficient continuous authentication method is presented using touch-based behavioral biometrics to save energy and protect user security. Users were asked to apply some specific touch gestures on the screen without any restriction or guideline to obtain more realistic raw data [17]. In SafeGuard [8], authors collect data from touch screen inputs, such as sliding dynamics and pressure intensity.

A continuous touch-based user authentication system that focuses on post-login user authentication is introduced in [18]. They have used a digital glove to complement and validate touch gesture data. A biometric-based system for smartphones that use the owner's finger movement patterns is presented in [19]. Multiple types of touch data are used to model a user by justifying their two properties: distinctiveness and permanence [20]. They stated that it is promising to implement a continuous authentication mechanism based only on the touch data collected during normal user operations.

The authors investigate whether a classifier can continuously authenticate users based on their interaction with the touch screen of their smartphones [21]. The proposed method is based on basic navigation movements, such as up-down and left-right scrolling. They suggest a set of 30 touch features extracted from raw data collected from the touch screen. They incorporated biometric touch sensing, which notifies running applications of 2D touch events and the user's identity for each touch [29]. Implementing the model, they use touch devices called Bioamp, a watch prototype that complements touch devices, senses biometric features, and conceptually converts them into an electric signal that mobile devices can detect upon touch.

### B. SENSOR-BASED AUTHENTICATION

One of the examples of sensor-based authentication is proposed in [23]. Authors aim to achieve authentication when an intruder physically accesses the device and possesses the pass codes to unlock it. A high-frequency deep learning-based approach using built-in sensors (accelerometer, gyroscope, and magnetometer) is examined in [25]. They have shown that the proposed method can authenticate users with a high F1-score, and the best authentication performance scores are achieved using all three sensors. The authors investigate the reliability and applicability of using motion-sensor behavior for active and continuous smartphone authentication. They use the accelerometer, gyroscope, orientation, and

[2] Abbreviations used in Table 1 are as follows: FRR: False Rejection Rate, FAR: False Acceptance Rate, EER: Equal Error Rate, TPR: True Positive Rate, kNN: k-Nearest Neighbors, SVM: Support Vector Machine, MLP: Multi-layer Perceptron, LDA: Linear Discriminant Analysis, RF: Random Forest, NB: Naive Bayes, LSTM: Long-short Term Memory, ROC: Receiver Operating Characteristic, Avg: Average

**TABLE 1.** A summary of comparison with related studies.

| Ref. | Input type | Sensors/Features | # of users | Classifier/Method | Performance | Resource usage |
|---|---|---|---|---|---|---|
| [12] | Touch | Pressure, finger size, x-y coordinate, time | 31 | Dynamic time wrapping | Accuracy=77%<br>FRR=19%<br>FAR=21% | - |
| [13] | Touch | Acceleration, pressure, finger size, time | 80 | One-class learning | EER=3.65% | Storage=11.195 MB<br>Preprocessing time= 0.496 s |
| [15] | Touch | Swiping and typing pattern features (130 features) | 31 | Naive Bayes, kNN, CART, SVM, MLP, logistic regression, LDA | Accuracy=89.67%<br>F1 score=88.61% | - |
| [8] | Touch | Angle, distance, pressure, intensity, time | 60 | SVM | FAR=0.03<br>FRR=0.05 | Storage=63 kB<br>Avg training+testing time= 3.1312 s |
| [21] | Touch | x-y coordinates, pressure, finger size, time, orientation (30 features) | 41 | kNN, SVM with Rbf-kernel | Median EER=0-4% | - |
| [14] | Touch | Proposed graphic touch gesture features, time, x-y coordinates, pressure | 30 | Distance-based method | EER=2.62% | - |
| [16] | Touch | x-y coordinates, angle, pressure, slope, velocity | 45 | One-class SVM, isolation forest | Accuracy=95.98% | - |
| [18] | Touch | Touchscreen, x-y coordinates, finger motion, finger motion speed, pressure, distance, sensor glow data (53 features) | 40 | Decision tree, Random Forest, BayesNET | FAR: 4.66%<br>FRR: 0.13% | - |
| [19] | Touch | Touchscreen, x-y coordinates, pressure, finger size, moving direction, distance, duration | 75 | SVM | ACC: 79.74-95.78% | Extracting features: 648 ms |
| [20] | Touch | Touchscreen, x-y coordinates, pressure, finger size | 32 | SVM | EER<10%<br>FAR, FRR, ROC | - |
| [22] | Sensor | Accelerometer, gyroscope, magnetometer, gravity (16 features) | 85 | BayesNET, Naive Bayes, SVM, kNN, J48, RF | True Acceptance RateRF =99.35%<br>AccuracyRF=98.98% | - |
| [23] | Sensor | Accelerometer, gyroscope, time | 48 | One-clss SVM, neural network, nearest neighbor | FRR=6.85%<br>FAR=5.01%<br>EER<12% | Storage=2 MB<br>Avg consumption per day=1.7% |
| [24] | Sensor | Accelerometer, orientation, magnetometer, gyroscope | 102 | Markov-based decision procedure using one-class learning, SVM, neural network | EER<10%<br>FAR,HMM=3.98%<br>FRR,HMM=5.03%<br>EER,HMM=4.71% | Storage=5.6 MB<br>Avg consumption per day=4.5% |
| [25] | Sensor | Accelerometer, gyroscope, magnetometer, screen touch, elevation | 84 | Deep learning/ LSTM | EER=0.09%<br>F1 score=98%<br>FAR= 0.96%<br>FRR= 8.08% | - |
| [26] | Sensor | Accelerometer, orientation, gravity, magnetometer, gyroscope, touch screen (142 features) | 95 | Naive Bayes, Neural network, RF | True Acceptance Rate=96% | - |
| [27] | Multimodal | Accelerometer, gyroscope, magnetometer, tap duration, contact size, keystroke features, x-y coordinates | 100 | One-class SVM | EER=15.1% | Overhead=6.2-20.5% |
| [28] | Multimodal | Accelerometer, orientation sensor, gyroscope, time, x-y coordinates, pressure, finger size | 26 | BayesNET, Naive Bayes, RF, sequential minimal optimization | FAR,NB=11.01%<br>FRR,NB=4.12% | - |
| DAKOTA | Multimodal | Touchscreen, accelerometer, magnetometer, gyroscope | 45 | SVM, KNN, Naive Bayes, MLP, Decision Tree, RF, Ensemble 1 (SVM and MLP) Ensemble 2 (SVM Polynomial kernel and SVM RBF kernel) | FAR: 0.04%<br>FRR: 3.88%<br>TPR: 96.12%<br>ACC: 99.88%<br>EER: 3.5 % | 27% increase in power, similar CPU and memory usage |

magnetometer readings while users perform touch-tapping and single-touch-sliding actions in three different scenarios, which were based on device position and users' activity (hand-hold, table-hold, and hand-hold-walk) [24].

In a different study, an authentication scheme is proposed for smartwatches and smartphones based on activities of daily living [30]. They focus on the movement patterns of these activities and use an accelerometer and gyroscope. It is shown that the best performance is achieved when both smartphone and smartwatch are used together, and also the accelerometer is reported to be the best performing sensor. This study does not focus on the interaction patterns of the users with the devices. Similarly, in another study, the authors propose a continuous authentication scheme for smart bands [31]. Instead of motion sensors, they use photoplethysmogram sensors, which are used to extract heart rate variability (HRV) features.

### *C. MULTIMODAL AUTHENTICATION*

A detailed list of modalities and features used in behavioral biometrics is given in [3]. The list includes i) gestures, such as hand-writing, touch gestures, ii) gait, iii) motion, such as hand movement, iv) voice, v) keystroke dynamics. This section includes the studies that only use a combination of touch gestures and hand movements from touchscreen and motion sensors. Considering the other modalities, text input, or keystroke dynamics are limited in our target mobile banking application, and there is no voice or gait input.

In the HMOG study [27] (Hand Movement, Orientation and Grasp), accelerometer, gyroscope, magnetometer readings, and tap-based features, such as x-y coordinates, finger covered area/finger size, pressure, etc., are collected. Besides the touchscreen-related data, the authors propose a new set of features, derived from micro-movements obtained from accelerometer, gyroscope, and magnetometer sensors data generated while users interact with the device. We also focused on the HMOG dataset in a previous study [32] and investigated the authentication performance with deep networks using different combinations of input data, namely only touchscreen data, sensor data, and their combination.

Another research [22] proposed a bi-modal behavioral biometric-based solution for smartphone user authentication by using an accelerometer, gyroscope, gravity, magnetometer, and touch screen while the user performed sliding and phone-lifting actions. They found that the Random Forest classifier proved to be the most consistent and accurate.

The authors propose a new multimodal biometric authentication model based on the features collected while the user slide-unlocks the smartphone to answer a call [28]. The features were populated by slide/swipe, arm movements of the user answering a call (accelerometer, gyroscope, orientation sensors), and voice recognition. Like our study, Buriro *et al.* [26] focused on the continuous authentication of a mobile banking application. The difference is that they only used the login screen.

In this paper, the proposed continuous authentication is opted for mobile banking, hence for generally sensitive data. However, even in a mobile banking application, there can be various non-sensitive transactions. We believe that continuous (passive) authentication in mobile devices through behavioral biometrics can bring brand new authentication systems to present devices. We follow a multimodal approach by analyzing touch gestures and micro-movements for continuous authentication. Moreover, we investigate resource consumption in comparison to the native mobile banking application.

**TABLE 2. Transactions and postures used in the data collection process.**

| **Transactions** |
|---|
| T1: Account and credit card balance control on the dashboard |
| T2: Account search on account list and balance control |
| T3: Money transfer from one account to another |
| T4: Foreign exchange buy operation |
| T5: Credit card debt payment |
| **Postures** |
| P1: Phone in hand and sitting |
| P2: Phone in hand and standing |
| P3: Phone on the table and sitting |

As mentioned in Table 1, we present a comparison with the related studies. In the literature, we observe that practical behavioral biometrics applications, especially for mobile banking, are not thoroughly explored. In related studies, data from different numbers of users (ranging between 10 to 104) is collected. Even our goal was higher; we could reach 45 users because of the Covid-19 pandemic social isolation in 2020. Forty-five users appear to be an average number, considering the works in Table 1. Since the studies were performed on different datasets using different classification algorithms and the results are reported in different metrics, it is difficult to compare the performance. A benchmark is needed to compare the exact performance. However, we observe that we can achieve similar performance or in some cases better performance than the related studies. For example, similar FAR values (0.03) were reported in [8] using the same classifier, namely SVM. Again, similar FAR values (0.96%) were reported in [25] using LSTM. However, these two studies report better results in terms of FRR [8] and EER [25]. In the literature, error values range between 0.1% and 40% and we also achieve similar results both in the validation phase and also in real-time authentication experiments. Most of the related studies focus on collecting the data and on the performance of the biometric models but the impact on resource consumption is not usually investigated, except a few studies [23], [24], [27]. We also investigate the resource usage and observe that DAKOTA does not bring extra overhead in terms of power and memory usage compared to the original banking application.

## III. DATA COLLECTION AND PROCESSING

To collect data from participants, we modified the mobile banking application on the Android platform by extending it with a logger component. We directly modified the mobile banking application version that is used for testing operations in the bank by software testers. The logger collects data from motion sensors, including accelerometer, gyroscope and magnetometer, as well as touch screen data. Details of the raw data are explained in Section III-B. We will share the dataset with the researchers interested in working on it via e-mail.

According to an analysis performed on the usage patterns of the current bank customers, the 5 most frequently used functions in the mobile application are identified. These 5 transactions are used to simulate the customer behavior, presented in Table 2. In $T1$, a user displays balance information about all the accounts by scrolling between the accounts listed on the dashboard and then displays the credit card balance details in the same way. In the test application, users have multiple accounts so that he/she scrolls between the accounts and cards. In $T2$, a user clicks on the menu icon, then clicks on ‘my accounts’, searches for a specific account with a given number, and then checks the balance. In $T3$, again, a user clicks on the menu icon, and then the money transfer icon and performs the transfer action between two accounts. In $T4$, a user selects the foreign exchange from the menu and buys a specific amount of Euros/Dollars. In $T5$, a user clicks on credit cards and pays a specific amount of debt to one of his/her accounts. All the users followed the same patterns in the data collection. However, there were small differences in the accounts or cards selected since the test environment of the banking application was dynamic.

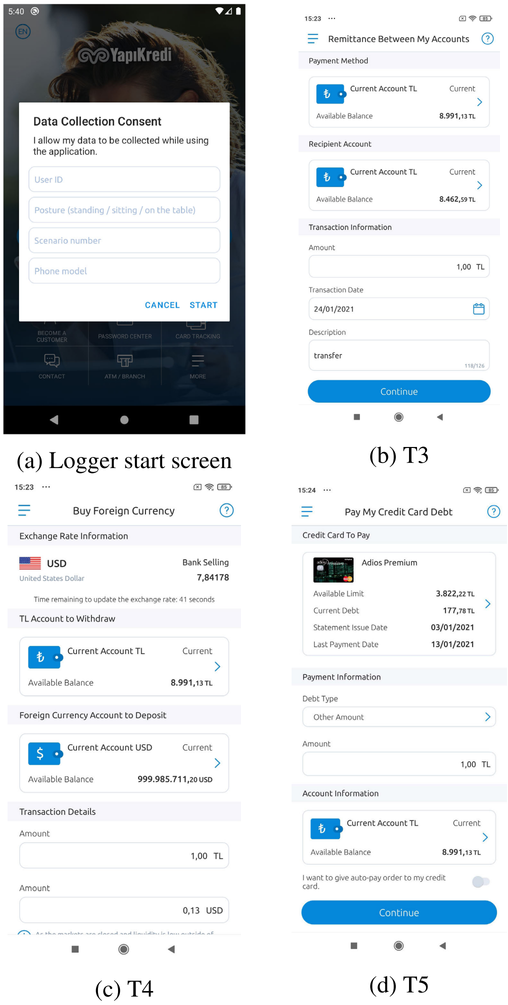
Figure 1a shows the logger’s consent screen: the user enters the ID, position, scenario number, and phone model before starting to use the application. Examples of screenshots for other transactions are also presented in Figure 1. As mentioned, $T1$-$T5$ are identified to be the most common operations on the mobile application and performing the same actions makes it more challenging to classify/authenticate the users compared to collecting the data while freely performing the transactions.

### A. DATA COLLECTION PROCESS

We also considered different usage postures to investigate the effect of the phone or user position on authentication performance. Each user has completed the transactions in each of the postures shown in Table 2: $P1$, $P2$, and $P3$. Although in related studies [8], [27], different postures, such as walking, are also considered, other postures are not very common in a mobile banking application. While the users were holding the phone in hand, they were not instructed to use a specific hand, single hand, or both hands. They used the application in their ordinary usage of a smartphone. For the last posture, $P3$, the phone was kept on the table, and they were asked not to take the phone in hand, but to keep it on the table until finishing the transactions.

For completing each transaction, a user first logged in, performed the operations, and then logged out. Users usually performed each transaction one after another. The data were collected on different days when they did not have time to complete all transactions in different postures for some users. While collecting the data, an assistant/student was reading the operation to be done, bank account number, credit card number, etc., assisting the user.

The data is collected from 45 subjects, mainly consisting of undergraduate/graduate students and bank employees. The age range was between 18 and 42. All participants signed a written informed consent before participating in the data collection. Each subject followed the 5 transaction scenarios in 3 postures (given in Table 2), making 15 sessions per user. Only one user has 13 sessions, who did not complete the data collection for the last two transactions while the phone was on the table. Data collection in each session was around 1.5 minutes. Hence, for each user completing the 5 sessions in one posture took around 7.5 minutes. In total, 22.5 minutes were spent by each user to complete in 3 postures. In terms of the number of samples, raw data is around 50 MB per user collected at 100Hz. The number of scroll gestures per user is around 240 in total, including 3 positions and 5 sessions. Hence, the number of scrolls per user per session was around 16. In total, 1.9 GB of raw data is collected from 45 users.



Data Collection Consent

I allow my data to be collected while using the application.

(a) Logger start screen

Payment Method

Current Account TL Current
Available Balance 8.991,13 TL

Recipient Account

Current Account TL Current
Available Balance 8.462,59 TL

Transaction Information

Amount: 1,00 TL
Transaction Date: 24/01/2021
Description: transfer 118/126

(b) T3

Exchange Rate Information

USD Bank Selling
United States Dollar 7,84178

Time remaining to update the exchange rate: 41 seconds

TL Account to Withdraw

Current Account TL Current
Available Balance 8.991,13 TL

Foreign Currency Account to Deposit

Current Account USD Current
Available Balance 999.985.711,20 USD

Transaction Details

Amount: 1,00 TL
Amount: 0,13 USD

(c) T4

Credit Card To Pay

Adios Premium
Available Limit 3.822,22 TL
Current Debt 177,78 TL
Statement Issue Date 03/01/2021
Last Payment Date 13/01/2021

Payment Information

Debt Type: Other Amount
Amount: 1,00 TL

Account Information

Current Account TL Current
Available Balance 8.991,13 TL

I want to give auto-pay order to my credit card.

(d) T5

**FIGURE 1. Screenshot examples of the transactions.**

Two different phone models are used for data collection: i) Samsung Galaxy S9 ii) Xiaomi Mi8. Both devices have an Android 8 operating system on an octa-core CPU and are equipped with motion sensors (accelerometer, gyroscope, and magnetometer). While the Galaxy S9 has a 5.8-inch display, Xiaomi Mi8 has a 6.2-inch display. The rationale for using two different phone models was to see the authentication performance on different screen sizes. 24 users collected the

**TABLE 3. Attributes of raw data.**

| Accelerometer, Gyroscope, Magnetometer | Touch screen/ Scroll |
|---|---|
| Time | Time |
| X-Axis | Finger Pressure |
| Y-Axis | X Coordinate |
| Z-Axis | Y Coordinate |
| Fragment Name | Finger Size |
|  | X-Axis Velocity |
|  | Y-Axis Velocity |
|  | Direction |
|  | Fragment Name |

data with Xiaomi and 21 users collected the data with the Galaxy model. At the end of the data collection process, the raw data was transferred to a PC for further analysis.

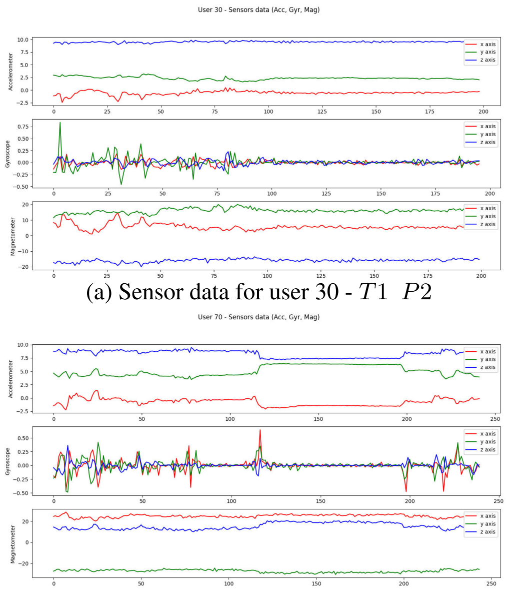
### B. PROPERTIES OF THE RAW DATA

The collected data is of two types: One type is from the three sensors (accelerometer, gyroscope and magnetometer), and the other type is from the touch screen. Table 3 summarizes the properties of the raw data. Both types of data are accompanied by timestamp information. Sensor data includes $x$, $y$, and $z$-axis reading for all sensors, whereas the touch data includes finger pressure and size, $x$ and $y$ coordinates on the screen, as well as x-Axis Velocity and y-Axis Velocity and direction of a scroll. The fragment name mentioned in Table 3 is related to the mobile banking application, and it identifies the screen name while the data is collected.

The logger records the data from the sensors and touch screen in four separate files on the phone. The accelerometer, gyroscope, and magnetometer data were collected continuously at a rate of 100 Hz during each transaction (actually, we started to collect at 5 Hz, but then we considered that a high rate would be better since we can perform under-sampling if required, and changed to 100 Hz. To check this, we down-sampled the data collected at 100 Hz to 5 Hz and recomputed the features, and we did not observe significant differences). Touch screen data was recorded when a gesture is detected on the touch screen, such as a scroll, using Android API. Figure 2 presents a graphical representation of the motion data from three sensors for two different users, completing the same transaction in the same posture. The $x$-axis corresponds to the sample number. Although we collected data from all touch gestures (double tap, long press, fling), we only used the scroll data. Considering the mobile banking application usage, the most common gesture was the scroll and the data collected from other gestures was more limited in size compared to scroll data.

### C. DATA TRANSFORMATION AND FEATURE EXTRACTION

The raw data is formatted as time-series data, and we focus on a classification problem. Since classification algorithms require the data formatted as labeled examples, first, we need to transform the raw data to the labeled instances, where several features describe each instance.



(a) Sensor data for user 30 - $T1\ \ P2$

(b) Sensor data for user 70 - $T1$, $P2$

**FIGURE 2. Graphical plot of the motion sensors' data from two different users, transaction 1 (*T*1), phone in hand and standing (*P*2).**

To create these instances/features, we preprocessed the data to identify the beginning and the end of each scroll event as follows: while collecting the data, we used the built-in class called Gesture Detector in Android SDK. The gesture detector is responsible for detecting gestures like a scroll, fling, long press, show press, single tap down, single tap up, and double-tap. Hence, when a scroll is detected in our logger by the gesture detector, the touch screen data is stored. However, sensor data is continuously stored. To compute features from the sensors only during a gesture/scroll, we used the timestamp information. Therefore, we used the sensor data segments between the start and end time of a scroll for extracting the features.

For each scroll gesture, we extracted 8 basic features from the sensor data: i) *mean*, ii) *standard deviation*, iii) *median*, iv) *minimum*, v) *maximum*, vi) *variance*, vii) *kurtosis*, viii) *skewness*. These 8 features are extracted from all three axes of each motion sensor. We also computed the magnitude value from the three axes, which is the square-root of the sums of squares of the values in each axis. These 8 features are also extracted for the magnitude values. Hence, in total, we had 96 features from the sensor readings. These features are easy to compute and give necessary information about hand movements. Mean, standard deviation, and maximum were also used in a related study [27]. One may argue that computing all these features may be unnecessary since some are relevant to each other; however since we apply dimensionality reduction and feature selection on the data, extracting many features does not cause extra work. Moreover, we extracted the following 30 features from the touch screen readings:

- *Start_X_first* and *Start_Y_first*: The x and Y coordinates of each scroll event's starting point.
- *Current_X_last* and *Current_Y_last*: The $x$ and $y$ coordinates of the ending point for each scroll event.

- *Current_X_maxdev* and *Current_Y_maxdev*: The maximum deviation of $x$ coordinate from the mean among all points for each scroll event.
- *Current_X_dev*20 and *Current_Y_dev*20: The 20th percentile of the deviation of $x$ and $y$ coordinates from the mean among all points for each scroll event.
- *Current_X_dev*50 and *Current_Y_dev*50: The median of the deviation of $x$ and $y$ coordinates from the mean among all points for each scroll event.
- *Current_X_dev*80 and *Current_Y_dev*80: The 80th percentile of the deviation of $x$ and $Y$ coordinates from the mean among all points for each scroll event.
- *V_pairwise*20, *V_pairwise*50, and *V_pairwise*80: The 20th percentile, the median, and 80th percentile of the velocity distribution among all points for each scroll event. The velocity at a point is the displacement ratio over the duration between the previous point and the current point.
- *A_pairwise*20, *A_pairwise*50 and *A_pairwise*80: The 20th percentile, median, and the 80th percentile of the acceleration distribution among all points for each scroll event, respectively. The acceleration at a point is the ratio of the velocity at the point over the duration.
- *V_medianVelocityLastThree*: The median of the velocity of the last three points for each scroll event.
- *A_averageAccFirstFive*: The median of acceleration of the first five points for each scroll event. If there are less than five points in a scroll event, then the median is calculated.
- *PairwiseDisplacement_lengthOfTrajectory*: The magnitude summation of displacement vectors at all points for each scroll event.
- *Current_Pressure_median*: The median of finger pressure among all points for each scroll event.
- *Current_Size_median*: The median of finger size among all points for each scroll event.
- *Distance*: The Euclidean distance between the ending point and the starting point for each scroll event.
- *DirectionOfEndtoEndLine*: The signed angle in radians of the vector formed by the starting point and the ending point for each scroll event.
- *Ratio*: The ratio of the *Distance* over the *Pairwise Displacement_lengthOfTrajectory*.
- *Duration*: The time difference between the occurrence of the ending point and the starting point for each scroll event.
- *AverageVelocity*: The ratio of the length of the trajectory over the *Duration*.
- *MeanResultantLength*: The average complex exponential magnitude of the angle between displacement vector pairs.
- *AverageDirectionEnsemble*: The average angle of displacement vector pairs in radians.

Most of these features were used in related studies [8], [21]. Together with the sensor features, in total, we extracted 126 features for each scroll event.

To improve authentication performance and reduce the number of features, we performed feature transformation with principal component analysis (PCA). One may argue that a vector of 126 features is not considered a high-dimensional vector; however, we wanted to eliminate the redundant features and reduce the data size to be transmitted from the phone to the server. In the current end to-end implementation, we transfer raw data; however, we plan to extract the features on the phone and transfer them to the server for classification to reduce the amount of data to be transferred. After applying PCA, 126 features are transformed into 70 features in binary-classification and 30 to 70 features, varying for each user in the one-class analysis. In the scikit-learn tool [33], the number of components (i.e., reduced number of features) can be given as a parameter. We varied the value of this parameter and used a pipeline to search for the best combination of PCA transformation with SVM using grid-search. Before applying PCA, we also normalized the features using min-max scaling since their ranges were different from each other.

PCA is a feature transformation method that does not consider the target class. On the other hand, feature selection methods rank the input variables in terms of their usefulness to predict the target. Therefore, we also applied feature selection to see each feature's importance in the biometric model of each user. We compare the results with the selected features to the results when PCA is used to transform features. We applied the *sequential backward selection* method. In this method, first, the recognition score is computed for all $n$ features, then each feature is eliminated one by one. The score is computed for all subsets containing $n - 1$ features, and the worst-scoring feature is eliminated at each step. Using this method, 18 features on the average per user are selected. We also explore the impact of feature selection in the experiments and present the results in Section V.

## IV. METHODOLOGY FOR CONTINUOUS AUTHENTICATION

In this section, the experiment methodology is described. First, in Section IV-A, we present the classification algorithms used to build the biometric classification models, and in Section IV-B, we introduce the experiments' details for validating classifiers' performance. Section IV-C describes the experiments for authenticating the users in real-time while using the mobile banking application.

### A. CLASSIFICATION ALGORITHMS

We formulate the authentication process as a classification problem, either the subject is the real user or not, or in other words, the imposter. In literature, both one-class (unary) and/or binary-class classification methods are used for biometric authentication [4], [34]. In one-class classification, the training set contains only the examples of that class. This seems to be practical for authentication since normally, we have only the real user's data on the phone. We can train a model from user data, and imposters' data can be regarded as outliers. On the other hand, in binary classification, we have the training data from both classes. Since we focus on a mobile banking application, a bank can collect data from different users and train models for each user with training data, also including other users. Hence, we considered that binary classification is not the only solution for our case.

We train the biometric models using both one-class and binary-class algorithms to investigate their performances in our experiments. We used six different classification algorithms and also two ensemble learning algorithms:

- Support Vector Machine (SVM): SVM is commonly used in authentication studies [1], [3], [4], and performs well for binary classification problems. We use both binary and one-class SVM in the experiments.
- K-Nearest Neighbor (kNN): The distance between a new data point and the other training data points is computed and the class is assigned according to the majority class of the neighbors.
- Multilayer Perceptron (MLP) is an implementation of an artificial neural network.
- Decision Tree uses a tree-shaped model.
- Random Forest (RF) builds multiple decision trees during the training phase by randomly selecting the attributes.
- Naive Bayes: The probability of belonging to each class is calculated for an instance and the instance is classified according to the highest probability.
- Ensemble Learning: Ensemble learning uses a combination of models, either using a different or the same classification algorithm.

The parameters of these algorithms are explained in Section V. One of the reasons for our decision on using various classifiers was to compare the performance of different algorithms that are commonly used in authentication problems in the related studies, as highlighted in Table 1. kNN is a lazy learner and easy to update the training phase. SVM is designed initially as a binary classifier and is commonly used in authentication problems. Similarly, MLP is also a standard method. The decision tree and the random forest algorithms use entropy in classifying examples, and Naive Bayes is easy to train. Ensemble methods combine different classification algorithms with improving performance. We used the implemented versions of these classifiers on Python's Scikit-learn tool, an open-source library for data mining and analysis [33]. One-class SVM experiments, on the other hand, were performed on the WEKA platform [35].

### B. AUTHENTICATION EXPERIMENTS

In the binary-class approach, we label the data from a particular user as "user" and the data from other users, excluding this user, are labeled as "non-user". However, in this approach, the data becomes imbalanced, with a ratio of approximately 1:44, since we have 45 users in total, but the data size of all users is not the same. In this case, the classification algorithm learns the non-user class very well. On the other hand, the user class is not well classified since there are few examples of the minority class to find the decision boundary. In order to overcome this problem, in imbalanced datasets, mostly under-sampling or over-sampling approaches are followed [36] as a solution. We did not use under-sampling because of the limited data. Instead, we focused on two different over-sampling methods.

As a first approach, we used over-sampling for the "user" class, utilizing the "synthetic minority oversampling (SMOTE)" technique [37]. SMOTE randomly selects an instance from the minority class and finds its $k$ nearest minority-class neighbors. A synthetic instance is then created by randomly choosing one of the $k$ nearest neighbors and connecting the instance and the neighbor instance to form a line segment [36]. We used $k = 2$ in our evaluations. SMOTE is only applied to the training data, and accordingly, the two classes in the training data have approximately the same number of examples. The other approach was the random sampling of non-user data to match with the size of each user data. In this approach, we randomly picked scroll data from other users. In this case, no synthetic data was used, and the number of samples for the user and non-user classes are equal or very close to each other. The exact numbers change per user; however, it is in the range of 195-280 scrolls/instances per class. On the other hand, one-class SVM is an unsupervised learning algorithm that is trained only on the "user" data. Instead of balancing the dataset, we predict the minority user class using outlier detection. Hence, for one-class SVM, we do not use a sampling method.

After sampling the data, a model is trained for each user. Each classifier is evaluated using the 5-fold cross-validation method. We used the K-Folds cross-validator (sklearn.model, selection.KFold), and it provides train/test indices to split data into train/test sets. It splits the dataset into $k$ consecutive folds, without shuffling by default. Similarly, for one-class SVM evaluations, cross-validation is applied to collect the true positive rate (TPR) and false rejection rate (FRR) results, where all the users are tested one by one. However, for collecting the False Acceptance Rate (FAR) results, we test the one-class SVM models which were trained for each user, with the data from other users except the genuine user.

In the experiments, first, we evaluated the performance of different classifiers, including the impact of one-class or binary classification. Each classification algorithm is trained and tested accordingly. Then, we evaluated the performance according to sensor types and posture types, including data from a specific sensor or a posture in the training and testing. As the performance metrics, accuracy (ACC, Equation 1) and TPR (Equation 2) are used as the classical metrics used in the classification studies. However, in authentication studies, false acceptance rate (FAR), false rejection rate (FRR), and equal error rate (EER) should also be reported. The formulas for FAR and FRR are given in Equations 3 and 4. In Equations 1 to 4, TP represents true positives (positive predictions that are really positive, i.e., the user), FN represents the false negatives (negative predictions, i.e., the imposter, that are actually positive, i.e., the user), FP represents the false positives (positive predictions that are actually negative) and TN represents the true negatives (negative predictions that are really negative). EER is a biometric security metric to determine the thresholds for False Acceptance Rate (FAR) and False Rejection Rate (FRR). When FAR and FRR are equal, the common value is referred to as EER. Error rates should be low, whereas TPR and accuracy values should be high for a successful authentication system.

$$ACC = (TP + TN)/(TP + TN + FP + FN) \tag{1}$$

$$TPR = TP/(TP + FN) \tag{2}$$

$$FAR = FP/(FP + TN) \tag{3}$$

$$FRR = FN/(FN + TP) \tag{4}$$

### C. REAL-TIME AUTHENTICATION EXPERIMENTS

After training the biometric models for each user, we implemented an end-to-end authentication pipeline where we performed real-time experiments when the users interacted with the mobile banking application. For this system, we extended the logger-added mobile banking application such that the collected data is transmitted to a server. On the server, we store the authentication model for each user. The data received from the mobile banking application is tested on the server with these models, which are deserialized. The classification result, which reveals whether the user is valid or not, is returned to the mobile application.

The data is transferred at the end of a usage session after specific actions, such as opening a transaction approval screen. The classification is performed for each gesture or each scroll. Each session contains multiple gestures, and we calculate the average of performance metrics considering all gestures in a session. If the value is below a predefined threshold, the system returns an exception for this session and interrupts the use of the mobile banking application. During real-time authentication tests, the users were not instructed to follow the exact test scenarios (the same order, etc.) in Table 2; however, they again performed similar transactions in three postures.

End-to-end transmission of the data is SSL encrypted. Thanks to the certificate pinning method, the server certificate added to the client application prevents man-in-the-middle attacks and data between client-server cannot be monitored. The data was sent in raw form, not zipped. However, if this authentication scheme is started to be used as part of the mobile application, it can be zipped to reduce data usage overhead.

End-to-end authentication tests were performed by five users who originally participated in the data collection. We aimed to perform the tests with more users, but it was impossible to reach other users due to the COVID-19 lockdown. The five users are user 3, 4, 30, 31, and 70 (there are 45 users, but the user ids are not sequential in the dataset, and these are the IDs used in the dataset).

We should note that these experiments were performed months after the initial data collection when the biometric models were trained. In a similar study [21], inter-week authentication was investigated, which means that a model for a user is trained during an enrollment phase, and then the classifier must authenticate the user with the model trained a week ago. It was reported that, as the temporal distance to the training phase increases, the challenge of authentication increases the error rate increases. It was mentioned that a large number of outliers for the inter-week authentication indicates higher error rates for some users. In a related study [8], it was mentioned that "in practice, user's biometric metrics may vary with time." Similarly, in [9], [10], authors investigated the changes in biometric features, particularly using accelerometer data and keystrokes. They also discuss solutions for adapting the models to address these changes. We also observed similar challenges in the real-time recognition tests.

In machine learning, this issue is defined as the "concept drift" or "covariate shift" [38]. There are different solutions to address the concept drift problem, such as periodically updating the model with new data, learning the change with ensemble methods, etc. Finding the best solution to this problem is out of the scope of this paper. However, we utilize the update method by adding new data to the training model. We use both the old data and also part of the new data to update the model.

Besides the performance metrics related to authentication accuracy, we collected other metrics related to system performance, including delay, transferred data size, and resource consumption, such as the battery. System performance tests are performed on Samsung Note 3 (SM-N9000Q) with Android OS version 5.0. We used a different phone model in these tests since we used the PowerTutor [39] application, and it is not supported above Android version 5.0. PowerTutor is a monitoring application that predicts power consumption based on CPU usage, developed by the University of Michigan. For memory usage, we used the Simple System Monitor application.[3] For network usage, we used the TCP/IP monitoring tool on Eclipse IDE. While performing the system tests, we used two different mobile application versions to compare the performance and the original/unmodified version of the mobile banking application..

## V. EVALUATION AND RESULTS

In this section, we present the results of our experiments and evaluations. Specifically, we aim to answer the following research questions:

1) What is the performance of different classifiers on authentication performance in terms of different

[3] https://play.google.com/store/apps/details?id=com.dp.sysmonitor.app

metrics, including FAR, FRR, EER, and accuracy and TPR?

2) What is the impact of using feature transformation with PCA or using feature selection with sequential backward selection on the authentication performance?
3) Does knowing the usage posture and training a model per posture improve the authentication performance?
4) What is the impact of using data from motion sensors or data from touchscreen interaction on the authentication rates?
5) How does the authentication performance change when users are authenticated in real-time with an end-to-end authentication pipeline?
6) What is the impact of performing real-time authentication in terms of resource usage on the phone?

We start by summarizing the experiment set up and discuss the performance of classifiers in Section V-A. Then we focus on the impact of usage in different postures (phone on a table, sitting, and standing) in Section V-B and the impact of using different sensors in Section V-C. All the evaluations are performed on the Scikit Learn machine learning platform [33], except the one-class SVM classifier tests, performed on the Weka Platform [35], version 3.8.4. Finally, in Section V-D, we present the results of online or real-time recognition.

### A. PERFORMANCE OF CLASSIFIERS

In this section, we focus on the first and second questions mentioned at the beginning of Section V. As mentioned, we use six different classification algorithms, two ensemble methods for binary classification, and we use one-class SVM as an alternative method. We present the parameters of each classifier for the repeatability of the experiments.

- SVM:We used Radial Basis Function (RBF) as the kernel type and set the regularization parameter $C$ to 10.
- kNN: We set the number of neighbors to 2.
- MLP: We set the hidden_layers_size parameter, representing the number of hidden layers and neurons in a hidden layer, between (100, 10).
- Decision Tree: We used ‘entropy’ as a metric for impurity.
- Random Forest: We chose the number of estimators, representing the number of trees in the forest, specifically 30 for the random sampling method and 22 for the SMOTE method.
- Naive Bayes: We used Gaussian Naive Bayes (GaussianNB) with default parameters.
- Ensemble (SVM and MLP): We combined the SVM and MLP classifiers mentioned above using the same parameter values and specified the voting parameter as ‘hard.’
- Ensemble (SVM Polynomial kernel and SVM RBF kernel): We combined two different SVM classifiers with RBF and polynomial kernel types. While the $C$ parameter is set to 10 for both, the voting parameter is specified as ‘hard’.

**TABLE 4. Comparison of classifiers’ performance, average values.**

| Sampling | Classifier | FAR | FRR | TPR | ACC | EER |
|---|---|---|---|---|---|---|
| **SMOTE** | SVM | 0.04 | 3.88 | 96.12 | 99.88 | 3.50 |
| | MLP | 0.08 | 3.37 | 96.63 | 99.85 | 3.17 |
| | kNN | 0.30 | 5.85 | 94.15 | 99.58 | 5.39 |
| | Random Forest | 0.14 | 17.83 | 82.17 | 99.49 | 14.79 |
| | Decision Tree | 0.28 | 11.13 | 88.87 | 99.49 | 9.61 |
| | Naive Bayes | 8.63 | 15.73 | 84.27 | 91.23 | 14.91 |
| | MLP-SVM | 0.04 | 5.62 | 94.38 | 99.85 | 5.17 |
| | SVM(RBF-POLY) | 0.07 | 6.67 | 93.33 | 99.80 | 6.04 |
| **Random Sampling** | SVM | 3.37 | 1.66 | 98.34 | 97.51 | 4.03 |
| | MLP | 4.78 | 1.37 | 98.63 | 96.92 | 4.92 |
| | kNN | 7.24 | 2.96 | 97.04 | 94.88 | 7.49 |
| | Random Forest | 4.85 | 9.09 | 90.91 | 92.97 | 9.58 |
| | Decision Tree | 10.34 | 9.42 | 90.58 | 90.11 | 12.24 |
| | Naive Bayes | 19.59 | 15.72 | 84.28 | 82.33 | 20.43 |
| | MLP-SVM | 2.86 | 2.42 | 97.58 | 97.34 | 4.03 |
| | SVM(RBF-POLY) | 2.93 | 2.18 | 97.82 | 97.45 | 3.96 |
| | One-class SVM | 9.65 | 31.58 | 68.42 | 79.38 | 19.26 |

- One-class SVM: 1.0E-7 gamma value has the best performance according to our experiments. Nu parameter performs differently for each user. We examined all different values in the range of 0.01-0.99 for each user and assigned the nu value according to the best performance. Additionally, all the target attributes are normalized by setting the normalized parameter to true.

Except for the mentioned ones, all other parameters remained as default. Values for the parameters were tuned using grid search on the Scikit Learn platform. The results with different classifiers are presented in Table 4 with the SMOTE method, and random sampling used in the training phase to balance the number of instances from the two classes, as explained in Section IV-B, and PCA was applied in the data preprocessing phase. These results are the average results of the models trained individually for 45 users and they are given in percentage format, hence vary between 0 and 100. When we analyze the SMOTE method results, FAR values are noticed to be less than 1%, and accuracy values are above 99% for all classifiers, except Naive Bayes. Naive Bayes performs worse in terms of other metrics as well. TPR values are less than ACC (accuracy) results, which means that the negative class (not user) is predicted with a higher rate. SVM and MLP exhibit the best results in terms of FRR and EER. On the other hand, ensemble methods perform the same and could not exceed the performance of SVM alone.

As mentioned, SMOTE creates synthetic examples in the dataset to balance the number of samples in different classes. As an alternative method, we evaluated the performance of classifiers with a random sampling method (Table 4). With random sampling, SVM and MLP are again the top two classifiers. The number of random sampling instances is around 500 scrolls (half user, half non-user classes). Similar to the SMOTE method results, SVM and MLP are the best performing classifiers: the error rates are less than 5%, and TPR and ACC values are above 96%. Here, the ensemble classifiers perform better in terms of FAR and EER; however, the difference is around 1%. Naive Bayes exhibits the worst performance again. kNN, decision tree, and random forest

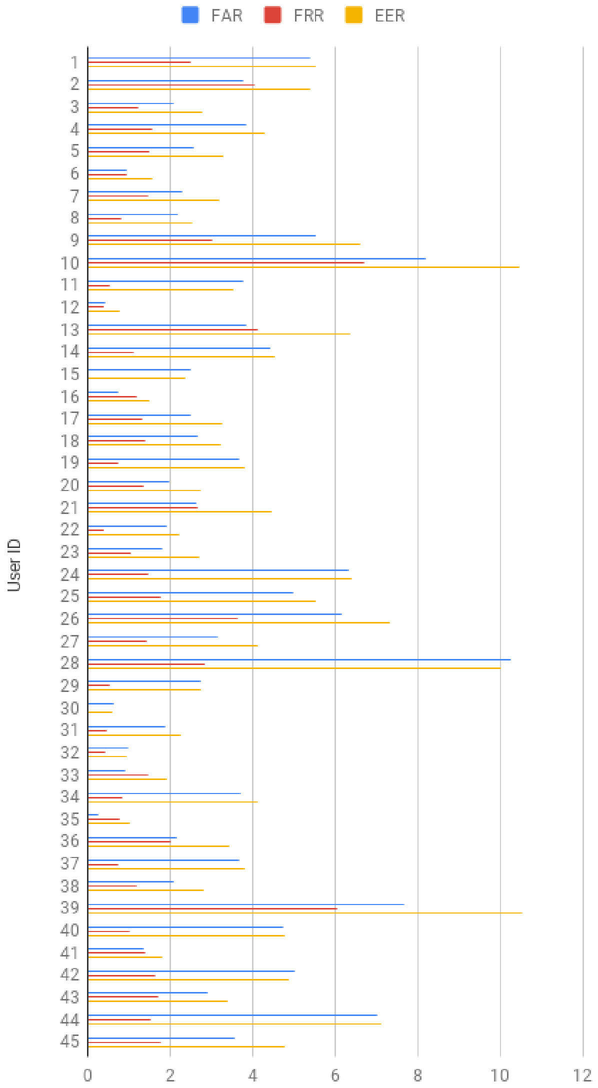


**FIGURE 3.** Error rates (in %) for all users (SVM, Random Sampling).

also exhibit TPR and ACC values above 90%, but they cannot achieve SVM and MLP in terms of FRR and EER. SVM is widely used in related studies, as shown in Table 1. SVM is a binary classifier and hence suitable for an authentication task. In Table 4, we present only the average results, but as an example in Figure 3 ($x$-axis presents the error rates in % and $y$-axis presents the user ID. In the original dataset, user IDs are not sequential from 1 to 45, but here they are presented sequentially for ease of presentation), error rates for SVM with random sampling are presented: the minimum FAR is 0.3% (user 95), while the maximum is 10.2% (user 88), minimum FRR is 0% (user 56 and 90), and the maximum is 6.6% (user 51), minimum EER is 0.5% (user 90) and the maximum is 10.5% (user 99). TPR values vary between 92% and 100%.

Compared to binary classification algorithms, the performance of one-class SVM is lower in terms of TPR and ACC and high in terms of error rates, FAR, FRR, EER. One-class SVM is simply an outlier detection method, which learns the boundaries of the training points that belong to the user's data. The *nu* parameter indicates the ratio of outliers to observe in the data, which varies from user to user. Moreover, to compute FAR, the model is tested with all other users' data; hence there is a 1:44 imbalance in the test. If there are similarities among some users, then the model makes wrong predictions. FRR is also larger, which is due to intra-class variation, we believe.

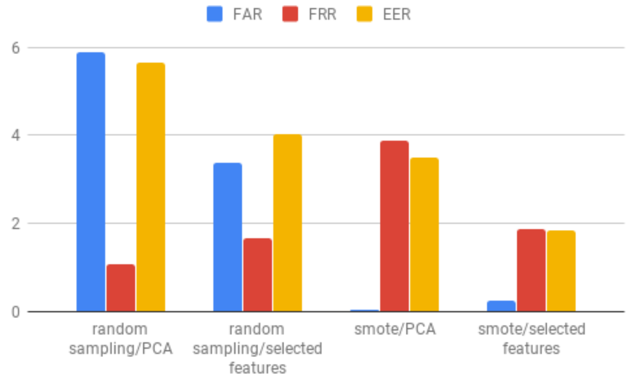
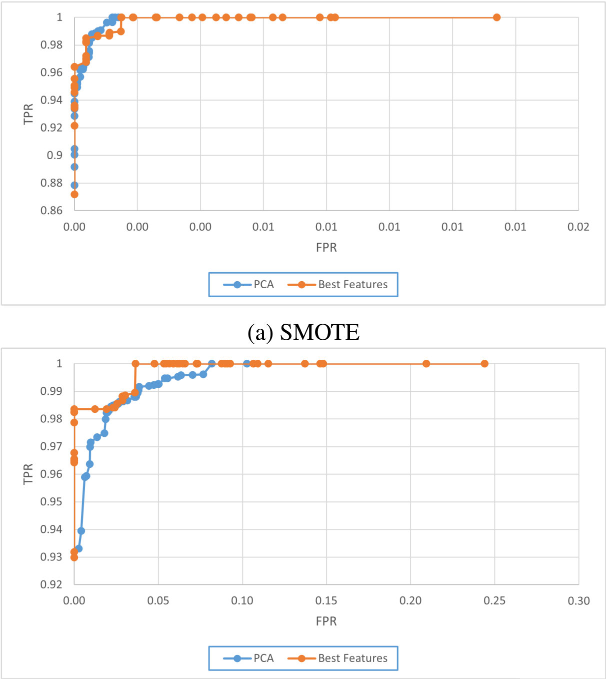


**FIGURE 4.** Comparison of PCA and feature selection in terms of error rates.

As presented in Section III-C, we also applied sequential backward feature selection, and in Figure 4, we introduce the results achieved when the selected features are used in the model training compared to using PCA as a dimensionality reduction method. As mentioned, with feature selection, 18 features on the average per user are selected. On the other hand, after applying PCA, 126 features are transformed into 70 features in binary classification. As we see in the figure, we achieve lower error rates for models trained after feature selection. As mentioned we initially have 126 features, and some are not discriminating features for some users and with feature selection, we could achieve better models. Selected features differ among users; however, features from the gyroscope sensor were the least selected features. Since there is not too much rotation movement while using a mobile application, this is expected. Mean, max, min, and median values from the three axes of accelerometer and magnetometer were among the most selected features. When we look at the touch features, *Current_Size_median* (median finger size), *Current_X_last*, *Current_X_last*, *AverageDirectionEnsemble* and *Direction Of End to End Line* were among the most selected ones. Having a smaller number of features can also reduce the amount of data transferred if the biometric models are kept on a server. In our end-to-end implementation, we transferred raw data and extracted the features on the server, but it is always possible to perform feature extraction on the phone.

ROC curves are very useful in interpreting the results of binary classification problems. On the other hand, in authentication problems, using error rates (FAR, FRR, EER) is more common, as we also observed in the related studies (Section II). To compare with the results reported in similar studies, we preferred using tables and figures that include



(a) SMOTE

(b) Random Sampling

**FIGURE 5.** ROC curves for two sampling methods.

**TABLE 5.** Confusion matrix for user 99, with SVM classifier.

| | Class/Predictions | user | non-user |
|---|---|---|---|
| **SMOTE** | user | **175** | 20 |
| | non-user | 4 | **10387** |
| **Random Sampling** | user | **186** | 9 |
| | non-user | 16 | **160** |

these error rates. However, as an example, we also included ROC curves for comparing the results of applying PCA and feature selection in Figure 5 with the two sampling methods. With SMOTE, similar performance is observed for both PCA and feature selection, whereas with random sampling selected features exhibit a slightly better performance.

Table 5, offers example confusion matrices for a user (user 99) using the SVM classifier, both for random sampling and SMOTE. As mentioned, SMOTE is only applied to training data. That is why we have fewer examples for the user class in the confusion matrix. The lowest scores were achieved for this user with SVM among all users. For random sampling, the results were as follows: FAR 7.6%, FRR 6%, TPR: 93.9%, accuracy 93.5%, and EER 10.5% and for SMOTE, FAR: 0.03%, FRR: 10.8%, TPR: 89.2%, Accuracy: 99.8% and EER 9.4%. Since we have more examples from the non-user class, FRR and accuracy values are better with SMOTE. Hence, the number of testing instances also affects the performance results, shown in related studies [8], [21]. If the bank uses the DAKOTA system, they can easily collect data from customers and expand the dataset.

Coming back to the first question raised at the beginning of Section V, we can summarize that SVM is the best performing classifier both with the SMOTE and random sampling methods in terms of error rates and accuracy values. In the rest of the analysis, we present the results only with the SVM classifier. Furthermore, about the second research question, we observe that lower error rates are achieved for models trained with the selected features compared to models trained after feature transformation with PCA.

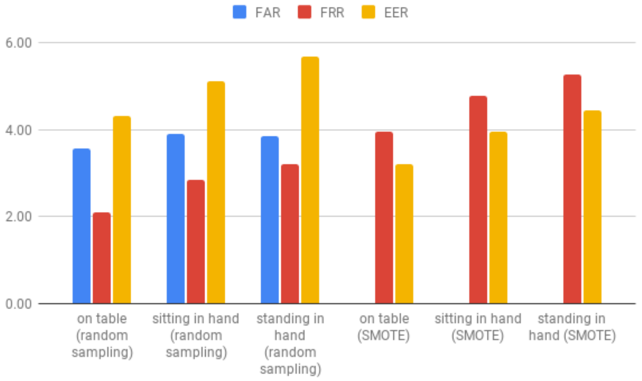


**FIGURE 6.** Impact of posture with SMOTE and random sampling.

### B. IMPACT OF POSTURE

In this section, we focus on the third question mentioned at the beginning of Section V: "Does knowing the usage posture and training a model per posture improve the authentication performance?" Every user completed the scenarios in each of the following stances: sitting and phone in hand, standing and phone in hand, sitting and phone on a table. In order to examine the effect, we trained the SVM classifier with the data only from a specific position. We investigate whether knowing the usage position may improve the authentication performance.

Figure 6 shows the results with the SVM classifier when trained with data from a specific posture. Results are similar for each position. Compared to the SVM results, where data from all positions were used (Table 4 and Figure 4), results are very close. In Table 4, we observe that, with random sampling, using SVM, we achieve: 3.37% FAR, 1.66% FRR, 98.34% TPR, 97.51% accuracy, and 4.03% EER when the data from all postures was trained and tested together. EER increased for sitting in hand and standing in hand with position-specific training. FRR also increased for these positions and slightly increased for the "on table" position. One may argue that when the phone is kept on the table, motion sensors do not produce discriminating data. However, in this case, the touch-features help to identify the users. These results are obtained after feature selection on the data from a specific position. Hence, redundant features were eliminated. In Table 4, we observe that with SMOTE, we achieve: 0.04% FAR, 3.88% FRR, 96.12% TPR, 99.88% accuracy and 3.50% EER. Again, EER slightly increased for sitting in hand and standing in hand when data from these positions are used in training and testing, as shown in Figure 6. In the figure, FAR values for SMOTE appear as if they were not presented, but

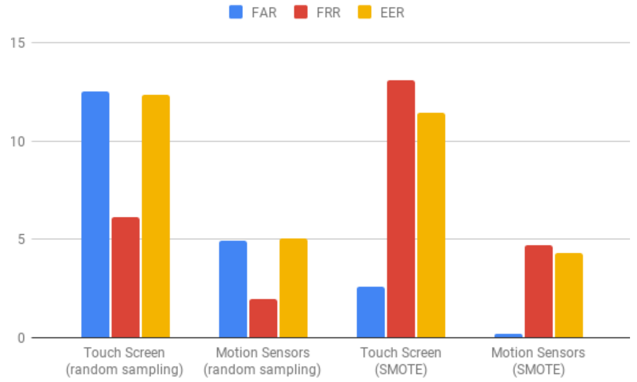


**FIGURE 7.** Impact of sensor features with SMOTE and random sampling.

this is because they have minimal values, ranging between 0.01 and 0.02 for three positions.

This slight increase in error rates may be due to the decrease in the number of testing instances. As mentioned, we were interested in whether knowing the usage position may improve the authentication performance, however as we observe from the results, the error rates, TPR, and accuracy values are similar when data from all positions are combined or used separately.

### C. IMPACT OF SENSORS

In this section, we focus on the fourth question mentioned at the beginning of Section V: ''What is the impact of using data from motion sensors or data from touchscreen interaction on the authentication rates?'' We analyze and compare the impact of using only features from motion sensors and touch screen features. Again, we use binary SVM in the analysis.

Figure 7 presents the results with the SMOTE and random sampling methods. Using only features from motion sensors exhibits better performance than touch screen features in terms of all metrics. One of the reasons is that the number of features extracted from motion sensors is much higher than the touch screen features. Moreover, when we analyze the feature selection results, we see that motion features are better discriminators, particularly mean, max, min and median values from accelerometer and magnetometer were different for most of the users, and these were among the most selected features. Here, in the analysis we used all the features; however, features from the gyroscope sensor were among the least selected features. Compared with the results presented in Table 4, where all features were used together, we observe worse performance: EER was 4.03% and here it is more than 5% with random sampling. Therefore, using both features from the sensors (accelerometer and magnetometer) and those from the touch screen exhibit better performance. It may be argued that the touch screen is kept on while using a mobile application, but sampling the sensors at the same time brings an extra overhead in terms of resource consumption. However, with the touch screen features, EER is higher, which is above 10%. Moreover, since a banking application is not continuously used, sampling the sensors simultaneously does not significantly affect the battery life and user experience, as we discuss in Section V-D. Coming back to the research question mentioned at the beginning of this section, we can summarize that using only features from motion sensors exhibit slightly better performance compared to touch screen features, however using both sensors (accelerometer and magnetometer) and touch screen exhibit much better performance.

**TABLE 6.** Real-time authentication results with old and updated models.

| | OLD MODEL | | | UPDATED MODEL | | |
|---|---|---|---|---|---|---|
| | FAR | FRR | TPR | FAR | FRR | TPR |
| **User 3** | 8.1 | 32.7 | 67.3 | 12.0 | 7.8 | 92.2 |
| **User 4** | 12.05 | 44.0 | 56.0 | 9.6 | 36.1 | 63.9 |
| **User 30** | 16.2 | 58.9 | 41.1 | 14.8 | 20.8 | 79.2 |
| **User 31** | 23.5 | 59.9 | 40.1 | 7.8 | 13.2 | 86.8 |
| **User 70** | 15.3 | 19.8 | 80.2 | 0 | 1.8 | 98.2 |
| **Average** | 15.0 | 43.1 | 56.9 | 8.8 | 15.9 | 84.1 |

### D. REAL-TIME RECOGNITION

#### 1) AUTHENTICATION RESULTS

This section focuses on end-to-end authentication and gives the results of the real-time recognition experiments that were explained in Section IV-C. We investigate on the fifth research question mentioned at the beginning of Section V: ''How does the performance change when users are authenticated in real-time with an end-to-end authentication pipeline.''

The biometric models for the test users were trained with the data collected with the scenarios in Table 2. For testing, the participants used the mobile banking application and the collected data is transferred to a server and are classified with these trained models in real-time. Users did not follow the same transactions but performed similar operations in three postures using the mobile application. In these experiments, we only used SVM models trained with random sampling per user.

Table 6 presents the results in terms of FRR, TPR, and FAR. The ''old model'' column shows the results of the models trained with the data of that user, given in Section V-A. The results are presented per user. To compute FAR for a user, we used the other users' data as the test data. Error rates and TPR values vary among the users. FAR value for user 70 is 0%; however, for other users, the FAR value varies between 8 and 23%, and FRR values are very high, up to 59%. Compared to the results presented in Table 4 and Figure 4, the error rates are also much higher. One of the reasons for this difference and high error rates is that the training data was collected months ago. In the related studies [8]–[10], [21], it was mentioned that changes in biometric features may reduce the predictive performance of the biometric systems. As mentioned in Section IV-B, we decided on updating the model with new data to address this challenge. For each user, we added part of the new data and re-trained the models together with the old data. The number of scrolls in the test data was around 100 examples for each user.

The results of the updated models are also shown in Table 6 under the "updated model" column. Error rates, both FRR and FAR, decrease for all users and TPR values increase when the model is updated with the new data. We tested with a different number of new scrolls to be added: 15, 30, and 45 scrolls. For users 3 and 70 adding 30 scrolls and for other users, 45 scrolls achieved the best scores. Moreover, we achieved better scores by adding new scroll data from each posture instead of randomly adding new data.

The classifiers test each scroll individually and during a session. Some of them might be misclassified and reporting the result of each test to the user is not useful. Instead, once a user finishes a transaction, we are interested in capturing whether the user is the valid user by looking at the outcome of multiple scroll events. To address this, we can decide on the authentication outcome by looking at the number of predictions and use the majority vote as the final decision. However, the majority vote may be misleading particularly for a mobile banking application. Instead, we decided to use a threshold value, 25%. If the number of predictions for the other class is below this threshold in a transaction, we label this transaction as belonging to the majority class. For example, if there are 10 scroll events in a transaction, and there are 8 predictions for the user class and 2 predictions for the non-user class, the user is authenticated for this transaction. This threshold can be changed concerning the requirements of the application. Bundling several consecutive strokes was proposed in [21]. However, instead of correction after classification, they combine the data from multiple strokes at an earlier stage. In our approach, we combine the outcome of prediction for multiple scroll events.

Table 7 presents the session-based results after applying the threshold rule and correcting the misclassified scroll events during a transaction. We computed the error and TPR values per session. For example, for user 30, there were 22 test sessions and 120 scrolls in the test sessions. As we see in Table 6 under the Updated Column, the FRR score for this user was 20.8% (25 scrolls were misclassified out of 120). When we analyze the results per session, in two transactions, 2 scrolls were misclassified out of 8 scrolls, and in another transaction 1 scroll was misclassified out of 5 scrolls. When we classify and correct these sessions as "user", then in 19 sessions the classification was correct. Hence, the FRR is computed as 13.6%. On the other hand, for user 31, the session-based correction did not improve the results. FAR increased for users 3 and 4 because there were 2-3 scrolls in some of the transactions and session-based correction did not improve the results and even increased the error rates. Most of the misclassifications were among these two users. As mentioned, this threshold can be changed with respect to the requirements of the application or the threshold can be dynamically updated according to the number of the scrolls in a transaction.

To summarize or answer the research question mentioned at the beginning of this section, we can say that we could not directly use the models trained before but we need to update the models with new data. Also, it is better to use session-based authentication instead of testing each scroll individually.

**TABLE 7.** Results with session-based authentication.

| | FAR (session-based) | FRR (session-based) | TPR (session-based) |
|---|---|---|---|
| **User 3** | 19.3 | 0 | 100 |
| **User 4** | 13.1 | 20 | 80 |
| **User 30** | 14.8 | 13.6 | 86.4 |
| **User 31** | 7.8 | 13.2 | 86.8 |
| **User 70** | 0 | 0 | 100 |
| **Average** | 11.0 | 9.36 | 90.64 |

### 2) RESOURCE CONSUMPTION RESULTS

In addition to the authentication performance, we also present the power consumption, memory usage, and network usage results of the end-to-end Dakota system compared to the original banking application. We answer the last research question: "What is the impact of performing real-time authentication in terms of resource usage on the phone?"

We present the results of four different versions of the application for comparison in Table 8. The presented results are the average of five tests. We see that the logger and backend integration increase power consumption by 27% compared to the original application. Since the usage periods in the mobile banking application are short (an average session takes 150 seconds), we think that the effect of these on battery life will be insignificant. As mentioned, we used the PowerTutor [39] application for collecting the power usage results. One may argue that PowerTutor is a relatively old application and estimates the resource usage instead of producing exact usage values. We should mention that we aim to compare the resource usage of different versions of the mobile application and all the results are relative. In future work, exact power measurements can be performed, but as we observe in the results, relative to the original application Dakota modifications do not excessively increase the resource usage.

The DAKOTA integration is not significantly different from the original application in terms of memory usage. The average memory is 214.33 MB and 215.63 MB for the original and modified versions, respectively. The average network usage is 362 bytes/s for the original banking application and 3705 bytes/s with the DAKOTA modifications. Since the increase for an average session duration is 0.3 MB (each session was around 1.5 minutes), there will be no significant overhead for users. Hence as a response to the last research question (What is the impact of performing real-time authentication in terms of resource usage on the phone?), we can say that in terms of resource usage, DAKOTA does not bring extra overhead in terms of power and memory usage compared to the original banking application.

## VI. DISCUSSION

- *Comparison with related studies*: As we present in Table 1, the performance in terms of authentication,

**TABLE 8. Resource consumption results.**

| | Power (mJ/sec) | Network usage (bytes/s) | Memory (MB) |
|---|---|---|---|
| Original Banking Application | 118 | 362 | 214.33 |
| Original Banking Application<br>+ Touch screen<br>+ ACC (5 Hz)<br>+ GYR (5 Hz)<br>+ MAG (5 Hz) | 150 | 3705 | 215.63 |

including error rates and TPR values, is in line with the findings of similar studies, particularly those that utilize multimodal authentication. In the literature, error values range between 0.1% and 40% and we also achieve similar results both in the validation phase and the in real-time authentication experiments.

- *Are motion sensors useful for authentication?*: Extracting efficient features is one of the essential steps in the classification phase. Touch screen features were used in related studies [8], [21]. Features from the sensors were eight basic features, and more sophisticated features can also be extracted, such as the ones in the HMOG study [27]. However, these basic features could also capture the phone's basic movements or the basic statistical properties of the sensor data. One may argue that if the user does not move, the phone does not move, sensor values will be useless. This could be the case when the phone is kept on the table. In that case, touch features can be used. When the phone is kept in hand, then basic hand-movements and orientation patterns can be captured with the sensor data [27]. We should mention again that features from the gyroscope sensor were among the least selected features. This was because there is not too much rotation movement while using a mobile application. However, it can be useful for authentication using body movements or for applications when using on the go, such as fitness trackers and step counters.
- *Error rates for a banking application*: A FAR of 10% may be acceptable for authentication on many applications, but not for others, like banking applications. Moreover, strict rules and regulations exist for banking applications and removal of a login-password scheme may not be possible. However, continuous authentication can be used as an additional or complementary solution, similar to OTP, and it provides a continuous control in comparison to one-time login. Our research question was "Can we continuously authenticate the users of a mobile banking application using behavioral biometrics with a certain performance so that we can remove the requirement of OTP in certain transactions with low risk?". Our findings show that, yes we can provide continuous authentication in real-time with 11% FAR and 9% FRR, on average, with session-based authentication and without significantly increasing resource usage in comparison to the original banking application.
- *Changes in behavioral profiles of users:* The biometric models should be able to deal with changes in the interaction patterns of a user. For example, the user's behavioral pattern may change suddenly, e.g., due to an unexpected accident such as a sprained finger [8]. This difference may be hard to recognize in the authentication process. In such a case, other authentication mechanisms should be used. In case of long-term changes in biometric features, we need mechanisms to regularly update the biometric models [7], as we also show in Section V-D.
- *Limitations and extensions*: In the data collection phase, we used two different phone models, which are relatively new. However, the experiments should be repeated with different models. We only used scroll data in the analysis, but users also perform other gestures, such as tap, fling. These can also be considered. In the real-time experiments, we could only include 5 users due to Covid-19 lockdown; however, we plan to extend this number. For some users, FAR and FRR values were observed to be high. Smoothing methods, such as majority or quorum voting, can be incorporated to decrease the error rates. We did not perform any experiments where an attacker mimics a real user. However, since the participants performed the same transactions in the same postures, adversary imitation is partially taken into account. The biometric diversity within the negative class (the 'non-user') may impact the results. Here, data from 44 other users are used to train the negative class, but, the banking application currently has more than 6.4 million users and clearly training data can easily be extended once the Dakota system is used as a product.

## VII. CONCLUSION

In this paper, we proposed the DAKOTA continuous authentication scheme designed to work on a mobile application. We aimed to develop a complementary solution to the strict one-time authentication schemes used in banking applications. The DAKOTA system is built directly on a mobile banking application and logs data from the touch screen and the motion sensors, namely accelerometer, gyroscope and magnetometer, to monitor usage/behavioral patterns and build biometric models. In order to train the models, we collected data from 45 users, while performing popular banking transactions in three different postures, while sitting, standing and keeping the phone on table. Seven different classification algorithms, together with two ensemble algorithms, are trained and tested and the results reveal that binary-SVM with RBF kernel is observed to reach the highest true positive recognition rate, 99%, and the lowest error scores, 3.5% equal error rate (EER). We investigated the impact of sensors and built models, only from the sensor data and touch screen data, and showed that although models from sensor data provide a better performance, touch screen data is also critically important for some users. We also investigated whether knowing the usage position may improve the authentication performance; however the error rates and TPR values are similar when data from all positions are combined or used separately. Finally, we tested the DAKOTA system as an end-to-end pipeline to authenticate the users in real-time. In this pipeline, the client application collects and sends touch and sensor data to a backend, where the trained models are stored, and the classification result is returned to the application. We investigated the performance of the pipeline in terms of authentication accuracy and resource usage. In terms of authentication accuracy, we initially obtained true recognition rates, ranging between 40% and 80%; however after updating the model and using a session-based authentication, we could achieve 90% true positive recognition rate, on average. In terms of resource usage, we show that DAKOTA does not bring extra overhead in terms of power and memory usage compared to the original banking application. As future work, we plan to extend the dataset with the bank customers where we can perform the tests without following a scenario and apply deep learning algorithms in the classification phase on a larger dataset.

## REFERENCES

[1] V. M. Patel, R. Chellappa, D. Chandra, and B. Barbello, ''Continuous user authentication on mobile devices: Recent progress and remaining challenges,'' *IEEE Signal Process. Mag.*, vol. 33, no. 4, pp. 49–61, Jul. 2016.

[2] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith, ''Smudge attacks on smartphone touch screens,'' in *Proc. 4th USENIX Conf. Offensive Technol.*, 2010, pp. 1–7.

[3] M. Abuhamad, A. A. Abusnaina, D. Nyang, and D. Mohaisen, ''Sensor-based continuous authentication of smartphones' users using behavioral biometrics: A survey,'' 2020, *arXiv:2001.08578*. [Online]. Available: https://arxiv.org/abs/2001.08578

[4] A. Alzubaidi and J. Kalita, ''Authentication of smartphone users using behavioral biometrics,'' *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 1998–2026, 3rd Quart., 2016.

[5] Y. Liang, S. Samtani, B. Guo, and Z. Yu, ''Behavioral biometrics for continuous authentication in the Internet-of-Things era: An artificial intelligence perspective,'' *IEEE Internet Things J.*, vol. 7, no. 9, pp. 9128–9143, Sep. 2020, doi: 10.1109/JIOT.2020.3004077.

[6] B. Menkus, ''Understanding the use of passwords,'' *Comput. Secur.*, vol. 7, no. 2, pp. 132–136, 1988, doi: 10.1016/0167-4048(88)90325-2.

[7] H. G. Kayacik, M. Just, L. Baillie, D. Aspinall, and N. Micallef, ''Data driven authentication: On the effectiveness of user behaviour modelling with mobile device sensors,'' 2014, *arXiv:1410.7743*. [Online]. Available: https://arxiv.org/abs/1410.7743

[8] L. Lu and Y. Liu, ''Safeguard: User reauthentication on smartphones via behavioral biometrics,'' *IEEE Trans. Comput. Social Syst.*, vol. 2, no. 3, pp. 53–64, Sep. 2015.

[9] P. H. Pisani, A. C. Lorena, and A. C. Carvalho, ''Adaptive algorithms in accelerometer biometrics,'' in *Proc. Brazilian Conf. Intell. Syst.*, Oct. 2014, pp. 336–341.

[10] P. H. Pisani, A. C. Lorena, and A. C. P. L. F. de Carvalho, ''Adaptive biometric systems using ensembles,'' *IEEE Intell. Syst.*, vol. 33, no. 2, pp. 19–28, Mar. 2018.

[11] Y. Barlas, O. E. Basar, Y. Akan, M. Isbilen, G. I. Alptekin, and O. D. Incel, ''DAKOTA: Continuous authentication with behavioral biometrics in a mobile banking application,'' in *Proc. 5th Int. Conf. Comput. Sci. Eng. (UBMK)*, Sep. 2020, pp. 1–6.

[12] A. De Luca, A. Hang, F. Brudy, C. Lindner, and H. Hussmann, ''Touch me once and i know it's you!: Implicit authentication based on touch screen patterns,'' in *Proc. ACM Annu. Conf. Hum. Factors Comput. Syst. CHI*, 2012, pp. 987–996.

[13] N. Zheng, K. Bai, H. Huang, and H. Wang, ''You are how you touch: User verification on smartphones via tapping behaviors,'' in *Proc. IEEE 22nd Int. Conf. Netw. Protocols*, Oct. 2014, pp. 221–232.

[14] X. Zhao, T. Feng, and W. Shi, ''Continuous mobile authentication using a novel graphic touch gesture feature,'' in *Proc. IEEE 6th Int. Conf. Biometrics, Theory, Appl. Syst. (BTAS)*, Sep. 2013, pp. 1–6.

[15] K.-W. Tse and K. Hung, ''Behavioral biometrics scheme with keystroke and swipe dynamics for user authentication on mobile platform,'' in *Proc. IEEE 9th Symp. Comput. Appl. Ind. Electron. (ISCAIE)*, Apr. 2019, pp. 125–130, doi: 10.1109/ISCAIE.2019.8743995.

[16] Y. Yang, B. Guo, Z. Wang, M. Li, Z. Yu, and X. Zhou, ''BehaveSense: Continuous authentication for security-sensitive mobile apps using behavioral biometrics,'' *Ad Hoc Netw.*, vol. 84, pp. 9–18, Mar. 2019, doi: 10.1016/j.adhoc.2018.09.015.

[17] A. Ramadan, H. Hemeda, and A. Sarhan, ''Touch-input based continuous authentication using gesture-level and session-level features,'' in *Proc. 8th IEEE Annu. Inf. Technol., Electron. Mobile Commun. Conf. (IEMCON)*, Oct. 2017, pp. 222–229.

[18] T. Feng, Z. Liu, K.-A. Kwon, W. Shi, B. Carbunar, Y. Jiang, and N. Nguyen, ''Continuous mobile authentication using touchscreen gestures,'' in *Proc. IEEE Conf. Technol. Homeland Secur. (HST)*, Nov. 2012, pp. 451–456.

[19] L. Li, X. Zhao, and G. Xue, ''Unobservable re-authentication for smartphones,'' in *Proc. NDSS*, vol. 56. Citeseer, 2013, pp. 57–59.

[20] H. Xu, Y. Zhou, and M. R. Lyu, ''Towards continuous and passive authentication via touch biometrics: An experimental study on smartphones,'' in *Proc. 10th Symp. Usable Privacy Secur. (SOUPS)*. Menlo Park, CA, USA: USENIX Association, Jul. 2014, pp. 187–198. [Online]. Available: https://www.usenix.org/conference/soups2014/proceedings/presentation/xu

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

[22] A. Buriro, B. Crispo, and M. Conti, ''AnswerAuth: A bimodal behavioral biometric-based user authentication scheme for smartphones,'' *J. Inf. Secur. Appl.*, vol. 44, pp. 89–103, Feb. 2019, doi: 10.1016/j.jisa.2018.11.008.

[23] C. Shen, T. Yu, S. Yuan, Y. Li, and X. Guan, ''Performance analysis of motion-sensor behavior for user authentication on smartphones,'' *Sensors*, vol. 16, no. 3, p. 345, Mar. 2016.

[24] C. Shen, Y. Li, Y. Chen, X. Guan, and R. A. Maxion, ''Performance analysis of multi-motion sensor behavior for active smartphone authentication,'' *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 1, pp. 48–62, Jan. 2018.

[25] M. Abuhamad, T. Abuhmed, D. Mohaisen, and D. Nyang, ''AUToSen: Deep-learning-based implicit continuous authentication using smartphone sensors,'' *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5008–5020, Jun. 2020, doi: 10.1109/JIOT.2020.2975779.

[26] A. Buriro, S. Gupta, and B. Crispo, ''Evaluation of motion-based touch-typing biometrics for online banking,'' in *Proc. Int. Conf. Biometrics Special Interest Group (BIOSIG)*, Sep. 2017, pp. 1–5, doi: 10.23919/BIOSIG.2017.8053504.

[27] Z. Sitova, J. Sedenka, Q. Yang, G. Peng, G. Zhou, P. Gasti, and K. S. Balagani, ''HMOG: New behavioral biometric features for continuous authentication of smartphone users,'' *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 5, pp. 877–892, May 2016.

[28] B. Attaullah, B. Crispo, F. Del Frari, J. Klardie, and K. Wrona, ''ITSME: Multi-modal and unobtrusive behavioural user authentication for smartphones,'' in *Technology and Practice of Passwords* (Lecture Notes in Computer Science), vol. 9551. Cham, Switzerland: Springer, Dec. 2015, pp. 45–61.

[29] C. Holz and M. Knaust, ''Biometric touch sensing: Seamlessly augmenting each touch with continuous authentication,'' in *Proc. 28th Annu. ACM Symp. User Interface Softw. Technol.*, Nov. 2015, pp. 303–312, doi: 10.1145/2807442.2807458.

[30] G. M. Weiss, K. Yoneda, and T. Hayajneh, ''Smartphone and smartwatch-based biometrics using activities of daily living,'' *IEEE Access*, vol. 7, pp. 133190–133202, 2019.

[31] D. Ekiz, Y. S. Can, Y. C. Dardagan, and C. Ersoy, ''Can a smartband be used for continuous implicit authentication in real life,'' *IEEE Access*, vol. 8, pp. 59402–59411, 2020.

[32] H. C. Volaka, G. Alptekin, O. E. Basar, M. Isbilen, and O. D. Incel, ''Towards continuous authentication on mobile phones using deep learning models,'' *Procedia Comput. Sci.*, vol. 155, pp. 177–184, Jan. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S187705091930941X

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, ''Scikit-learn: Machine learning in Python,'' *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[34] R. Kumar, P. P. Kundu, and V. V. Phoha, "Continuous authentication using one-class classifiers and their fusion," in *Proc. IEEE 4th Int. Conf. Identity, Secur., Behav. Anal. (ISBA)*, Jan. 2018, pp. 1–8.

[35] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *The WEKA Workbench. Online Appendix for Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2016.

[36] H. He and Y. Ma, *Imbalanced Learning: Foundations, Algorithms, and Applications*, 1st ed. Hoboken, NJ, USA: Wiley, 2013.

[37] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," 2011, *arXiv:1106.1813*. [Online]. Available: http://arxiv.org/abs/1106.1813

[38] J. Gama, I. Žliobaitundefined, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 1–37, Apr. 2014, doi: 10.1145/2523813.

[39] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proc. 8th IEEE/ACM/IFIP Int. Conf. Hardw./Softw. Codesign Syst. Synth. CODES/ISSS*, 2010, pp. 105–114, doi: 10.1145/1878961.1878982.

**ÖZLEM DURMAZ INCEL** received the B.S. and M.S. degrees in computer engineering from Yeditepe University, Turkey, in 2002 and 2005, respectively, and the Ph.D. degree in computer science from the University of Twente, The Netherlands, in 2009. She was a Visiting Student with the Autonomous Networks Research Group, University of Southern California, as a part of her Ph.D. studies, from 2007 to 2008. She was a Post-Doctoral Researcher with Boğaziçi University from 2009 to 2012. She is currently an Associate Professor with the Computer Engineering Department, Galatasaray University. Her research interests are in the design and analysis of algorithms for sensing systems, particularly on wearable devices and smartphones, to support human activity recognition, context recognition, and biometrics. She has authored more than 90 articles in refereed journals and international conference proceedings, and has been serving as a reviewer and program committee member for numerous journals and conferences in these areas. She is currently the principal investigator of the Dakota project, funded by Tubitak under the university-industry collaboration program (1505).

**SEÇIL GÜNAY** was born in İstanbul, Turkey. She received the B.S. degree in computer engineering from Galatasaray University in 2020. She worked on behavioral biometrics for mobile banking applications for her engineering project under the supervision of Dr. Incel. She is interested in the application of machine learning and deep learning algorithms for practical systems.

**YASEMIN AKAN** was born in Denizli, Turkey. She received the B.S. degree in computer engineering from Galatasaray University (GSU), in 2019. She is currently pursuing the M.S. degree with the Department of Computer Engineering, Boğaziçi University (BOUN). She worked as a Research Scholar in the Dakota project. She is also working as an Assistant Software Engineer with the Insurance Application Development Team, Yapı Kredi Teknoloji A.Ş., Istanbul, Turkey.

**YUNUS BARLAS** was born in Kırşehir, Turkey. He received the B.S. degree in computer engineering from Hacettepe University, in 2014. He is working as a Principal Software Engineer with the Internet Banking Development Team, Yapı Kredi Teknoloji A.Ş., Istanbul, Turkey. He is interested in data science and particularly applying machine learning techniques for fraud detection. He worked as a Research Scholar with the Dakota project.

**OKAN ENGIN BASAR** was born in Ankara, Turkey. He received the B.S. and M.S. degrees in computer engineering from Middle East Technical University (METU), in 2012, and Galatasaray University (GSU), in 2019. He is currently pursuing the Ph.D. degree with the Department of Computer Engineering, Istanbul Technical University (ITU). He is also working as a Senior Software Engineer in Mobile Applications Development Team, Yapı Kredi Teknoloji A.Ş., Istanbul, Turkey. His research interests include security and privacy in mobile devices, mobile payment systems, continuous authentication, and fraud detection and prevention systems. He worked as a Research Scholar with the Dakota project.

**GÜLFEM ISIKLAR ALPTEKIN** received the B.Sc. degree in computer science in 2001, the M.Sc. degree in industrial engineering from Galatasaray University, in 2003, and the Ph.D. degree in computer engineering from Boğaziçi University, in 2010. She is currently an Associate Professor with the Computer Engineering Department, Galatasaray University. Her research interests contain pricing and optimization models in computer networks, software process modelling, energy efficiency of mobile software, and decision support systems in real life applications.

**MUSTAFA ISBILEN** received the B.Sc. degree in electrical engineering from Istanbul Technical University (ITU), Turkey, in 1990, and the M.Sc. degree in computer engineering from ITU, in 1994. He worked as a Software Engineer at TurkPetrol Holding, Nortel, Koçbank and Yapı Kredi Bank, from 1994 to 2009. Since 2009, he has been working at Yapı Kredi Technology as the Project Manager and Research and Development Coordinator. He is also a Patent and Trademark Attorney. He teaches courses on intellectual property and project management as a Visiting Lecturer at Istanbul University, Turkey. He likes to participate actively in experimentation and modeling studies in Research and Development projects.