iProfile: Collecting and Analyzing Keystroke Dynamics from Android Users

Haytham Elmiligi[1][a] and Sherif Saad[2][b]
[1]*Computing Science Department, Thompson Rivers University, BC, Canada*
[2]*Department of Computer Science, University of Windsor, ON, Canada*

Keywords: iProfile, Keystroke Dynamics, Machine Learning, Android Application, Behavioral Biometric, Smartphone, Data Collection, Feature Extraction.

Abstract: Keystroke dynamics is one of the most popular behavioural biometrics that are currently being used as a second factor of authentication for many web services and applications. One of the reasons that makes it really popular is that it is a resettable biometric, which meets one of the main usability requirements of authentication systems. With the recent advances in mobile technologies, developers and researchers utilized several machine learning algorithms to identify smartphone users based on their keystroke dynamics. The biggest problem that faces researchers in this area is the ability to collect datasets from smartphone users that could be used to train the machine learning algorithms and, hence, create accurate predictive model. This paper introduces iProfile, a native Android application that collects keystroke dynamics from Android smartphone users. This application opens the door for researchers to recruit participants from all over the world to contribute to the data collection of keystroke dynamics. Our iProfile application allows researchers to study the impact of several parameters, such as hardware brands, users' geolocation, native language text direction, and several other factors, on the accuracy of machine learning classifiers. It also helps maintain a standard benchmark for keystroke dynamics. Having a standard benchmark helps researchers better evaluate their work based on consistent data collection procedures and evaluation metrics. This paper explains the main building blocks of the iProfile application, the algorithms used in the implementation, the communication protocol with the database server, the structure and format of the generated dataset and the feature extraction approaches. As a proof of concept, the app was used to develop a novel feature-set that identifies Android users based on 147 features.

1 INTRODUCTION

Keystroke dynamics is currently considered one of the most popular behavioral biometrics that are being considered, as a second factor of authentication, to authenticate users to different services and applications (Lu et al., 2020). There are two different implementations for user authentication using keystroke dynamics: one-time or static authentication and continuous or dynamic authentication. In one-time authentication, a personal profile is created for the users based on their typing patterns and then used only at login time to authenticate those users to the target service or application (Chen et al., 2021). In continuous authentication, a personal profile is crated for the users and then used at scheduled time periods to verify the identity of those users as along as the service is running or the session is active (Baig and Eskeland, 2021). The main two problems that face researchers in this area are 1) the lack of data collection tools for different smartphone operating systems, and 2) the lack of standard benchmarks, which makes it difficult for researchers to compare the accuracy of their predictive models without questioning the impact of the used datasets.

This paper addresses these two problems by introducing iProfile, a native Android application that could be used to collect keystroke dynamics data from Android users. The proposed application allows researchers to study the impact of different attributes on the feature selection and the accuracy of machine learning classifiers. These attributes include the smartphone hardware brand, users' geolocation, and native-language text-direction. Our application also helps in maintaining a standard benchmark for keystroke dynamics on Android platform, which pro-

[a] https://orcid.org/0000-0003-1458-3035
[b] https://orcid.org/0000-0002-5506-5261

Elmiligi, H. and Saad, S.
iProfile: Collecting and Analyzing Keystroke Dynamics from Android Users.
DOI: 10.5220/0011002600003120
In *Proceedings of the 8th International Conference on Information Systems Security and Privacy (ICISSP 2022)*, pages 621-628
ISBN: 978-989-758-553-1; ISSN: 2184-4356

vides a high level of consistency for comparative analysis. This paper explains the design of the iProfile application and how the application utilizes cloud services, the algorithms used in the implementation, the communication protocol with the database server, the structure and format of the generated dataset and the feature extraction approaches.

The proposed iProfile application can be used in both static and continuous authentication. The app can be used in various application domains, such as preventing impersonation in online exams and ensuring the entire session is secure and not affected by any unauthorized accesses.

This paper is organized as follows. Section 2 discusses related work in the literature and explains the difference between our approach and the work published in this research area. Section 3 discusses the details of the application design, including user interface design, administration activities, user identification, connection with could services, etc. Section 4 focuses on the feature extraction approaches used to create an accurate user profile based on raw data. This section compares different approaches and uses machine-learning libraries to conduct comparative analysis between different feature-sets. Section 5 presents a case study to show how researchers could use this application to explore keystroke dynamics authentication on Android smartphones. Finally, we draw our conclusion and discuss future work in Section 6.

2 LITERATURE REVIEW

There are three different research directions in the literature with respect to keystroke dynamics. The first research direction focuses on the implementation of data collection tools and the different techniques used to collect as much information as possible from the users' keystrokes. The second approach discusses the feature extraction and selection methods used to create an accurate user profile. The third approach discusses the utilization of machine learning algorithms to classify users' typing patterns into different classes and identifying users based on their profiles. Other research work also suggested mixing other techniques, such as handwriting, with keystrokes to create a unique profile (Condorelli, 2019).

An example of the first research direction is a software tool for determining of the keystroke dynamics parameters developed by Andrey A. Vyazigin et al. (Vyazigin et al., 2019). The tool used Standard Java libraries JAXB (Java Architecture for XML Binding) and Java NIO.2 for conversion of java objects into aggregated datasets and to implement the file I/O functions, respectively. The advantage of using this design style is that it makes it easy for developers to maintain the project because they have to maintain only one workspace, while creating different interfaces for different operating systems. However, this approach does not capture the real touch screen events on the main screen layout. For example, it does not handle events on the activities in smartphone devices, such as android nor the events on the storyboards in iOS.

Another example is the Android application developed by Antal et al. (Antal et al., 2015). This application was developed using native Android Software Development Kit (SDK) and created a custom layout for the user login window to capture the keystroke dynamics. The application collected the touch events time stamps and created a feature vector that has 71 features. Data collected from 42 participants with different genders and age groups. Although this approach is more accurate than the first example in capturing the touch events using the native SDK, there is a major drawback. Authors decided to remove all user inputs that had errors due to typos or backspaces. Although this approach normalizes the datasets and makes it easy for the classifier to identify similar patterns, it generates a predictive model based on unrealistic assumptions because users make mistakes and making typos or pressing a backspace button should be handled in the pre-processing phase, not totally ignored.

For the second research direction, data extraction methods focused, in most cases, on the timing features. The two main timing features that were considered in almost all research work are the dwell time and the flight time (Stavanja et al., 2020). Dwell time is defined as the time between a touch-down event and the next touch-up event on the same key, whereas flight time is defined as the time between a touch-up event on one key and a touch-down events on the next key. Researchers extended these two concepts to calculate different time combinations, such as the time between two consecutive touch-down events or touch-up events. Researchers also collected other features such keystroke pressure and touch area. An extension to this approach is utilizing smartphone sensors. Corpus et al. collected smartphone accelerometer sensor data in conjunction with timing and swipe dynamics features (Tse and Hung, 2019).

For the third research direction, researchers considered two different types of machine learning classifiers: one-class classifiers and multi-class classifiers. In the context of user authentication using keystroke dynamics, one-class classifiers use binary classification to label an instance (i.e. one sample) as positive

or negative. A positive response means the instance is accepted and hence the user is authenticated, whereas a negative response means the instance is rejected and the user will not be authenticated to the session or the service. An example of one-class classifiers is the one-class Support Vector Machine (SVM) (Schölkopf et al., 1999). The main idea behind one-class SVM is to create origin points in the space that represent one class, for example X, and maximize the distance between all other data points and the space that host the origin points. The main objective is to create a clear gap between the origin data set and all other samples that do not belong to the origin. In this paper context, the X class represents the training set of the users' keystroke dynamics. One-class SVM has been used in the literature for many anomaly detection applications (Hejazi and Singh, 2013), and has been recently been used for authenticating users based on keystroke dynamics (Liu et al., 2014). In addition to one-class SVM, few other one-class algorithms were used to authenticate users using keystroke dynamics. Kazachuk et al. discussed the utilization of fuzzy kernel-based classifier and a modified version of random-forests one-class classifier to achieve an efficient implementation of user authentication systems (Kazachuk et al., 2016).

The second type of machine learning classifiers used for user authentication is the multi-class classifiers (Sahu et al., 2020). Chinmay Sahu et al. developed a novel distance-based localization algorithm that could be used to classify keystroke dynamics data for multi-user biometric classification (Sahu et al., 2020). Abir Mhenni et al. used both long short-term memory (LSTM) model and bidirectional long short-term memory (BLSTM) to classify keystroke dynamics data. LSTM is used to recognize a continuous sequences of keystroke dynamics and BLSTM is used to maintain information about future data. There are several studies in the literature that analyzed the differences between single-class and multi-class classifiers. Pui Koh et al. used the Artificial Bee Colony (ABC) algorithm as a classifier to classify keystroke dynamics datasets (KOH and LAI, 2019). The work published in (KOH and LAI, 2019) provided detailed description on the development and design of a classification system to identify 10 different users based on different sets of features. The authors also explained the implementation of the ABC algorithm to categorize users.

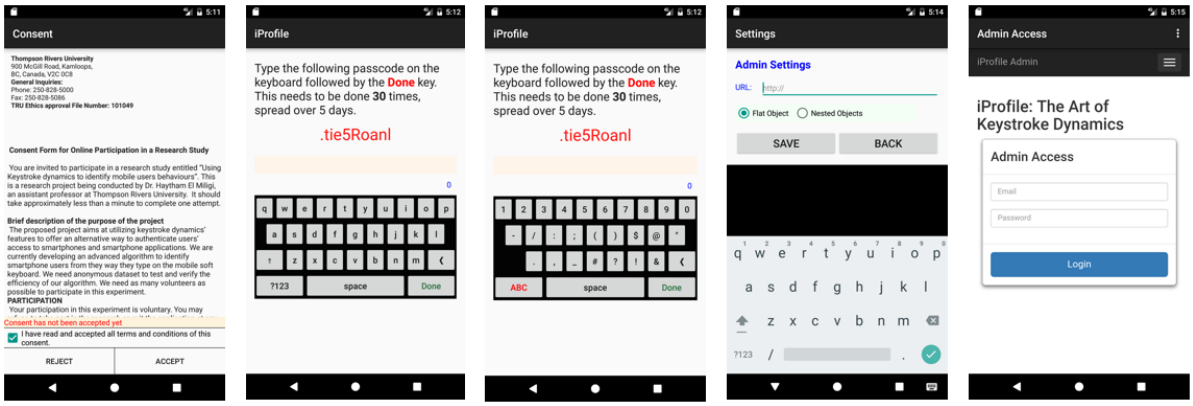
This paper focuses mainly on the first and second research directions. We explain in details the design and implementation of a new mobile application that is currently being used to collect keystroke dynamics data from Android users. With the new releases of Android OS, it becomes important to understand the limitations and challenges that face developers before implementing such an authentication mechanisms into their applications or services. We also discuss the different approaches of providing administrative services to manage the data collection procedure, change the data format, and support different cloud database servers.

3 iProfile DESIGN AND IMPLEMENTATION

3.1 User Interface Design

The iProfile app has several user interface windows. Figure 1 shows five screenshots of the application that show some of its functionalities. Due to ethics requirements, the iProfile app starts with a digital consent, as shown in Figure 1(a). The digital consent is used to: 1) explain the data collection project to the user, 2) emphasize that users' participation in the data collection is voluntarily, 3) illustrate that participants will receive no direct benefits from participating in this research study, 4) assure that data will be saved in an anonymous format with no direct links to the identity of participants, 5) inform the user that the app collects information about the device and the way users type on the keyboard, 6) provides contact information in case the user has questions at any time about the study or the procedures, and finally 7) request the user's consent. Users have to confirm that they have read the consent, they voluntarily agree to participate, and they are 18 years of age or older in order to proceed to the next window. Otherwise, the application will shut down and will not allow them to proceed to the data collection window.

Once the user accepts the digital consent form, the data collection window will appear. Figure 1(b) and Figure 1(c) show screenshots of the data collection window for letters and numeric entries, respectively. In the data collection window, users are asked to type the key word ".tie5Roanl" 30 times, spread over 5 days. They keyboard shows in Figure 1(b) and Figure 1(c) is a custom keyboard that we designed in house and built inside the application. The keyboard design allows users to type in small letters, capital letters, numbers and special characters. There are three standard buttons included to switch between capital and small and to correct typos: a shift, a space and a backspace buttons. There is also an "?123" /"ABC" button to switch between the letter-keyboard interface and the number-keyboard interface. After the users



(a) Consent form. (b) Keyboard – letters. (c) Keyboard – numbers. (d) Administrator settings. (e) Administrator login.

Figure 1: Screenshots from iProfile Android application. Version 2.1.

finishes typing the keyword, they must press on the "Done" key to submit their attempt. Once the "Done" key is pressed, a method inside the Main Activity of the application is called to check for three conditions. The first conditions is whether the user types the correct keyword or not. The second condition is whether there is at least 5 minutes gap between the current attempt and the previous attempt. The third condition is whether the user's device is connected to the WiFi network. If all these conditions are met, the application 1) sends the users' keystroke raw data to our cloud database server along with the timestamp for further processing, 2) displays a toast message to the user to confirm that the attempt was successful and the keyword was entered correctly, and 3) updates a small counter on the main screen to display the number of successful attempts and the timestamp of the last attempt. If one or more of these conditions are not met, the application rejects the attempts and displays an error message to the user to let them know why this attempt was rejected. In all cases, the Text View that holds the data entry will be cleared despite the status of the attempt.

In addition to data collection, our iProfile application provides advanced features for administrators. Through a special menu that can be accessed after receiving instructions from the principle investigator, researchers can set the path to the cloud-based server that hosts the datasets. The network connection module inside the application expects a path to a php script that receives either a flat JSON object for each touch event or a nested JSON array that splits the user's features and the device's features into two JSON objects. It is important to explain here that a touch event refers to any touch-down or touch-up event on any key. That means, for each single character, two events at least will be sent to the database server; one for touch-down and one for touch-up. Each event will be associated with 10 features related to the host Android device and 10 features related to the user's behavior.

One observation should be noted here based on our experimental work. Depending on the touch-screen sensitivity on the target device, the device might send more than one touch event for a single action. That means, a single touch on any key can result in multiple events registered for the same key. The redundant records are filtered out in the data preprocessing phase to make sure that only one single event is registered for touch-up and the same thing for touch-down for each key. Figure 1(d) shows a screenshot of the administrator settings window, where administrators can change the default URL to the cloud database server and the type of the JSON format. The application also provides an access to a web-interface page to access the database server from the mobile application. Figure 1(e) shows a screenshot of the login window from the web interface. This access is implemented in the mobile application to make it easy for administrator to retrieve or check specific records from the data collection application. This access is implemented as a Web View in a separate activity inside the application and can be accessed only through the administrator menu.

3.2 Data Collection Logic

The data collection logic depends mainly on the event handling mechanism. Different alternatives for event handling have been considered before implementing the data collection logic. One important decision design was the design of the keyboard.

The first approach we examined was to use the built-in keyboard in Android and create touch event listeners to listen to touch events on the keys. Although this approach is the optimum implementation, it has limitations on extracting timestamps and pres-

Table 1: Raw data collected on a touch-down event on "t" button.

Features	Value		Feature	Value
ID	48637		Button	t
UID	PMWDS1455200829526		Touch Pressure	0.2832
Language	English		Touch Size	0.419355
HW Model	Nexus 7		X Coordinate	552.6786
SDK Version	23		Y Coordinate	753.4286
Manufacture	Asus		X Precision	1.12
Screen Size	6.77382		Y Precision	1.166667
Time Zone	New York		Action Type	Down
Country Code	US		Action Timestamp	3747694
CPU Cores	4		HR Timestamp	2/11/2021 9:27:43 AM

sure data from the standard Android button views using the on-touch listener class. Based on our tests with Android API 25, that target device information about users' touch-events. Since rooting the target device is not a practical solution, we decided not to consider this option.

The second approach we examined was to create option was to create a keyboard as an Input Method Editor (IME). To implement this option, we have to complete three tasks: 1) creating a class that extends Input Method Service class, 2) create a settings-activity to pass options to the IME service, and 3) create a setting user interface that should be displayed as part of the standard system settings on the android device. Although this option sounds promising since it allows users to change the input method and set our keyboard as the default one for all applications, our surveys show that users did not like to change their defaults password and did not like the extra step on the setting menu.

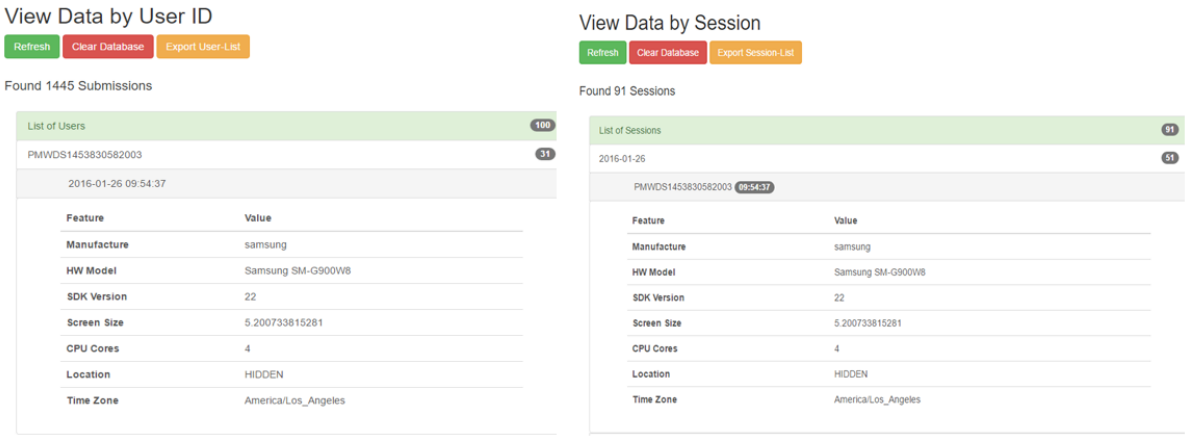
The third approach we examined was to create a custom keyboard layout and design the keyboard logic from scratch to have full control over the touch event listeners. The keyboard could be called as a fragment inside any activity and all data collected from the keyboard fragment can be sent back to the Main Activity when needed. The application collects 20 features: 10 static features to create a profile for the Android device and 10 dynamic features to create a profile for the user.

Table 1 shows an example of data collected at one of our experiments to create a profile for the target device. The ID field is an auto-increment field that identifies a touch event record in the database. The UID is a randomly generated user ID that is associated with each submission from the same user. The other 8 fields are used to identify the device. The right hand-side of Table 1 shows an example of data collected at the same experiment on a touch-down event on "t" button. Every time a touch event is received from any button, a method inside the on-touch listener collects the user's features and stores them into a local database inside the application. The database is implemented as SQLite with a SQLiteOpenHelper class to create a standard interface with the database. Once the user presses the "Done" button and all three conditions discussed before are met, all the records of the keyword is sent to the database server as multiple flat JSON objects or nested JSON arrays based on the settings of the application. The data is filtered out and processed later in the server to extract the timing, position and pressure features. Once the data is sent to the database server, the application waits for a confirmation from the server that it received the JSON objects correctly with no errors. The confirmation is logged in the application for debugging purposes.

The iprofile application provides a new set of features, in addition to the classic features collected in previous work, that describe the target device technical specifications. This new addition allows researchers to study the impact of changing the target Android device on the user's behavioral biometric profile.

3.3 The Server Side

The server side consists of two parts: the front-end and the back-end. The front-end is built using HTML5, java script, bootstrap and php. The front-end allows administrators to run several queries and display data in four different formats: 1) view data by user id, 2) view data by session id, 3) view data in a summary table, and 4) view raw data. Figure 2 shows screenshots of the front-end interface. Figure 2(a) shows a query by user's ID. There are currently 100 users registered in the system but not all of them completed the 30 samples. We are using bootstrap badges, which are numerical indicators of how many items are associated with a link. The administrator can see the number of entries per user as badges in



(a) View data by user ID.

(b) View data by session ID.

Figure 2: Screenshots from iProfile administrator web interface, server side -– front-end.

the right hand side. In this example, the selected user has entered 31 samples and is using Samsung device that is running SDK 22. Figure 2(b) shows a different query by sessions. A session represents 24 hours. We currently have 91 sessions registered in the system. These sessions are spread over more than 6-month period. The administrator can see the number of entries per session. In this example we have 51 entries in the selected session and the timestamp is displayed as a badge next to each user id.

The back-end of the server consists of three modules, MySQL database server, phpMyAdmin as an administration platform for the database server and several php scripts to handle the communication between the front-end and the database server. The database server is password protected and data can be extracted from the server in raw format as a csv file or as an excel sheet. We were planning to collect GPS locations and sensors data and we implemented these features in the first version of the application but we had may participants declining to continue in the data collection phase because they did not feel comfortable with registering their GPS locations. Participants felt that they were tracked although we assured them that data is collected in anonymous format. One of the main reasons for getting these strong concerns from participants is related to the Android handle permissions in the recent releases. Android classifies the access to GPS location in the Android device as a dangerous permission and therefore, developers has to request that permission at runtime, which raises more concerns from the user's perspective compared to the old traditional way of accepting all permissions at installation time. Therefore, we had to remove the part of the code that collects the GPS collection in the latest release of the application (Version 2.1).

4 FEATURE EXTRACTION

When we designed the communication module inside the iProfile application, the highest priority was to transfer as much raw information as possible to the server. We made a critical design decision by no performing any data processing before the transmission stage because we did not want to lose any opportunity to utilize the raw data in the future by creating a new feature or analyzing certain information, etc. While this approach preserved the raw contents, it required a per-processing step to re-format the data before the feature extraction phase.

The data collection was done over 2 weeks with invitation emails sent to over 1000 users and research groups. We also created a home page for the project to answer all questions from participants and referenced the ethics approval obtained from our home university. Throughout the data collection phase, we received 211,304 records for touch events from volunteered from all over the world.

The raw data was sent as a collection of 32 touch events (records) for each successful attempt for each user. Because users had to type the keyword ".tie5Roanl", each successful attempt generated 16 touch-down events and 16 touch-up events, as shown in Table 2.

The "?123" key switches the keyboard from letter to numeric and the "ABC" returns the keyboard from numeric back to letters. The shift key is used to switch the keyboard to upper-case letters, which lasts for the next click only and then returns back to lower-case letters. To prepare the dataset for the feature analysis phase, a python script is developed to execute the following procedure:

Table 2: 32 touch events for each attempt to write .tie5Roanl.

1	?123	Down		17	ABC	Down
2	?123	Up		18	ABC	Up
3	.	Down		19	Shift	Down
4	.	Up		20	Shift	Up
5	t	Down		21	R	Down
6	t	Up		22	r	Up
7	ABC	Down		23	o	Down
8	ABC	Up		24	o	Up
9	i	Down		25	a	Down
10	i	Up		26	a	Up
11	e	Down		27	n	Down
12	e	Up		28	n	Up
13	?123	Down		29	l	Down
14	?123	Up		30	l	Up
15	5	Down		31	Done	Down
16	5	Up		32	Done	Up

1. Data Sorting: All record were originally sorted based on timestamps, which makes it possible to find overlapped instances from different users. The first step in our pre-processing procedure is to sort data based on user id and timestamp in an ascending order.
2. Removing Redundant Records: Our volunteers used different brands of smartphone devices with different sensitivity levels of touch screens. We observed in few records that one touch is generating multiple down and up events for the same key. Our second step was to remove all the redundant records and keep only the last registered event for each key.
3. Excluding Users with typos or Low Number of Samples: We decided to exclude all users with less than 50 correct attempts and all attempts with typos. Scince this paper focuses only on the data collection and initial analysis, we decided to leave the error handling to a future extension of this paper.

5 FEATURE ENGINEERING AND EXPERIMENTAL ANALYSIS

We extracted 147 unique features from each user-attempt. Our generated features are grouped in 12 groups as shown in Table 3. We extracted these 147 features for all users. The app can be used to collect all 147 features and report them back to the research administrators for further processing. As a proof of concept, we created a data-frame using pandas libraries in python for 10 valid users, 10 instances each. We used the generated dataset to train various machine learning models. These models were evaluated based on 10 K-fold cross validation. The experimental results show that Naive Bayes achieved 96% classification accuracy, Decision table achieved 89% classification accuracy, whereas Random Forest achieved 100% classification accuracy. We believe that this application can be used on a wide range within the research community to provide a standard data collection environment. Having such a tool helps improve the quality of comparative analysis between research findings because the same collection tool was used to generate the datasets used in these comparisons. This first version of the app is currently in review on Google Play store (Elmiligi, 2021) and we are planning to give researchers the ability to configure the IP address of the destination server so that they can customize the app for their needs. We are also planning to create an open source project to invite researchers from allover the world to collaborate in the development of this tool so that we can advance the research in this field.

6 CONCLUSION AND FUTURE WORK

This paper presented the fine implementation details of a data collection tool for keystroke dynamics on Android phones. The collection tool, iProfile, helps the research community to use a common environment to create comparable datasets. iProfile can be used as a stand-alone app for data collection or as an Android library that can be imported in any Android application to facilitate the data collection feature. The paper explained how the collected data can be analyzed and processed to create a new dataset of 147 features. Experimental results show that the new feature vector can achieve high accuracy of 100% if used with Random Forest classifier. Although this results were obtained based on samples from only 10 users, it proves that the feature vector created by our tool can be used to standardize the data collection and feature engineering processes. It also generates a feature vector that can be used to train existing machine learning classifiers to achieve high accuracy compared to similar trained models in the literature.

Table 3: 147 features extracted from raw data based on touch events for 18 users, 50 valid samples per user.

N	Feature-Type	Definition	Number of Features
1	Down-up	Dwell time	16
2	Up-down	Flight time	15
3	Down-Down	Flight time	15
4	Up-Up	Flight time	15
5	Down-up (2-graph)	Flight time	15
6	Pressure	Touch pressure	16
7	Size	Touch area	16
8	X-Y P	X-Y precision	16
9	X-Y C	X-Y coordinates	16
10	Averages for timing features	Average UD, DD, DU, UU, 2-graph	5
11	Average pressure	Average pressure for all keys	1
12	Average size	Average area for all keys	1
	Total		147

REFERENCES

Antal, M., Szabó, L. Z., and László, I. (2015). Keystroke dynamics on android platform. *Procedia Technology*, 19:820 – 826. 8th International Conference Interdisciplinarity in Engineering, INTER-ENG 2014, 9-10 October 2014, Tirgu Mures, Romania.

Baig, A. F. and Eskeland, S. (2021). Security, privacy, and usability in continuous authentication: A survey. *Sensors*, 21(17).

Chen, Z., Cai, H., Jiang, L., Zou, W., Zhu, W., and Fei, X. (2021). Keystroke dynamics based user authentication and its application in online examination. In *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 649–654.

Condorelli, M. (2019). Eva lindgren and kirk p. h. sullivan (eds.). observing writing. insights from keystroke logging and handwriting. *Written Language & Literacy*, 22:159–162.

Elmiligi, H. (2021). iprofile – app landing page on googleplay store - https://play.google.com/store/apps/details?id=blue.pyramid.iprofile.

Hejazi, M. and Singh, Y. P. (2013). One-class support vector machine to anomaly detection. *Applied Artificial Intelligence*, 27(5):351–366.

Kazachuk, M., Kovalchuk, A., Mashechkin, I., Orpanen, I., Petrovskiy, M., Popov, I., and Zakliakov, R. (2016). *One-Class Models for Continuous Authentication Based on Keystroke Dynamics*, pages 416–425. Springer International Publishing, Cham.

KOH, P. M. and LAI, W. K. (2019). Keystroke dynamics identification system using abc algorithm. In *2019 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, pages 108–113.

Liu, J., Zhang, B., Zeng, H., Shen, L., Liu, J., and Zhao, J. (2014). The beihang keystroke dynamics systems, databases and baselines. *Neurocomputing*, 144:271 – 281.

Lu, X., Zhang, S., Hui, P., and Lio, P. (2020). Continuous authentication by free-text keystroke based on cnn and rnn. *Computers & Security*, 96:101861.

Sahu, C., Banavar, M., and Schuckers, S. (2020). A novel distance-based algorithm for multi-user classification in keystroke dynamics. In *2020 54th Asilomar Conference on Signals, Systems, and Computers*, pages 63–67.

Schölkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., and Platt, J. (1999). Support vector method for novelty detection. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 582–588, Cambridge, MA, USA. MIT Press.

Stavanja, J., Peer, P., and Emeršič, . (2020). Use of keystroke dynamics and a keystroke-face fusion system in the real world. In *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*, pages 1758–1763.

Tse, K.-W. and Hung, K. (2019). Behavioral biometrics scheme with keystroke and swipe dynamics for user authentication on mobile platform. In *2019 IEEE 9th Symposium on Computer Applications Industrial Electronics (ISCAIE)*, pages 125–130.

Vyazigin, A. A., Tupikina, N. Y., and Sypin, E. V. (2019). Software tool for determining of the keystroke dynamics parameters of personal computer user. In *2019 20th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices (EDM)*, pages 166–171.