International Journal of Emerging Trends in Information & Knowledge Management
https://www.eurekajournals.com/knowledge.html
ISSN: 2581-3579
Special Issue: "Transcending Tech Ed Nexus: Advancing Education through Technological Integration" - 12th June-2024

Keystroke Dynamics Data Collection and Analyzing from Android Users

Dr. Sunitha K M

Computing Science Department, Vijaya College.

Abstract

One of the most widely used behavioural biometrics for second factor authentication in a lot of web services and applications nowadays is keystroke dynamics. Resettable biometrics satisfy a primary usability requirement for authentication systems, which is one of the factors contributing to its immense popularity. Developers and researchers used a variety of machine learning algorithms to identify smartphone users based on their keystroke dynamics, taking advantage of the latest developments in mobile technologies. The ability to gather smartphone user datasets that could be used to train machine learning algorithms and, ultimately, produce accurate predictive models, is the largest challenge facing researchers in this field. In this paper, a native Android application called iProfile is presented. It gathers keystroke dynamics from Android smartphone users. With the help of this application, researchers can now find volunteers to help collect data on keystroke dynamics from anywhere in the world. Researchers can examine how various factors, including hardware brands, users' geolocation, native language text direction, and other factors, affect machine learning classifier accuracy by using our iProfile application. It also contributes to the upkeep of a common benchmark for keystroke dynamics. Having a common benchmark makes it easier for researchers to assess their own work using standardised metrics and data collection techniques. The primary components of the iProfile application, the implementation algorithms, and the database communication protocol are all explained in this paper.

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

Introduction

Based on Lu et al. (2020), one of the most widely used behavioural biometrics for user authentication to various services and applications is keystroke dynamics. This biometric is being evaluated as a secondary authentication technique. One-time or static authentication and continuous or dynamic authentication are the two different implementations for user authentication using keystroke dynamics. During the process of one-time authentication, users' typing patterns are used to create a personal profile, which is only used during login to authenticate them to the target service or application (Chen et al., 2021). A personal profile is created for each user in continuous authentication, and it is used on prearranged intervals to confirm the users' identities as 2021). The two main issues facing researchers in this field are 1) the absence of tools for gathering data for various smartphone operating systems and 2) the lack of standard benchmarks, which makes it challenging for researchers to compare the predictive models' accuracy without considering the influence of the datasets they used.

In order to address these two issues, this paper introduces iProfile, a native Android application that can be used to gather information on Android users' keystroke dynamics. Researchers can examine how various attributes affect feature selection and machine learning classifier accuracy with this suggested application. These characteristics include the brand of smartphone hardware, the location of the user, and text direction in the user's native tongue. Additionally, our program contributes to the upkeep of an industry standard for keystroke dynamics on the Android platform.

A high degree of consistency in terms of comparison analysis. The iProfile application's architecture, its use of cloud services, the algorithms employed in its implementation, the database server's communication protocol, the format and structure of the generated dataset, and the feature extraction techniques are all covered in this paper.

Both static and continuous authentication can be used with the suggested iProfile application. The application has a wide range of potential uses, including guarding against impersonation during online tests and ensuring that no unauthorised access has compromised the session's security.

The structure of this paper is as follows. In Section 2, relevant literature is discussed and the distinctions between our methodology and published research in this field are explained. The specifics of the application design, such as the user interface, administrative tasks, user identification, integration with third-party services, etc., are covered in Section 3. The methods for feature extraction that are employed to produce a precise user profile from unprocessed data are covered in Section 4. In this section, various methodologies are compared, and comparative analysis between various feature-sets is carried out using machine-learning libraries. A case study is provided in Section 5 to demonstrate how this application can be used by researchers to investigate keystroke dynamics authentication on Android smartphones.

Literature Review

Regarding keystroke dynamics, the literature has three distinct research avenues. The first line of inquiry focusses on data collection tool implementation and various strategies to extract as much information as possible from user keystrokes. The techniques for feature extraction and selection that are employed to produce a precise user profile are covered in the second method. The third method talks about using machine learning algorithms to identify users based on their profiles and categorise their typing patterns into various classes. Additional research also recommended combining keystrokes with other methods, like handwriting, to generate a distinctive profile.

A software tool created by Andrey A. Vyazigin et al. (Vyazigin et al., 2019) for determining keystroke dynamics parameters is an example of the first research direction. The tool converted Java objects into ag-objects using the Standard Java libraries JAXB (Java Architecture for XML Binding) and Java NIO.2. aggregated datasets and, in turn, to carry out the file I/O functions. Using this design approach has the benefit of simplifying project maintenance for developers, as they only need to create distinct interfaces for various operating systems while maintaining a single workspace. Nevertheless, this method fails to record the actual touch screen activities on the primary screen arrangement. For instance, it doesn't deal with activities' events.

An additional illustration would be the Android application created by Antal and colleagues (2015). In order to record keystroke dynamics, a unique layout for the user login window was created for this application, which was developed using the native Android Software Development Kit (SDK). The program gathered the time stamps for the touch events and produced a feature vector with 71 features. information gathered from 42 individuals of various ages and genders. This approach has a significant drawback even though it uses the native SDK to capture touch events more accurately than the first example did. The authors made the decision to eliminate any user-submitted text that contained typos or backspaces. This method normalises the datasets and facilitates the pre-processing phase, not totally ignored.

The majority of the time, data extraction techniques for the second research direction concentrated on the timing features. The dwell time and the flight time are the two primary timing features that were taken into consideration in nearly all research projects (Stavanja et al., 2020). Flight time is the interval between a touch-up event on one key and a touch-down event on the next key; dwell time is the interval between a touch-down event and the subsequent touch-up event on the same key. These two ideas were expanded upon by researchers to calculate various time combinations, such as the interval between two successive touch-down or touch-up events. Other characteristics like touch area and keystroke pressure were also gathered by the researchers. An expansion of this is collected smartphone accelerometer sensor data in conjunction with timing and swipe dynamics features.

One-class classifiers and multi-class classifiers are the two types of machine learning classifiers that researchers considered for the third research direction. One-class classifiers use binary classification to label an instance (i.e., one sample) as positive or negative in the context of user authentication using keystroke dynamics. In contrast, a negative response indicates that the instance is rejected and the user will not be authenticated to the session or the service. A positive response indicates that the instance is accepted and the user is therefore authenticated. The one-class Support Vector Machine is an illustration of a one-class classifier (SVM) One-class SVM's primary goal is to construct origin points in space that correspond to a single class. For example X, and maximise the separation between the space containing the origin points and all other data points. Making a distinct separation between the origin data set and all other samples that are not part of the origin is the primary goal. The training set of the users' keystroke dynamics is represented by the X class in the context of this paper. In the literature, one-class SVM has been applied to numerous anomaly detection applications (Hejazi and Singh, 2013). More recently, it has been utilised to authenticate users based on the dynamics of their keystrokes (Liu et al., 2014). A few other one-class algorithms were employed in addition to one-class SVM to identify users based on their keystroke patterns. The use of a modified version of a fuzzy kernel-based classifier was covered by Kazachuk et al.

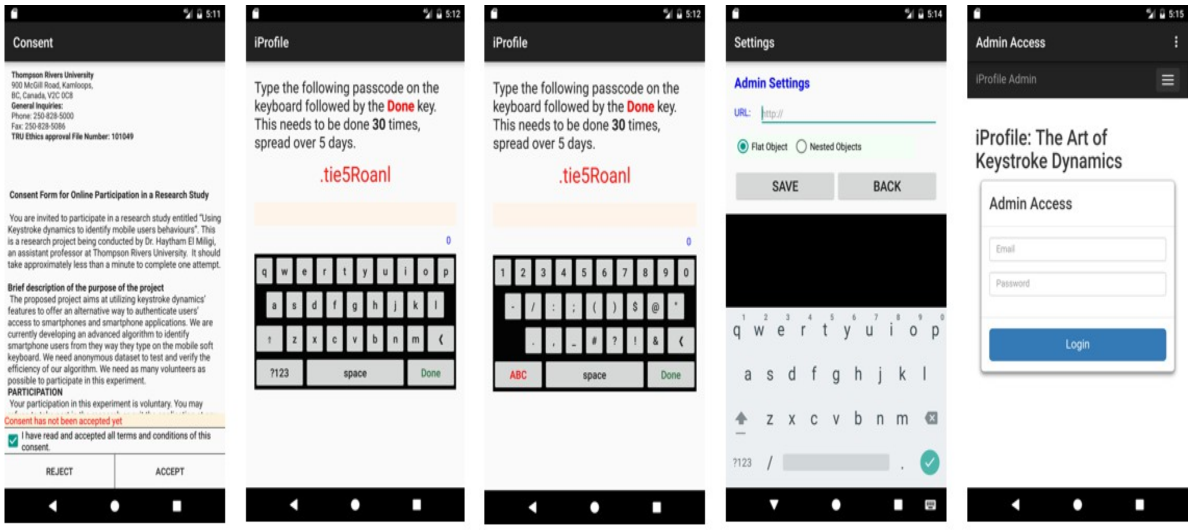
Multi-class classifiers are the second kind of machine learning classifiers used for user authentication (Sahu et al., 2020). In order to classify keystroke dynamics data for multi-user biometric classification, Chinmay Sahu et al. developed a novel distance-based localisation algorithm (Sahu et al., 2020). Keystroke dynamic data was classified by Abir Mhenni et al. using both the bidirectional long short-term memory (BLSTM) and the long short-term memory (LSTM) model. While BLSTM is used to store information about upcoming data, LSTM is used to identify a continuous sequence of keystroke dynamics. The literature contains a number of studies that examined the distinctions between single-class and multi-class classes. The Artificial Bee Colony (ABC) algorithm was utilised by Pui Koh. This paper primarily addresses the first and second lines of inquiry. We provide a detailed explanation of the planning and execution of a novel mobile application that is presently being utilised to gather Android users' keystroke dynamics data. With the most recent releases of Android OS, it becomes crucial to comprehend the constraints and difficulties faced by developers prior to integrating such authentication mechanisms into their services or apps. We also talk about the various strategies for offering administrative services to control the process of gathering data, modify its format, and work with various cloud database servers.

Design and Implementation

User Interface Design

There are various UI windows in the iProfile application. Five screenshots of the application that showcase some of its features are displayed in Figure 1. The iProfile app begins with a digital consent, as seen in Figure 1(a), due to ethical requirements. The purpose of the digital consent is to: 1) inform users about the data collection project; 2) underline that users' participation in the data collection is voluntary; 3) demonstrate that taking part in this research study will not directly benefit participants, 4) guarantee that information will be saved in an anonymous manner without establishing a direct connection to the identities of participants; 5) notify users that the application gathers data regarding their device and keyboarding style; 6) Upon acceptance of the digital consent form by the user, the data collection window will open. Screenshots of the data collection window for letter and numeric entries are shown in Figures 1(b) and Figure 1(c), respectively. Users are requested to type the key word ".tie5Roanl" thirty times over the course of five days in the data collection window. The custom keyboard that we created in-house and integrated into the application is seen in Figures 1(b) and 1(c). Users can type small letters, capital letters, numbers, and special characters thanks to the keyboard's design. To change from capital to small letters and to fix typos, three standard buttons are provided: shift, space, and backspace. Additionally, there's a "?123" / "ABC" button. Once they have completed typing the keyword, they have to hit the "Done" key to submit their attempt. The application's Main Activity calls a method to verify three conditions when the "Done" key is pressed. Whether or not the user types the correct keyword is the first condition. The second requirement is that there must have been a minimum of five minutes between the last attempt and the current one. Whether the user's device is linked to the WiFi network is the third requirement. The application

1) transmits the user's keystroke raw data and the timestamp to our cloud database server for additional processing if all these requirements are satisfied, 2) notifies the user through a toast message. The application rejects attempts if any of these requirements are not met, and it displays an error message to the user explaining the reason for the rejection. Regardless of the outcome of the attempt, the Text View containing the data entry will always be cleared. After the users finishes typing the keyword, they must press on the "Done" key to submit their attempt. Our iProfile application offers administrators advanced features in addition to data collection. Researchers can specify the path to the cloud-based server hosting the datasets through a unique menu that they can access after getting instructions from the principal investigator.



Thompson Rivers University
900 McGill Road, Kamloops,
BC, Canada, V2C 0C8
General Inquiries:
Phone: 250-828-5000
Fax: 250-828-5086
TRU Ethics approval File Number: 101049

Consent Form for Online Participation in a Research Study

You are invited to participate in a research study entitled "Using Keystroke dynamics to identify mobile users behaviours". This is a research project being conducted by Dr. Haytham El Miligi, an assistant professor at Thompson Rivers University. It should take approximately less than a minute to complete one attempt.

Brief description of the purpose of the project
The proposed project aims at utilizing keystroke dynamics' features to offer an alternative way to authenticate users' access to smartphones and smartphone applications. We are currently developing an advanced algorithm to identify smartphone users from they way they type on the mobile soft keyboard. We need anonymous dataset to test and verify the efficiency of our algorithm. We need as many volunteers as possible to participate in this experiment.
PARTICIPATION
Your participation in this experiment is voluntary. You may

Consent has not been accepted yet

✓ I have read and accepted all terms and conditions of this consent.

Type the following passcode on the keyboard followed by the **Done** key. This needs to be done **30** times, spread over 5 days.

.tie5Roanl

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

Figure 1: Screenshots from Android application. Version 2.1.

The application's network connection module expects a path to a PHP script that gets either a nested JSON array that divides the device's features and the user's features into two JSON objects, or a flat JSON object for each touch event. It is crucial to clarify that any touch-down or touch-up event on any key is referred to as a touch event. This implies that the database server will receive at least two events—one for touch-down and one for touch-up—for every single character. Ten characteristics pertaining to the host Android device and ten features pertaining to user behaviour will be linked to each event.

Based on the results of our experiments, one thing should be mentioned here. The target device may send multiple touch events for a single action, depending on the sensitivity of its touch screen. This implies that numerous events may be recorded for a single touch on any key. During the data pre-processing stage, duplicate records are removed to ensure that each key has a single touch-up event and a single touch-down event. An image of the administrator settings window is displayed in Figure 1(d), where administrators can modify the default URL to the cloud database server and the JSON format type. Additionally, the application offers access to a web interface. To make it simple for administrators to retrieve or verify particular records from the data collection application, this access has been implemented in the mobile application. Only the administrator menu will grant access to this Web View, which is implemented as a separate activity within the application.

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 de- sign 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. Al- though this approach is the optimum implementation, it has limitations on extracting timestamps and pres -sure data from the standard Android button views us- ing 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 creation of a keyboard as an Input Method Editor (IME) was the second strategy we looked at. In order to put this option into practice, we must finish three tasks: In order to pass options to the IME service, three things need to be done: 1) create a class that extends the Input Method Service class; 2) create a settings activity; and 3) create a setting user interface that should be visible as part of the android device's default system settings. Our surveys indicate that users disliked having to change their default passwords and the additional step in the setting process, despite the fact that this option seems promising because it lets users change the input method and make our keyboard the default one for all applications.

The third strategy we looked at was designing the keyboard logic and layout from scratch in order to have complete control over the touch event listeners. Any activity could call the keyboard as a fragment, and when necessary, all the data gathered from the keyboard fragment could be sent back to the main activity. Ten static features are collected by the application in order to create a profile for the Android device, and ten dynamic features are collected in order to create a profile for the user.

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

An example of data gathered during one of our experiments to develop a target device profile is displayed in Table 1. An auto-increment field called ID is used in databases to uniquely identify touch event records. Each submission made by the same user is linked to a unique user ID, or

UID, which is generated at random. The device's identity is determined by the remaining 8 fields. An example of data collected during the same experiment on a touch-down event on the "t" button is displayed on the right-hand side of Table 1. A method within the on-touch lander gathers the user's features and saves them into a local database within the application each time a touch event is received from any button. After the data is filtered, it is processed in the server to extract the timing, position, and pressure features. After the data is sent to the database server, the application waits for confirmation from the server that it received the JSON objects correctly and error-free. For debugging purposes, the confirmation is recorded in the application.ted in

The Server Side

The front-end and back-end comprise the server side components. The front-end is constructed with bootstrap, php, java script, and HTML5. Administrators can run multiple queries on the front end and view data in four different formats: view data in a summary table, view data by user ID, view data by session id, and view raw data. Screenshots of the front-end interface are displayed in Figure 2. Figure 2(a) displays a user-ID-based query. Not all of the 100 users who have registered on the system have finished the 30 samples. Bootstrap badges, which are numerical representations of the number of items connected to a link, are what we're using. The administrator can see the number of entries per user as badges inthe side on the right. The chosen user in this example is using a Samsung device running SDK 22 and has entered 31 samples. A different query by sessions is displayed in Figure 2(b). A session lasts for a full day. There are 91 sessions that are registered in the system right now. These meetings take place over a period longer than six months. Each session's total number of entries is visible to the administrator. The timestamp is shown as a badge next to each user ID in this example, which has 51 entries in the chosen session.

Three modules make up the server's back end: a MySQL database server, an administration platform called phpMyAdmin, and numerous PHP scripts that manage communication between the database server and the front end. Data can be extracted from the password-protected database server in raw format as a CSV file or an Excel sheet. We included features for GPS location tracking and sensor data collection in the initial iteration of the application, but several users declined to continue the data collection process because they were uncomfortable with logging the positions they found using GPS. Even though we told participants that the data was collected anonymously, they still felt as though they were being followed. The way Android handles permissions in the most recent releases is one of the key causes of participants' serious concerns. In contrast to the previous conventional method of approving all permissions at installation time, Android considers access to GPS location on an Android device to be a dangerous permission. As a result, developers must request this permission at runtime, which increases user concerns. Consequently, in the most recent version of the application, we had to remove the portion of the code that gathers the GPS data.

Feature Extraction

Transferring as much raw data as possible to the server was our top priority when designing the iProfile application's communication module. We took a crucial design choice in not doing any data processing prior to the transmission stage because we did not want to miss the chance to use the raw data in the future for things like developing new features or doing specific information analysis, etc. This method required a per-processing step to re-format the data prior to the feature extraction phase, even though the raw contents were preserved.

More than a thousand users and research groups received invitation emails during the course of the two-week data collection period. In order to respond to any enquiries from participants and provide references to the ethics approval our home university granted, we also made a home page for the project. We received 211,304 records for touch events from volunteers worldwide during the data collection phase.

For each user's successful attempt, a collection of 32 touch events (records) comprising the raw data was sent. Users had to enter the keyword ".tie5Roanl," so 16 touch-down and 16 touch-up events were produced for every successful attempt.

The keyboard can be changed from letter to numeric mode using the "?123" key and back to letter mode using the "ABC" key. The shift key causes the keyboard to change to uppercase letters for the duration of the subsequent click before returning to lowercase letters. A script written in Python is created to carry out the subsequent steps in order to get the dataset ready for the feature analysis stage:

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:** By initially classifying all records according to timestamps, it is feasible to identify instances of overlap between various users. Sorting data according to user ID and timestamp in ascending order is the first step in our pre-processing process.
2. **Eliminating Duplicate Records:** Our participants utilised various smartphone brands with varying touch screen sensitivity levels. A single touch is generating multiple down and up

events for the same key in a small number of records, as we have observed. The next thing we did was eliminate every duplicate record, leaving only the most recent event that was registered for every key.

3. **Removing Users with typos or Low Sample Count:** We made the decision to remove any users who had fewer than 50 accurate tries as well as any typographical attempts. Since this paper only on the initial analysis and data collection, we have chosen to leave the handling of errors to a future extension of this paper.

Feature Engineering and Experimental Analysis

From every user attempt, 147 distinct features were extracted. Table 3 illustrates sample groups into which our generated features are divided. For every user, we extracted these 147 features. All 147 features can be gathered using the app, and then they can be reported back to the research administrators for additional handling. As evidence of idea, we used pandas libraries in Python to create a data-frame for ten legitimate users, 10 instances apiece. We trained a number of machine learning models using the generated dataset. Ten K-fold cross validation was used to evaluate these models. Naive Bayes achieved 96% classification accuracy, Decision Table achieved 89% classification accuracy, and Random Forest achieved 100% classification accuracy, according to the experimental results. Because the datasets used in these comparisons were generated using the same collection tool, having such a tool enhances the quality of comparative analysis between research findings. The app's initial iteration is presently being reviewed on the Google Play store (Elmiligi, 2021), and we intend to allow researchers to customise the app to their specifications by allowing them to set the destination server's IP address. In order to further the research in this area, we also intend to establish an open source project and invite scholars from all over the world to work together on the development of this tool.

Conclusion and Future Work

The fine-grained implementation details of an Android phone keystroke dynamics data collection tool were presented in this paper. The iProfile collection tool facilitates the use of a standard environment by the research community to produce comparable datasets. To enable the data collection feature, iProfile can be used as a stand-alone app or as an Android library that can be imported into other Android apps. The study described the methods for processing and analysing the gathered data to produce a new dataset with 147 features. According to experimental results, when combined with the Random Forest classifier, the new feature vector can achieve a high accuracy of 100%. Despite the fact that these results were derived from samples.

Table 3: Sample features

Feature-Type	Definition
Down-up	Dwell time

References

Antal, M., Szabo´, L. Z., and La´szlo´, I. (2015). Keystroke dynamics on android platform. *Procedia Technology*, 19:820-826. 8th International Conference Interdis- ciplinarity 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., Orpa- nen, 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 Confer- ence on Signals, Systems, and Computers*, pages 63- 67.

Scho¨lkopf, 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 Emersˇicˇ, (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 dy- namics parameters of personal computer user. In *2019 20th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices (EDM)*.