Journal of Innovations in Computer Science and Engineering
Shahid Beheshti Unversity

May 2023, Volume 1, Number 1

Continuous User Authentication Using a Combination of Operation and Application-related Features

Ahmad Ali Abin[a,1,*,✉] Code Orcid: 0000-0003-0795-8206,
Parisima Hosseini[b,2] Code Orcid: 0009-0003-2045-7569,
Alireza Torabian Raj[c,3] Code Orcid: 0009-0003-5507-7187
[a] *Faculty of Computer Science and Engineering, email: a abin@sbu.ac.ir*
[b] *Faculty of Computer Science and Engineering, email: par.hosseini@mail.sbu.ac.ir*
[c] *Faculty of Computer Science and Engineering, email: a.torabianraj@mail.sbu.ac.ir*

Abstract

Protection of computer systems is a challenge facing the users, who usually define passwords, fingerprints, face detection patterns, and other identification solutions in order to secure their systems against the misuse and unauthorized access. Nevertheless, these solutions are effective in preventing anonymous people from logging in to the system. If a user leaves a system unlocked for a while or a password has already been disclosed for any reason, such trivial solutions will then fail to secure the system. In this study we introduce new dynamic features considering the time, category and type of the applications a user uses and use them in combination with existing operation-related features in a anomaly detection framework for user authentication. A combination of operation-related and application-related features are then taken into account to create a base profile for each authenticated user in order to detect any unauthorized access. The proposed method can secure systems even if an unauthorized access occurs. In other words, this method compares the current user's behavior with the base profile of authenticated user momentarily. If an anomaly is detected, that user is recognized as an unauthorized user and will then be prohibited from working with the system or asked to undergo a two-step authentication process.

***Keywords**:* Anomaly detection, Continues Authentication, Machine learning, User profiling, Insider threat.

[*] Corresponding author
[1] Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran, email: par.hosseini@mail.sbu.ac.ir
[2] Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran, email: a abin@sbu.ac.ir.
[3] Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran, email: a.torabianraj@mail.sbu.ac.ir.
Preprint submitted to Journal of Innovations in Computer Science and Engineering, May 2, 2023

1. Introduction

With the growing use of computer systems in different aspects of life, users always seek to protect the information stored in their computer systems. Defining security policies like passwords for the system and some other files, users try to prevent any unauthorized access to the systems.
In fact, security of computer systems is analyzed in two main areas of net- works and systems. The network-wide algorithms refer to those algorithms that always benefit from network traffic and specific features to detect the unauthorized access. Moreover, the system-wide algorithms are the ones that try to provide security for personal computer (PC) against unauthorized access by defining passwords, fingerprints, facial recognition patterns, and etc. However, such approaches always face challenges. For example, assume that an individual other than the authenticated user of a PC is aware of its password and can therefore log in the PC and use it in the same way as the owner by entering the password. In another case, imagine that the authenticated user enters the password, starts using the PC, and leaves the logged-in PC for a short while. In the meantime, an individual misuses the PC without the authenticated user's knowledge. Therefore, the user information might be at risk. As a result, the user must lock the PC for protection whenever he wants to leave it. If this procedure is repeated for several times, the user might consider it drudgery or might even forget to lock the system.
To rise up to such challenges, if the authenticated user's behavior is modeled, it will then be possible to discriminate the authenticated user's behavior from that of other individuals [1]. Considering both operation-related and application-related features of individuals, this paper tries to distinguish any unauthenticated access to a computer system. Generally, individuals have different behavioral features and habits. These differences also exist in the use of computer systems. It is possible to distinguish the normal patterns of be- haviour among users from anomalies by detecting those differences [2, 3, 4]. In fact, the authenticated user's behavior is considered normal behavior, whereas the behavior of other users is considered an anomaly.
Hence, the main research approach is anomaly detection in the use of a personal computer. In the proposed method, for each authenticated user, a base profile is created through the feature vectors resulting from the user activities. This base profile is used at any moment to detect the unauthorized user. In other words, a current user profile is created from the current system user within

Submit Date: 2022-07-21
Revise Date: 2023-03-27
Accept Date: 2023-05-03
✉ Corresponding author

the period in which the system was used, and compared against the authenticated base profile. If the proposed model recognizes the current user profile as an anomaly, it then starts taking security policies.

In summary the main contribution of this study is introducing new dynamic application-related features and use them in combination with existing operation-related features for user authentication in a anomaly detection frame- work.

The rest of this paper organized in the following way. The literature is reviewed in Section 2 , whereas the proposed method is presented in Section 3. After that, Section 4 reports the experimental results, and Section 5 draws a conclusion.

2. Literature Review

Based on the type of a platform or the type of available features on that platform, the user behavior detection activities can be analyzed. For instance, certain features such as using gyroscopes and touching different areas of a screen are important on smartphones, whereas the type and motion of a mouse and applications are considered on personal computers (PCs). This section reviews the previous works on securing PCs.

Corney et al. (2011) in [5] aimed to detect the unauthorized use of software applications through the internal users of an organization and reduction of false alarms. Their general approach was to create user profiles by registering the records of running applications in security audit logs. Based on the assumption that applications and tasks of individuals are clear in organizations, user profiles included specific pieces of information such as the hours and days when applications were used in addition to different periods of times when they were utilized. As a result, the authorized and unauthorized users were identified through the stored data. In this process, users might change their roles, and tasks might change in organizations. Moreover, the versions of applications may change. To solve the problem of role change or task change, user profiles were created through constant window, growing window, and sliding window approaches.

Ikuesan et al. (2019) in [6] proposed a user behavior detection method by using the mouse-based features. In fact, they performed the authentication through random forest also in [7, 8, 9] tried to detect intrusions and identify unauthenticated users by using the features related to keyboards and mice. They adopted neural networks and classifications technique for intrusion detection.

Pokhrel et al. (2019,2020) in [10, 11] proposed an anomaly-based intrusion detection method to protect the target systems against malicious activities. Their method employed a semi-supervised learning model to identify and learn the event logs and reduce false alarms. They integrated a one-class support vector machine (OCSVM) and a naive Bayes algorithm to develop an anomaly detection model consisting of two steps, i.e., profile creation and anomaly detection. In the profile creation step, the security event logs are considered the user activities, which are actually the processes such as the number of system logins, creation and termination of processes, system locks, wrong password insertion, and periods of system use. The user profile is created by counting the number of security event logs within the period in which the system was used. In the anomaly detection step, the observed behavior of a user is considered the nor- mal behavior. In fact, normal behaviors and anomalies are distinguished by comparing test data with the created profile. Mustafa Akpinar et al. (2021) in [12] used an SVM-based model for anomaly detection in remote working.

Kuen-Jhe Shih et al. (2019) in [13] used keystroke clusters map (KC-map) feature and classified the keystroke periods of users in free texts (selected by users) for authentication. Since the KC-map feature is obtained from clustering, it does not suit conventional classifications. To solve this problem, the authors proposed the keystroke clusters map similarity (KCMS).

Pokharel et al. (2019) in [14] proposed a pattern recognition method in log messages based on time-series techniques for anomaly detection. They used the seasonal ARIMA method to identify deviation between the real and predicted values. In fact, the deviations from a predetermined threshold were considered the anomaly data points. This approach was evaluated through the extreme studentized deviate (ESD) technique.

Emin Anarim et al. (2019) in [15] registered five mouse-based activities such as right and left clicks, double clicks, dragging, and mouse movement to extract certain features such as horizontal and vertical movement velocities. They used semi-supervised learning for anomaly detection.

In [16, 17, 18, 19] user authentication through mouse dynamics was proposed. Shen Fu et al. (2020) in [18] proposed a hybrid CNN–RNN model based on the mouse behavior for user authentication by using the consecutive mouse data as inputs. To perform these operations, the user motion model was first detected. For this purpose, a few targets were shown consecutively on a screen in order to register certain features such the lengths, widths, and timestamps of different events including right clicks and left clicks. The movements were classified as different groups, and the authentication operations were finally performed with respect to the obtained scores.

Some studies used combination of extracted hardware features. So Continuous Authentication System was proposed using the features was extracted from the dynamic mouse and keyboard and using machine learning techniques [20, 21].

As mentioned, the issue of recognizing user behavior in other platforms and organizational systems is also important, so some studies have identified user behavior in organizations and smartphones [22, 3].

Recently, in [23, 24, 25, 26, 4] some features of mice such as movement, angel of movement, click and drag used to distinguish the normal user from others. In other studies, such as [27], Martinez et al. modeled user behavior using clustering techniques in order to personalize digital libraries. Also, in [28], Addae et al. developed effective models for cyber security using behavioral analysis of users. Recently, in [29], Yakura et al. addressed the issue of *Focus Music Recommender* using user behavior analysis.

Existing studies in the topic of automatic user authentication stand more on using operation-related features and mostly on hardware-related ones. Al- though operation-related features can help distinguishing different users, there is another class of features namely application-related features which can help classifying different users precisely. Using the time, category and the type of applications each user uses can significantly improve the accuracy of user authentication.

3. Proposed method

In this study we aim at protecting the target systems against malicious activities. To this end, at the first step, we create base profile for each authenticated user. The base profile is created by extracting some operation- and application-related features within the period in which the system was used by the authenticated user. In the second step, every time the system was used by an individual, also known as current user, his activities are captured and mapped into a current user profile during his system usage. Finally, normal behaviors and anomalies are distinguished by comparing base profile with the current user profile and if the current user profile does not match the base profile, the authentication system locks the system or requests a two-step authentication based on some predefined configuration. Figure 1 demonstrates the general idea behind the proposed method. The proposed method consists of the following steps:

1. Discriminative features extraction
2. Base profile creation for authenticated user
3. Unauthorized access detection
4. Continuous update of the base profile and discriminative model These steps are discussed in more detail in the following sections.

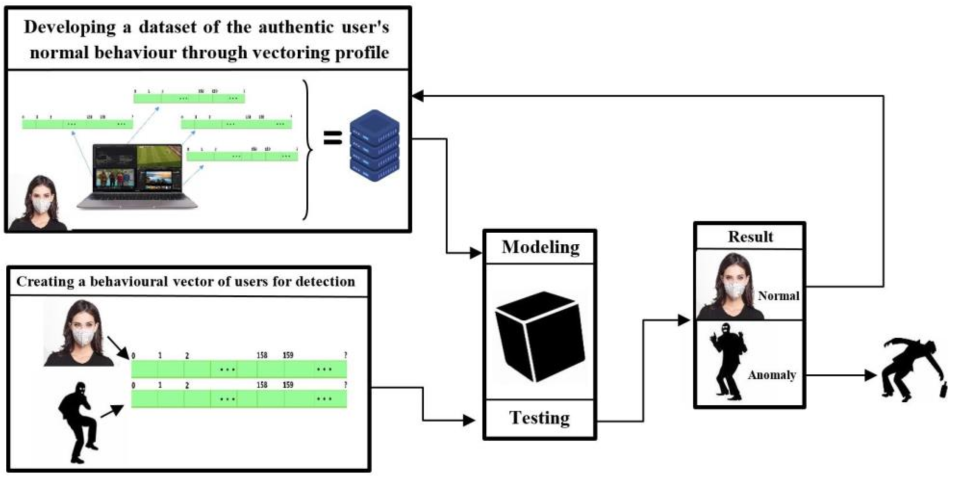

Developing a dataset of the authentic user's normal behaviour through vectoring profile
Creating a behavioural vector of users for detection
Modeling
Testing
Result
Normal
Anomaly

Figure 1: The schematic view of the proposed method

3.1. *Discriminative features extraction*

In this study, an important step is to extract features that can reveal the behavioral differences of different users. Hence, a set of effective features is first generated. The generated features are divided into two operation-related and application-related categories. The operation-related features are related to how the user treats the hardwares (e.g., mouse and keyboard), whereas the user's ways of using applications and tasks were extracted to generate application- related features.

The generated feature will be used in the next steps to distinguish normal behaviors and anomalies. To model the behavior of user within the period in which he uses the system, we capture his activities in every 2 minutes timestamp and create a feature vector consists of a static and a dynamic section for that time stamp. This vector represents the pattern of the users' activities during the time stamp. In the static section, the feature vector has 159 dimensions. However, the size of the feature vector is not fixed in the dynamic section and depends on the applications executed in that timestamp (See Figure 2).

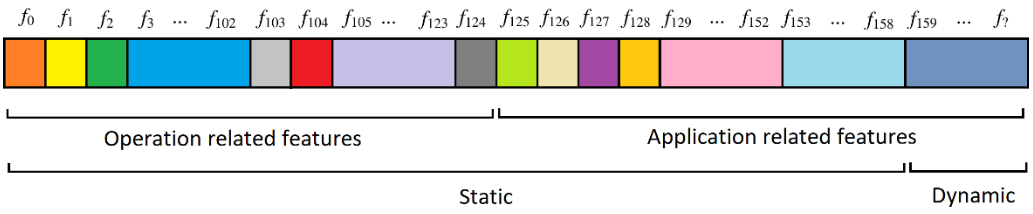

f_0 f_1 f_2 f_3 ... f_{102} f_{103} f_{104} f_{105} ... f_{123} f_{124} f_{125} f_{126} f_{127} f_{128} f_{129} ... f_{152} f_{153} ... f_{158} f_{159} ... $f_?$
Operation related features
Application related features
Static
Dynamic

Figure 2: The feature vector created in every timestamp for system user.

Table 1 indicates the features utilized to develop the feature vector. These features are described briefly in the following sections.
Table 1: Details on the features utilized to develop the feature vector created in every timestamp for system user.

Category	Subcategory	Features	Description	No. of dims
Operation-related	Mouse	$f0$	Mean velocity of mouse	1
		$f1$	Mean of velocity variations	1
		$f2$	Number of movements	1
		$f3, \dots, f102$	Number of clicks per screen grid	100
		$f103, f104$	Duration of holding and releasing right/left click	2
	Keyboard	$f105, \dots, f123$	Count of specific keys on the keyboard	19
		$f124$	Mean of the user's typing speed	1
Application-related	System processes	$f125$	Total number of system logins	1
		$f126$	Total number of system locks	1
		$f127$	Total number of new processes	1
		$f128$	Total number of wrong passwords	1
	Time	$f129, \dots, f152$	The hour of using the system	24
	Applications	$f153, \dots, f158$	No. of running applications in different groups	6
		$f159, \dots, f?$	Running applications	Dynamic

3.1.1. *Operation-related features*

Generally, the first type of features analyzed with respect to how individuals behave toward systems concern their behavior toward the pieces of system hardware in use. The features of mice and keyboards play central roles in detecting the unique behavior of computer users, especially Windows and Linux users. The operation-related features $f_0, \dots, f_{124}$ are described briefly in the following sections.

3.1.1.1. *Mouse-related features*

When a person uses a system that does not belong to him/her, there might be different behavior from that of the authentic user in the mouse movements for the following reasons:

- Difference in the velocity of mouse movement for every individual.
- During the incognito use of a system, users usually intend to finish the job quickly; therefore, they usually move the mouse anxiously. As a result, the mouse is moved faster than the time when it is moved by the authentic user.
- Due to unfamiliarity with the locations of files and system tools, the mouse is moved less slowly than the time when it is moved by the authentic user.

Accordingly, the three extracted features of the mouse movement include: 1) mean velocity of mouse movements, 2) mean of variations in velocity of mouse, and 3) the number of mouse movements, as described in the following.

▶ **Mean velocity of mouse movements**: In every timestamp, the previous position of the mouse is compared with its current position every 0.5 second, and if the two positions are not the same, the velocity of the mouse is determined through Equation (1). After a two-minute timestamp, the mean of the resultant velocities was saved as the final value of feature f_0 in the feature vector. The mean velocity was obtained from Equation (2), whereas x_t and y_t stands for to the vertical and horizontal position of the mouse at time t.

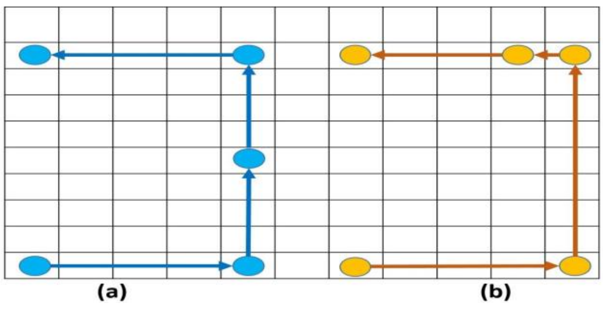
$$velocity_{[t-1,\dots,t]} = \frac{\sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2}}{\big(t-(t-1)\big)} \qquad (1)$$

$$mean\_velocity = \frac{1}{n}\sum_{i=1}^{T} v\,elocity_{[i-1\times 0.5,\dots,i\times 0.5]} \qquad (2)$$

where $T = \frac{timestamp\times 60}{0.5}$.

▪ **Mean of variations in velocity of mouse**: The mean velocity of a mouse might be nearly equal for different people, although they may have different means of variations. Hence, this feature can help distinguish users. Figure 3 indicates the trajectory of a mouse in four steps. Both trajectories are the same, and the mean velocity is equal to 4 in each movement (for both trajectories); however, there are velocity variations. In trajectory (a), the mean of velocity variations is equal to 4.3 pixels; however, it is equal to zero in trajectory (b). The mean of variations in velocity of mouse are calculated, and is stored in the feature vector as f_1.

Figure 3: The mean of velocity variations in two similar trajectories: (a) zero and (b) 4.3 pixels.



- **Number of mouse movements**: To calculate this feature, a counter in- creases by one whenever the position of the cursor changes within the 0.5 second timestamp. After two minutes, the number of movements of the cursor is stored as the counter in the feature vector as f_2.
- **Mouse clicks per screen grid**: The screen is considered a 10 × 10 grid and when the user clicks on each cell in the grid, the counter for the cell increases by one. At the end of the 2-minute timestamp, for each cell the total number of clicks is stored separately. In fact, this feature is determined to find the widely used areas of the user in terms of the number of clicks (See Figure
- **Duration of holding mouse buttons**: The duration of holding and releasing a left or a right mouse button is analyzed every 0.1 seconds, and if the click signal is active, the click counter increases by one. This feature is employed to calculate the total time of different activities such as dragging, clicking, and moving scroll bars on windows in the running timestamp.

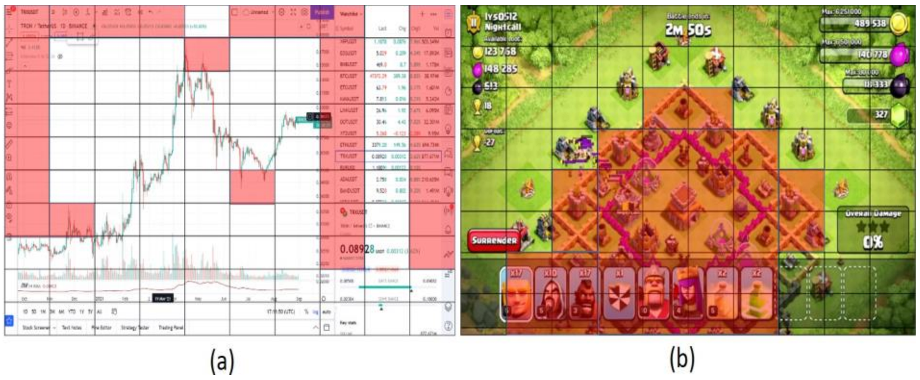


Figure 4: Differences in the areas clicked for different applications: (a) a financial analysis application and (b) a game application. The widely used areas are highlighted in red.

3.1.1.2. Keyboard-related features

The importance of analyzing keyboard-related features is highlighted due to the incognito use of a system, differences in the use of hardware, and other cases. The keyboard-related features include the way of using specific keys and the typing speed. For instance, many people use shortcut keys instead of using mouse.

- **Extent of using specific keys**: Users utilize the keys on keyboards based on their needs and behavioral habits. Some users employ shortcut keys for different purposes such as copying, closing the opened windows, and deleting different items. Given the activities which they perform on their systems, users may use some keys more often. In another case, users differ in the duration that they hold or release specific keys. Accordingly, 19 keys are regarded as specific keys (Left and right ALT, CTRL, and SHIFT keys were considered as the same.). Periodically, if a key is active in every 0.1 seconds timestamp, the corresponding counter increases by one. Finally, the value of every counter is stored separately in the feature vector. Figure 5 shows the considered keys.

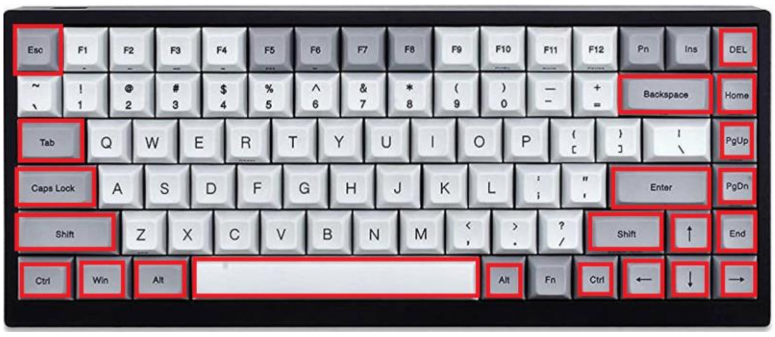


Figure 5: Keyboard specific keys considered in the feature vector.

- **Typing speed per minute**: Another feature is the user's typing speed. In other words, the number of keystrokes is counted in every timestamp, and the mean typing speed per minute is saved as the typing speed in the feature vector. In this case, the keys standing for letters are considered.

3.1.2. Application-related features

In addition to the operation-related features, it has always been important to consider application-related features. From this category, at each timestamp, we capture the: 1) information of system processes, 2) running time of applications,
3) group of running applications, and 4) running applications. In the following, the detail for each category is described.

3.1.2.1. Systemic processes features

To generate features form this category ($f_{125}, ..., f_{128}$ in Figure 2), separate counters are considered to capture the total number of system logins, system locks, wrong passwords and processes creation/completion at each timestamp. In Windows 10 platform, we extract such features from the corresponding Event IDs. Table 2 reports the Event IDs corresponding to the above-mentioned features on Windows 10.

Table 2: Event IDs of some systemic processes used to extract feature.

Feature name	Corresponding Event ID
Total number of system logins (f_{125})	4624 or 4648
Total number of system locks (f_{126})	4740 or 4800
Total number of new processes (f_{127})	4688, 4689 Total number of wrong passwords (f_{128}) 4688, 4625

3.1.2.2. Time-related features

At each timestamp, if we capture the time, it can help us distinguish user from the time when he uses the system. To this end, 24 dimensions of the feature vector (f_{129}, ..., f_{152} in Figure 2) was considered for time-related binary features. In which only one of these 24 dimensions (the cell that show hour at that moment) is one at a moment, whereas the other cells are zero.

3.1.2.3. Running applications group features

It can probably be claimed that a reason for the increasing use of computer systems is the presence of various applications in different fields. A feature that can be considered for discrimination of different users is the group of applications each user uses. For instance, programmers mostly use specific applications like IDE and DBMS of their own fields. More- over, designers and bloggers mostly use their own specific applications. These individuals employ different applications; however, their behavioral focus on applications would be based on their specialties and needs.

To consider applications group, applications are divided into 6 different groups namely general, social networks, programming, office, multimedia, and browsers. Each group contains a list of predetermined applications. If a user uses an application, the counter of that application group increases by one. If we only consider the running programs at the end of a timestamp, the user might execute an application and close it before the end of the timestamp. In an- other case, an application might be executed only at the end of the timestamp; however, it is considered a running program for the entire 2-minute timestamp. Therefore, all of the running programs are checked every 10 seconds to solve such problems, and the group counter changes accordingly. Applications grouping features (f_{153}, ..., f_{158} in Figure 2) are important because it can give us important information about the user in combination of time-related feature. Figure 6 demonstrates the dependency of application to execution times.

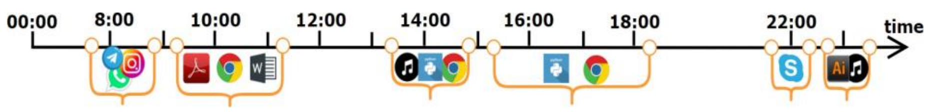
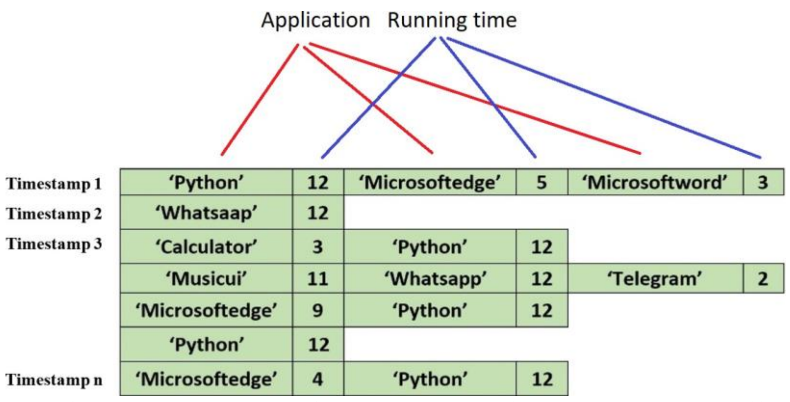


Figure 6: Dependency of applications to execution times. In most cases, the time of use is correlated with the tasks and habits of users.

3.1.2.4. Running applications features

According to Figure (2), the feature vector consists of a static section (f_0, ..., f_{158} in Figure 2) and a dynamic section (f_{159}, ..., $f_?$ in Figure 2). In the dynamic section, features are selected from running applications to show how long the user spends on different programs. Figure 7 shows an example of running applications features for a sample user in different times- tamps.



Timestamp 1	'Python'	12	'Microsoftedge'	5	'Microsoftword'	3
Timestamp 2	'Whatsaap'	12				
Timestamp 3	'Calculator'	3	'Python'	12		
	'Musicui'	11	'Whatsapp'	12	'Telegram'	2
	'Microsoftedge'	9	'Python'	12		
	'Python'	12				
Timestamp n	'Microsoftedge'	4	'Python'	12		

Figure 7: An example of running applications features extracted for a sample user in different timestamps.

An important point about the running applications features is that its length depends on the timestamp and the user because at different intervals, the user
may use different programs. It may be an issue because it results in variable length vectors in different timestamps. This issue was handled by homogenization of variable length feature vectors into fixed length feature vectors as described in Section 3.1.3.

3.1.3. Base profile creation for authenticated user

Now that all of the necessary features have been captured in every timestamp, preprocessing should be performed to creating an exclusive base profile for authenticated user. Figure (8) demonstrates the process of base profile creation for the authenticated user of a system. As the figure shows, in the first step, the dynamic part of the feature vectors is homogenized and reduced to 6 features, and then these homogenized features are concatenated to the static part of the vectors. Now that all vectors have a fixed dimension of 165, we apply a dimension reduction method to the vectors to get a better representation. In the following sections we explain the process for homogenization and dimension reduction, respectively.

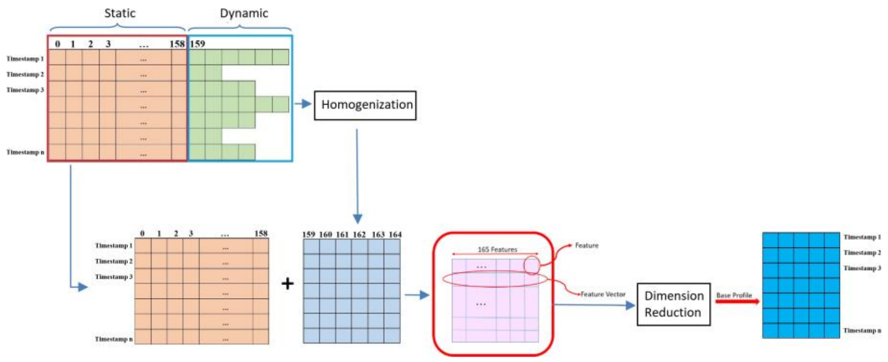


Figure 8: Method of creating base profile for the authentic user

3.1.3.1. Homogenization of variable length feature vectors

As discussed, the features captured in the dynamic section of every vector include the name of the running applications and its counter during a timestamp. These counters increase by one in every 10 seconds of the program execution; hence, there is a list of programs with their functionalities for every timestamp. Homogenization is done to make such variable length vectors in the same size for all timestamps. The homogenization steps are as follows:

1. First, a list of executed applications along all timestamps is created.
2. According to the values of counters for every program, the resultant list is sorted in descending order from the most widely used to the least widely used.
3. The first *n* cells of this list are selected as the “list of frequent programs” (In this study, it was considered that *n* = 6).
4. For every timestamp, a vector is considered with *n* cells, each of which is initialized with the counter of a program on the frequent list in that timestamp if the program is executed.

Figure (9) demonstrates the process of homogenizing the dynamic features.

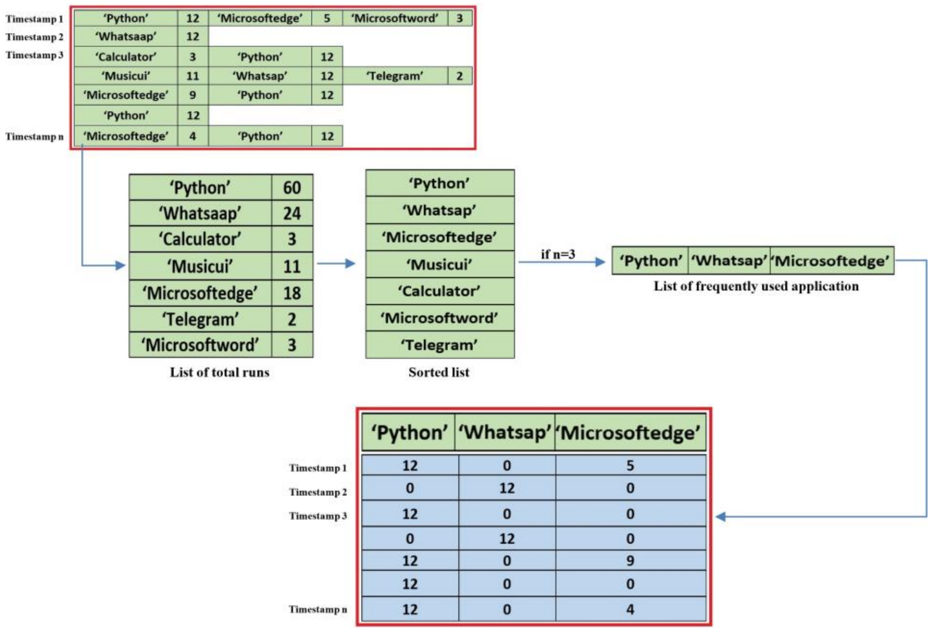


Figure 9: The process of homogenizing dynamic features.

3.1.3.2. Dimensionality reduction for homogenized vectors

After homogenization, there will be *n* + 159 features (159 features in static section and n=6 homogenized features) for every timestamp. The well-known principal component analysis (PCA) (with the retention of 95% of data variance) is used for the dimensionality reduction of feature vectors. The number of resultant dimensions (for the retention of 95% of data variance) varies for every dataset. Therefore, after ap- plying PCA, the resultant dimensions are different for different users. In fact, it can be concluded that the resultant dimensions depend completely on the captured features and partially on the user behavior toward the system. Hence, the mapped dataset is called the user normal profile.

At the end of this step, for each user, we have a set of 165-dimensional vectors that represent the behavior of the user in different timestamps. These vectors are then used as the normal profile for detecting unauthorized users. In Section 3.2, a discriminative model to identify unauthorized users is proposed.

3.2. Unauthorized access detection

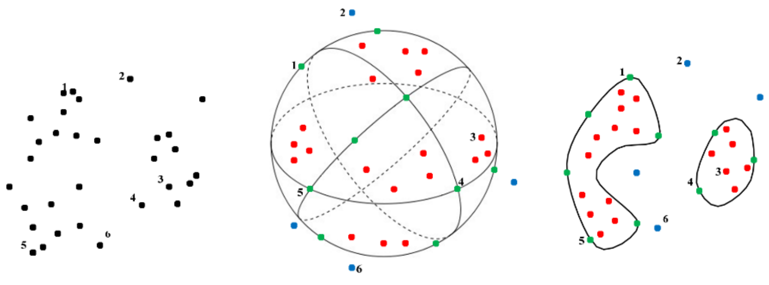
After preprocessing and base profile creation, machine learning techniques are now employed to develop a model for the distinguishing of the authenticated user from any unauthenticated users. In this study, two methods are proposed to model the authenticated user behavioral patterns. In the first method, the well-known One-Class Support Vector Machines (OCSVM) is used to build the model. In the second one, the idea of data clustering is used to identify any unauthorized access. In the following, the proposed methods for distinguishing of the authenticated user from unauthenticated ones are described.

3.2.1. Using anomaly detection approach for distinguishing unauthorized access

In order to discriminate authorized users from the unauthorized ones, we use the well-known OCSVM algorithm. OCSVM is an efficient approach for anomaly detection in which can fit a fitting complex nonlinear boundary between normal and novel data (See Figure 10). In the OCSVM there is a set of data of normal class (set of 165-dimensional homogenized feature vectors from authenticated user captured in different timestamps) and the goal is to test novel data. Also, particle swarm optimization (PSO) is employed to fine tune parameters in the OCSVM classifier. Figure 11 demonstrates the method of training the OCSVM model through the PSO. Normal data (from the authentic user) and novel data (from other users) were utilized to train the OCSVM model in order to deal with false positives. In fact, our motivation is to consider the authenticated user profiles as normal pattern and use the idea of one-class classification to distinguish any novelties in users' activities. The OCSVM model and PSO are depicted in more detail in Figures 10 to 11.

The steps to the development of OCSVM through the PSO are as follows:

1. Initializing the PSO parameters such as the population size (number of particles), number of iterations, and constant coefficients.
2. The PSO algorithm initializes the particles randomly in the first generation.
3. Normal data are used with the available solutions in the population to train the OCSVM model.
4. Normal and anomalous data are first predicted through the trained model to calculate the fitness of particles. After that, the values of area under curve (AUC) are used as the values of the fitness function [30].
5. Based on the fitness obtained for every particle, the local best and the global best are first updated, and then the position of every particle is updated.
6. The termination conditions of the algorithm are checked. Steps 3 to 6 are repeated until the maximum number of iterations is achieved.



(a) (b) (c)

Figure 10: An illustration of how OCSVM works. (a) Synthetic data points in input space R^2, (b) In the first step, OCSVM applies a non-linear transformation of data points to high dimensional feature space and find the smallest enclosing sphere of radius *R*, and (c) in the next step, OCSVM backs the enclosing sphere into the input space to result in nonlinear boundary of data. Any data outside this boundary is known as an anomaly.

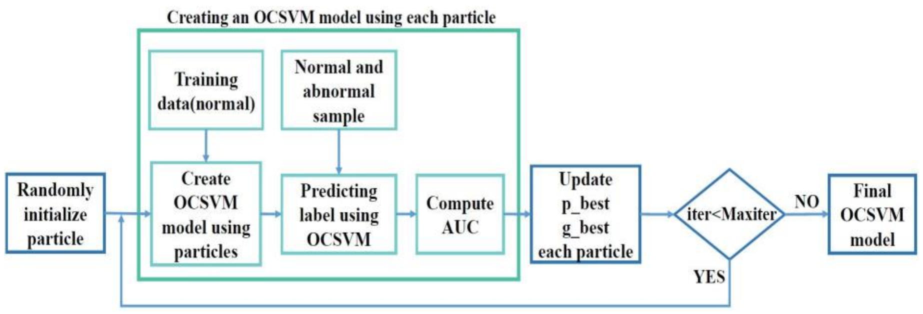

Creating an OCSVM model using each particle
Training data(normal)
Normal and abnormal sample
Randomly initialize particle
Create OCSVM model using particles
Predicting label using OCSVM
Compute AUC
Update p_best g_best each particle
iter<Maxiter
NO
YES
Final OCSVM model

Figure 11: Process of training the proposed OCSVM model through PSO.

3.2.2. Using data clustering for distinguishing unauthorized access

In this paper, an unsupervised technique was also employed to model each user behavioral pattern and authenticate users. In other words, the well- known K-means clustering algorithm was utilized to cluster the authenticated user's behavioral patterns and develop the behavioral pattern of every user exclusively. The idea of clustering originates mainly from differences in the various roles that a user plays while using a system (see Figure 12). A user sometimes has the role of a programmer, a typist, a designer, and a viewer. This idea was taken into account to develop a model through the K-means algorithm.

The normal profile data of authenticated user are used for model development and are clustered by the K-means algorithm. After clustering, a numerical value is used as the threshold that is proportional to the mean Euclidean distance of training data to its cluster center. For a novel data to be tested, it is first assigned to a cluster. If the distance from the novel data to the cluster center which it belongs to exceeds the threshold of that cluster, it is considered as anomaly (Here, unauthenticated user); otherwise, it is considered normal data. As we know, in the K-means clustering, it is essential to select the right number of clusters (K); therefore, the Elbow method is used before clustering to find the right number of clusters. As a result, since there are different users and training data, the value of K is not always constant and may differ with respect to the input data.

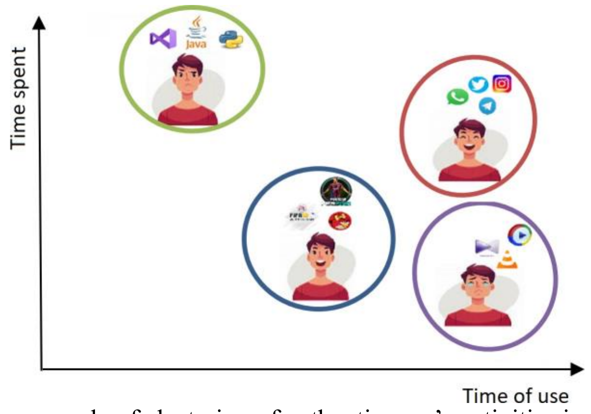


Figure 12: An example of clustering of authentic user's activities in 2 dimensions.

3.3. *Continuous update of the base profile and discriminative model*

In both proposed models (through OCSVM and K-means), according to Figure (1), when a user is using the system, his behavioral feature vectors are captured continuously in every timestamp and passed to the distinguishing models. If the feature vector is considered as normal, it will be stored to be used for updating user normal profile. After stacking feature vector of users in a dataset, the variations in user behavior are watched gradually so that the system can reconstruct a new normal profile set after 30% of variations in the dataset and upgrade the normal profile set along with the discriminative models.

4. Experimental Results

To evaluate the proposed method, we have experimented on several datasets and compared the quality of the results with other methods. The descriptions on the data collection, compared methods, clustering techniques, and evaluation metric are given in the following.

4.1. *Compared methods*

In this study, the following techniques were implemented and then compared on datasets to draw a comparison between different methods. The quality of results is compared against Random Forest method (RF), Naive Bayes method (NB), SVM method [31], Hybrid Naive Bayes and OCSVM method (HNS) [10], the proposed unsupervised method and, the proposed hybrid OCSVM and PSO method. To train the supervised RF, NB, SVM, and HNS models, 80% of the entire datasets including normal and anomalous data (with their labels) were used as training data, and the remaining 20% of datasets were used as test data. Table 3 shows the parameters setting in two proposed unsupervised, and hybrid OCSVM and PSO methods.

Table 3: The parameters setting in the proposed methods.

Proposed method	Technique	Parameter(s)	Value(s)
Hybrid OCSVM and PSO	OCSVM	υ, γ	Calculated by PSO
	PSO	c_0, c_1, c_2	0.2, 0.5, 0.8
		Iteration	100
		Population	50
Unsupervised method		K	Calculated by Elbow
		t	1.1

Due to the lack of an appropriate dataset with the mentioned features, the data were collected by the authors. In this study, 12 individuals collected the research data by using their PCs running Windows 10. The age range is 18-35 years and people have had different jobs such as graphic designer, programmer, gamer, student, architect, translator, etc. Every individual ran the data col- lection tool (designed by the researcher) on their system for nearly 50 hours in total. In fact, this is 50 hours of user activity and system usage. All users use their personal systems, which are the system under investigations, to perform any activity, including job duties, training, entertainment, etc., and this is 50 hours of system usage time by eliminating wasted time (times when the system is without Abandon use). During the use period, for every 2-minute timestamp, a feature vector was saved. Nearly 1500 feature vectors were captured for every user.
To train the models, 80% of the data existing in the authentic user's dataset were considered the normal class, whereas 80% of data existing in the datasets of other users were considered the anomaly class. Moreover, the remaining 20% data in the datasets of users were considered the test data.

4.2. *Evaluation Metric*

The proposed method was evaluated through the following criteria in com- parison with the other methods:

- **Precision and accuracy**: Obtained from Equation (3), precision and accuracy (ACC) are two classical and frequent criteria for evaluating the performance of classifiers:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \qquad (3)$$
$$Precision = \frac{TP}{TP + FP}$$

where TN is the number of negative data classified as negative. If the positive cases are classified as negative cases, they are considered false negatives (FN). Moreover, the FP shows the number of negative cases classified as positives, whereas TP denotes the number of positive cases classified correctly.

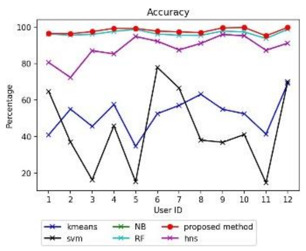
▪ **TPR, FPR and AUC**: Three frequent criteria used in anomaly detection problems are the false positive rate (FPR) in a normal class, true positive rate (TPR) in a normal class, and the area under the curve (AUC) of the ROC diagram. The FPR indicates the percentage of the recognized cases in which unauthorized individuals were identified wrongly as authorized individuals, whereas the TPR denotes the percentage of the recognized cases in which the authorized individuals were identified correctly as authorized. The AUC shows the ratio of TPR to FPR. An increasing AUC is desirable.

▪

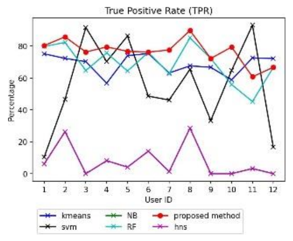
$TPR = \frac{TP}{TP + FN}$	(4)
$FPR = \frac{FP}{FP + TN}$	(5)

4.3. Results

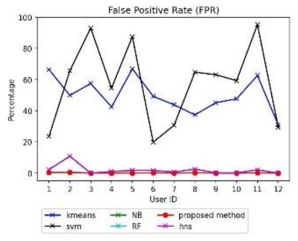
The evaluation results of the proposed methods in comparison with the other methods are presented in Table (4) by reporting the mean ACC, Precision, TPR, FPR, and AUC. The results indicate the superiority of the proposed method to the other techniques. Figure 13 demonstrates the Precision, FPR, TPR, ACC, and AUC for different users.



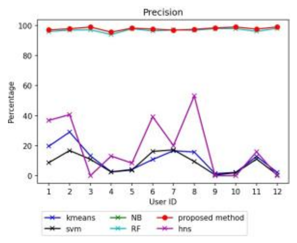
(ACC)



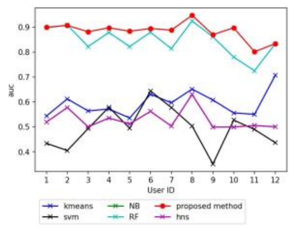
(TPR)



(FPR)



(Precision)



(AUC)

Figure 13: Comparison of the values of Accuracy, Precision, TPR, FPR and AUC for different users.

According to Table (4), the results indicate that the proposed method yielded a high ACC and a low FPR; thus, it is an appropriate method for detecting the anomalous behavior of users. Moreover, the largest AUC was obtained from the proposed method. The results also indicate the superiority of the proposed method to the other techniques. According to Figure 13, it is possible to detect and distinguish people's differences in their habits and roles by using the introduced features in some other modes such as RF. In addition, the proposed hybrid OCSVM and PSO method outperformed the proposed unsupervised method.

Table 4: Comparison of the values of Accuracy, Precision, TPR, FPR and AUC for different methods.

Algorithm	ACC	Precision	TPR	FPR	AUC
SVM [31]	34.68%	8.25%	56.10%	57.23%	0.494%
HNS [5]	88.29%	18.92%	7.64%	1.95%	0.528%
RF	96.48%	96.90%	69.22%	0.18%	0.845%
NB	88.29%	18.92%	7.64%	1.95%	0.528%
Unsupervised method	52.01%	10.82%	68.72%	49.99%	0.593%
Hybrid OCSVM and PSO	**97.83%**	**97.84%**	**76.75%**	**0.12%**	**0.883%**

It is also interesting to notice that the proposed hybrid OCSVM and PSO method has kept the performance of unauthenticated user identification in the same level of accuracy for different users and does not change with a change of user. This may indicate to some extent the insensitivity of the proposed method to the user.

5. Conclusion

With technological advances and changes in people’s lifestyles, they increasingly need to use computer systems on a daily basis. In other words, these systems cover many aspects of human life including occupation, entertainment, education, and personal use. The increasing use of computer systems has al- ways resulted in different problems such as the protection and security of computer systems, for the growing number of computer users would increase the unauthorized access of other individuals. Generally, computer users try to secure their systems against the misuse and unauthorized access of other individuals by defining passwords, fingerprints, facial recognition processes, and other solutions. This study proposed a method based on user behavior to dis- criminate authorized and unauthorized users. In this method, operation- and application-related features of users were employed to create a user profile. A hybrid OCSVM and PSO method were proposed to detect authorized users from unauthorized individuals by using the user profile. A promising future direction is to consider how the user interacts with applications as features.

References

[1] A. Ureña López, F. Mateo, J. Nav´ıo-Marco, J. M. Mart´ınez-Mart´ınez, J. Gómez-Sanch´ıs, J. Vila-Franc´es, A. Jos´e Serrano-López, Analysis of computer user behavior, security incidents and fraud using self-organizing maps, Computers & Security 83 (2019) 38–51.

[2] J. Kim, H. Kim, P. Kang, Keystroke dynamics-based user authentication using freely typed text based on user-adaptive feature extraction and novelty detection, Applied Soft Computing 62 (2018) 1077–1087.

[3] M. Akpinar, M. F. Adak, G. Guvenc, Svm-based anomaly detection in remote working: Intelligent software smartradar, Applied Soft Computing 109 (2021) 107457.

[4] N. Zheng, A. Paloski, H. Wang, An efficient user verification system using angle-based mouse movement biometrics, ACM Transactions on Information and System Security (TISSEC) 18 (3) (2016) 1–27.

[5] M. Corney, G. Mohay, A. Clark, Detection of anomalies from user profiles generated from system logs, in: Proceedings of the Ninth Australasian Information Security Conference, 2011, pp. 23–31.

[6] A. R. Ikuesan, H. S. Venter, Digital behavioral-fingerprint for user attribution in digital forensics: Are we there yet?, Digital Investigation 30 (2019) 73–89. doi:10.1016/j.diin.2019.07.003.

[7] A. A. E. Ahmed, I. Traore, A new biometric technology based on mouse dynamics, IEEE Transactions on Dependable and Secure Computing 4 (3) (2007) 165–179. doi:10.1109/TDSC.2007.70207.

[8] Z. Jorgensen, T. Yu, On mouse dynamics as a behavioral biometric for authentication, in: Proceedings of the 6th ACM Symposium on Infor- mation, Computer and Communications Security - ASIACCS '11, 2011. doi:10.1145/1966913.1966983.

[9] D. Gunetti, C. Picardi, Keystroke analysis of free text, ACM Transactions on Information and System Security (TISSEC) 8 (2005) 312–347.

[10] R. Pokhrel, Anomaly based–intrusion detection system using user profile generated from system logs, Ph.D. thesis, Pulchowk Campus (2016).

[11] P. Pokharel, R. Pokhrel, S. Sigdel, Intrusion detection system based on hybrid classifier and user profile enhancement techniques, in: 2020 Inter- national Workshop on Big Data and Information Security (IWBIS), IEEE, 2020, pp. 137–144.

[12] M. Akpinar, M. F. Adak, G. Guvenc, Svm-based anomaly detection in remote working: Intelligent software smartradar, Applied Soft Computing 109 (2021) 107457. doi:10.1016/j.asoc.2021.107457.

[13] C.-J. Tsai, K.-J. Shih, Mining a new biometrics to improve the accuracy of keystroke dynamics-based authentication system on free-text, Applied Soft Computing 80 (2019) 125–137. doi:10.1016/j.asoc.2019.03.033.

[14] P. Pokharel, S. Sigdel, R. Pokhrel, B. Joshi, Time series-based pattern recognition for anomaly detection from system audit logs, in: 2019 Artificial Intelligence for Transforming Business and Society (AITB), Vol. 1, IEEE, 2019, pp. 1–6. doi:10.1109/AITB48515.2019.8947448.

[15] M. Yıldırım, E. Anarım, Novel feature extraction methods for authentication via mouse dynamics with semi-supervised learning, in: 2019 Innovations in Intelligent Systems and Applications Conference (ASYU), IEEE, pp. 1–6. doi:10.1109/ASYU48272.2019.8946415.

[16] C. Shen, Z. Cai, X. Guan, Y. Du, R. A. Maxion, User authentication through mouse dynamics, IEEE Transactions on Information Forensics and Security 8 (1) (2012) 16–30.

[17] P. Chong, Y. X. M. Tan, J. Guarnizo, Y. Elovici, A. Binder, Mouse authentication without the temporal aspect–what does a 2d-cnn learn?, in: 2018 IEEE Security and Privacy Workshops (SPW), IEEE, 2018, pp. 15–21.

[18] S. Fu, D. Qin, D. Qiao, G. T. Amariucai, Rumba-mouse: Rapid user mouse- behavior authentication using a cnn-rnn approach, in: 2020 IEEE Confer- ence on Communications and Network Security (CNS), IEEE, 2020, pp. 1–9. doi:10.1109/CNS48642.2020.9162287.

[19] M. Antal, E. Egyed-Zsigmond, Intrusion detection using mouse dynamics, IET Biometrics 8 (5) (2019) 285–294.

[20] S. Mondal, P. Bours, Combining keystroke and mouse dynamics for continuous user authentication and identification, in: 2016 IEEE International Conference on Identity, Security and Behavior Analysis (ISBA), IEEE, 2016, pp. 1–8.

[21] I. Traore, I. Woungang, M. S. Obaidat, Y. Nakkabi, I. Lai, Combining mouse and keystroke dynamics biometrics for risk-based authentication in web environments, in: 2012 fourth international conference on digital home, IEEE, 2012, pp. 138–145.

[22] T.-Y. Chang, C.-J. Tsai, W.-J. Tsai, C.-C. Peng, H.-S. Wu, A changeable personal identification number-based keystroke dynamics authentication system on smart phones, Security and Communication Networks 9 (15) (2016) 2674–2685.

[23] M. Yildirim, E. Anarim, Mitigating insider threat by profiling users based on mouse usage pattern: ensemble learning and frequency domain analysis, International Journal of Information Security (2021) 1–13doi:10.1007/ s10207-021-00544-9.

[24] M. Antal, E. Egyed-Zsigmond, Intrusion detection using mouse dynamics, IET Biometrics 8 (5) (2019) 285–294.

[25] P. Chong, Y. Elovici, A. Binder, User authentication based on mouse dynamics using deep neural networks: A

comprehensive study, IEEE Transactions on Information Forensics and Security 15 (2019) 1086–1101. doi:10.1109/TIFS.2019.2930429.
[26] N. Zheng, A. Paloski, H. Wang, An efficient user verification system via mouse movements, in: Proceedings of the 18th ACM conference on Com- puter and communications security, 2011, pp. 139–150. doi:10.1145/ 2046707.2046725.
[27] E. Frias-Martinez, S. Y. Chen, R. D. Macredie, X. Liu, The role of human factors in stereotyping behavior and perception of digital library users: a robust clustering approach, User Modeling and User-Adapted Interaction 17 (3) (2007) 305–337.
[28] J. H. Addae, X. Sun, D. Towey, M. Radenkovic, Exploring user behavioral data for adaptive cybersecurity, User Modeling and User-Adapted Interaction 29 (3) (2019) 701–750.
[29] H. Yakura, T. Nakano, M. Goto, An automated system recommending background music to listen to while working, User Modeling and User- Adapted Interaction (2022) 1–34.
[30] J. Tian, H. Gu, Anomaly detection combining one-class svms and particle swarm optimization algorithms, Nonlinear Dynamics 61 (1) (2010) 303–310. doi:10.1007/s11071-009-9650-5.
[31] A. Widodo, B.-S. Yang, Support vector machine in machine condition monitoring and fault diagnosis, Mechanical systems and signal processing 21 (6) (2007) 2560–2574. doi:10.1016/j.ymssp.2006.12.007.

Parisima Hosseini graduated from Islamic Azad University, Karaj, with a Bachelor's degree in Software Engineering in 2016. Afterward, she pursued her Master's degree in Artificial Intelligence at Beheshti University, Tehran, Iran. Her research interests include machine learning and recommender systems.

Alireza Torabian Raj received his M.Sc. degree in Artificial intelligence from Shahid Beheshti University, Tehran, Iran, in 2021. His research interests include image processing and action recognition.

Ahmad Ali Abin received the B.Sc. in computer engineering from the Iran University of Science and Technology, Tehran, Iran, in 2005. He received the M.Sc. and Ph.D. degrees in computer engineering at the Sharif University of Technology, Tehran, Iran, in 2008 and 2014, respectively. Since 2014, he joined with the Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran. His research interests include pattern recognition, machine learning and neural computing.