Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2018) 26: 1698 – 1709

doi:10.3906/elk-1711-410

Anomaly detection through keystroke and tap dynamics implemented via machine learning algorithms

Hani JAWED, Zara ZIAD, Muhammad Mubashir KHAN*, Maheen ASRAR
Department of Computer Science & Software Engineering, NED University of Engineering & Technology, Karachi, Pakistan

Received: 29.11.2017 • **Accepted/Published Online:** 29.04.2018 • **Final Version:** 27.07.2018

Abstract: In our world of growing machine intelligence and increasing security risks, there is a dire need for authentication to be liberated from password dependency and restrictions. This paper discusses the implementation of keystroke biometrics to enhance security using machine-learning algorithms on both Windows and Android. Our research analyzes a user's behavior for authorization purposes by capturing the user's typing pattern. The system extracts several features from the user's typing pattern to apply unary classification for user behavior analysis so that we can detect unauthorized users. Our system implements machine learning on tap dynamics in Android, allowing both training and prediction and overcoming its computational restrictions.

Key words: Keystroke dynamics, tap dynamics, user behavior analysis, one-class support vector machine, user authentication

1. Introduction

Machine learning applications are making waves in tuning the intelligence of machines when it comes to everyday machine activities. The convenience provided by this intelligence has caused humans to rely more and more on machines for all sorts of activities from simple communications to completely entrusting a business into the hands of a machine. This creates a need for even stronger security systems to guard the systems we have become so dependent on. This research intends to model a machine learning-based computer security system, because what better way is there to tackle growing machine learning capabilities than to create stronger ones?

The problem with the present password security systems is their restrictions in maintaining and securing passwords. If any password information gets into malicious hands, all personal, financial, or commercial data could be at risk of exposure. To address this problem, the modeled system that we present secures a system's password through each authorized user's keystroke analysis. User behavior analytics (UBA) is the tracking, collecting, and assessing of user data and activities using monitoring systems. In our system, UBA is carried out via keystroke dynamics; that is, the system studies the patterns of human behavior when humans interact with the machine. Our focus for UBA is on keystroke dynamics as it is undeniably a recurring and irreplaceable method of interaction between human and machines. The aim is for the application's machine learner to be able to detect anomalous behavior or intruders by differentiating the behavior of a legitimate user's keystroke dynamics from that of an illegitimate user.

This paper is organized as follows: in Section 2, we discuss related work; in Section 3, we present an

*Correspondence: mmkhan@neduet.edu.pk

overview of our approach; in Section 4, we present the various features that we extract from our data set, the algorithms, and the training and testing techniques; in Section 5 we discuss our experimental results as performance evaluation of the applied algorithms; and finally, in Section 6, we derive a conclusion and present possible future research.

2. Related work

In this section, we first review the studies carried out for the methods of classification used for anomaly detection via keystroke dynamics. Secondly, we briefly discuss the work carried out for the implementation of machine learning algorithms on Android, including its limitations and the research so far.

There has been significant work on using keystroke dynamics for the purpose of anomaly detection. Researchers have carried out many studies to evaluate and report the performance of different classifiers on keystroke dynamics. Some researches use multiclass classification, in which the typing samples of multiple users are used to find decision boundaries that can be used to distinguish each user from the others. Others have reviewed the application of unary classification in anomaly detection, where the typing samples of a single user are used to build a model of the user's typing behavior, and when a new typing sample is presented, the detector tests the sample's similarity to the model and outputs an anomaly score.

Forsen et al. [1] first investigated in 1977 whether users could be distinguished by the way they typed their names, and many different techniques and uses for keystroke dynamics have been proposed.

Many studies are conducted to perform a comprehensive survey of the efforts performed in this domain [2]. Research [3] was conducted to compare anomaly-detection algorithms for keystroke dynamics. The authors collected a keystroke-dynamics data set and developed a repeatable evaluation procedure to measure the performance of a range of detectors so that the results could be compared soundly. In our research, we use a similar format as their keystroke-dynamics data set.

The authors of another research paper [4] discussed the reliability of user authentication through keystroke dynamics by conducting a study on a data set of 1254 participants and concluded that the keystroke-dynamics biometric can be used for user verification instead of user identification.

The research conducted in [5] encompasses all studies done up to 2010 regarding keystroke dynamics, for summarizing the evaluations of methods and to provide future recommendations.

Another study carried out in [6] included collecting keystroke dynamics from a number-pad, where the experimental variations were kept tightly controlled by providing the same number pad to all the participants entering the dynamics.

One of the interesting studies in this domain [7] performed a comparison between typing features to calculate interperson and intraperson distances. This research led to the conclusion that it is possible to learn someone's pattern, but a large number of imitation attempts are needed for further study.

As for the second aspect of our study, there is ongoing work on the application of machine learning-based anomaly detection in the Android system as shown in the research conducted [8–10]. The computational performance restrictions are still a significant concern of studies that aim at alternatives to combat these computational restrictions as presented in [11,12]. There has been significantly less work on machine learning in Java, Weka, and LibSVM on the Android platform.

3. Proposed model

Our proposed model is the application of machine learning in assisting security systems for anomaly detection. CyberSleep is both a Windows and an Android application. It works on keystroke pattern recognition based on two separate research fields of keystroke dynamics for a desktop application and tap dynamics for an Android application.

3.1. CyberSleep Windows application

The proposed system can be categorized into four main modules consisting of the GUI module, which provides the user interface and integrates the modules altogether. The data set collection module is integrated with the keylogger program, which captures user keystroke dynamics in the required data set format, and the data are fed to the next module. The machine learner module then implements a one-class support vector machine (OCSVM) as our anomaly-detection algorithm on the data set collected. Finally, the alert system module is based on SMS alert system, which freezes the system until the authorized user's confirmation is received.

3.2. CyberSleep Android application

The tap dynamics authentication system in an Android application serves the purpose of extending our research of machine learning algorithms in Java. It performs the same functionalities as the Windows desktop application. Two machine learning libraries are used on the Android platform for the classification of the keystroke dynamics collected by the keylogger program. One of these is the Waikato Environment for Knowledge Analysis (Weka). Although it contains a number of ML algorithms, it still depends on secondary help for support vector machine (SVM) algorithms. For this reason, the second machine learning library used on Android is LibSVM. It is a library composed of a variety of SVM algorithms that perform unary, binary, and multiclass classification. This library requires its own data set format. The keylogger program stores these dynamics in the format required by the algorithms of this library. The model is trained from the training data set file and then stored in a model file. The model file is then used against the upcoming dynamics to predict the outcomes.

Since training and testing models in the Android environment require a lot of computational power, a modified version of LibSVM is used in our research that runs in the Android JNI framework to effectively reduce computation time.

Figure 1 shows how the CyberSleep system takes the user through the registration process and then enters the training process. During the training process, the system only monitors the user's typing pattern and extracts the relevant features. When the training process is over, the CyberSleep system is finally implemented and can be used to authenticate users.

Figure 2 represents how a keylogger program extracts the relevant information from a user's keystrokes and then uses it to train the machine learning model.

4. Data set and measurement

To implement the above-mentioned ideas, our first step was to collect the keystroke dynamics data of multiple users by using a keylogger program on both Windows and Android. To provide further statistics on our data set, we present different measures calculated from the features collected of the subjects.

The values of mean, median, variance, standard deviation, inner quartile, outer quartile, and interquartile range of the 28 features collected by each of the subjects on Windows and Android platforms are represented in Tables 1 and 2.

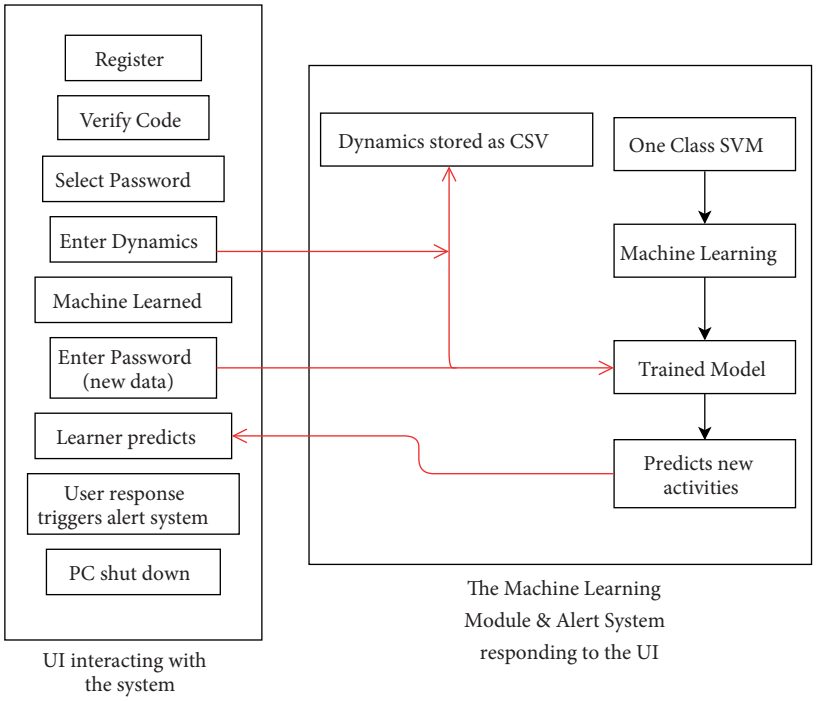

Figure 1. Proposed system of CyberSleep.

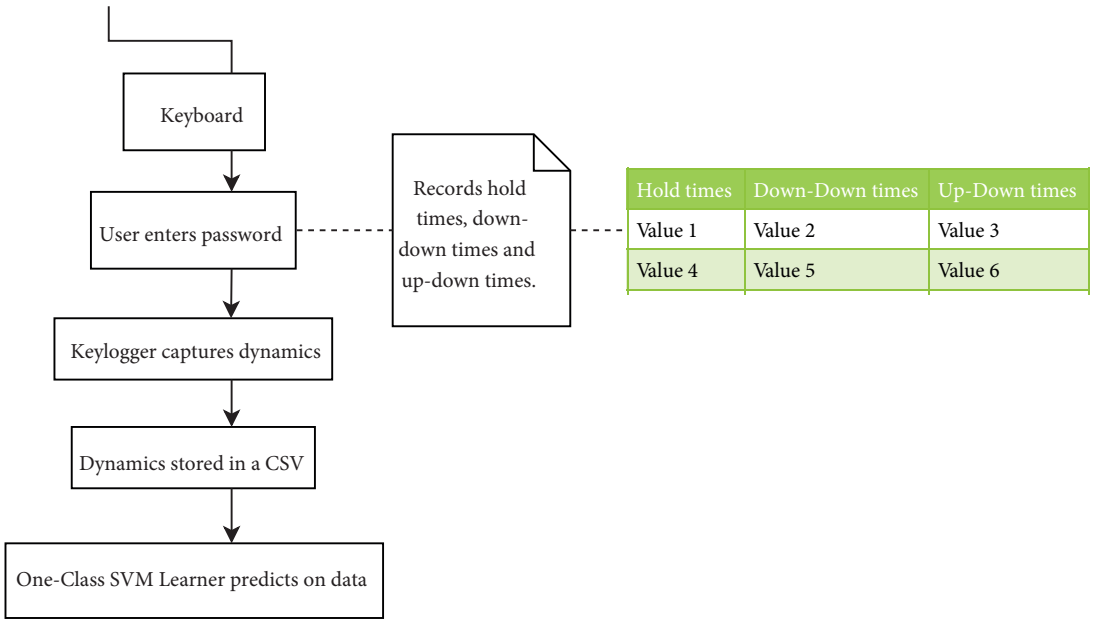

Figure 2. Extraction of keystroke dynamics.

4.1. CyberSleep Windows data set

Since we are working on novelty detection, we decided that all our subjects would type the same password and the timing features of these entries would be calculated, which will help us detect unauthorized access. The password chosen for this purpose was E3n.5Zx.

Table 1. The averages of the 28 features collected by each of the subjects on the Windows platform.

Subjects	Mean	Median	Variance	Standard deviation	Inner quartile	Outer quartile	IQR
Subject 1	176.118	167.428	4636.890	60.296	143.928	196.392	52.464
Subject 2	211.076	183.357	20408.676	118.049	157.151	232.142	74.991
Subject 3	159.032	152.5	2332.136	40.144	134.1875	181.321	47.133
Subject 4	217.010	202.089	12534.451	88.814	175.75	232.580	56.830
Subject 5	160.339	149.660	8192.461	69.126	129.517	171.857	42.339

Table 2. The averages of the 28 features collected by each of the subjects on the Android platform.

Subjects	Mean	Median	Variance	Standard deviation	Inner quartile	Outer quartile	IQR
Subject 1	106.579	102.893	3436.974	43.553	89.828	115.931	26.103
Subject 2	98.939	88.372	18715.665	92.8320	77.086	101.124	24.038
Subject 3	102.283	101.744	738.495	21.930	89.670	113.864	24.193
Subject 4	145.929	141.157	4862.149	51.465	125.354	160.449	35.094
Subject 5	106.735	102.278	6321.475	56.202	90.348	115.740	25.392

Our Windows data set consists of 5 different subjects typing this password into an application that runs a keylogger program in the background. The keylogger program captures the time at which every key of the password is pressed and the time at which it is released. The users were given the choice of using either the shift key or the caps-lock key for capitalizing the necessary letters in the password. As our research involves differentiating different users based on their keystroke dynamics, all users were expected to enter the password using the same means (either shift key or caps-lock key) so that the evaluation of the data set and detectors would be genuine and accurate. This arises from the fact that it is easier to differentiate two users that use different key presses rather than two users with relatively similar typing patterns.

After the down and up times of each key press are captured, the features are extracted by calculating the hold time for every key press (i.e. the difference between the up and down time of the same key), the down-down time (i.e. the difference between the down times of two corresponding keys), and the up-down time (i.e. the difference between the up time and the down time of two corresponding keys).

Thus, the data set involves 28 features (undertaking the enter key and the caps-lock key). From every password entry by each subject, 28 features are captured and stored in a csv-format. Each subject entered the password using the same keyboard to keep the environment real because an authorized person will enter the password on the same keyboard as the authorized user while illegitimate access to a system is being made. We have 5 subjects in our CyberSleep data set and every subject has 600 rows of keystroke data, which leads to a total of $5 \times 600 = 3000$ rows of complete data set.

4.2. CyberSleep Android data set

Keylogging the Android keyboard requires a lot of research but still some Android devices do not allow such access. For the sake of ease in research on any Android device, a number keypad was programmed into the

CyberSleep Android application, which captures the up and down times at every key touch. The features were extracted the same way as they were extracted for the Windows data set. However, in the Android data set, we have an additional feature, which is the finger/thumb area in contact with the touch screen when a user presses a particular key. This adds a great deal of accuracy to our system.

Hence, in the CyberSleep Android data set, we have the same 5 subjects typing the same password (which is 918720) into the number keypad of our application. Each user contributes 600 rows of data, which leads to a total of $5 \times 600 = 3000$ rows of complete data set. This data set is stored in LibSVM format, which is the library used in Android for machine learning purposes, as explained in Section 3.1.2.

5. Algorithms

We present a number of detectors that were used to evaluate the performance of our CyberSleep Windows data set. The best-known machine learning library in Python, Scikit-learn [13], is used for implementing these algorithms.

Although we require a novel detector for our proposed model, for the sake of evaluation, we present results of different multiclass classifiers as well. These detectors include the decision trees classifier, K-neighbors classifier, logistic regression, random forest classifier, multiclass support vector machine (one against one), GaussianNB of naïve Bayes, multilayer perceptrons of neural networks, and the OCSVM (for novelty detection).

5.1. One-class SVM model

The main part of the module is the OCSVM algorithm. We use the implementation of this algorithm as described by Schölkopf et al. in [14]. This algorithm separates all the data points from the origin to feature space F and maximizes the distance from this hyperplane to the origin. This results in a binary function that captures regions in the input space where the probability density of the data is found. Thus, the function returns +1 in a "small" region (capturing the training data points) and –1 elsewhere.

A data set consisting of only positive user data (i.e. data of the authorized user) is input to the algorithm. This method of OCSVM application is called unary classification, where only positive training data are provided and a check is performed to see where the new value lies. It is different from binary classification in that binary classification separates classes by boundaries and UC separates the data by forming 'regions' of similar data. The algorithm is then adjusted according to the data set by tuning the parameters to each of the subjects in our data set to achieve the highest accuracy. The following are the OCSVM parameters:

i) Nu: This parameter decides the number of outliers and the number of support vectors that are allowed.

ii) Kernel: The RBF kernel is selected as we require taking the data to a higher dimension as linear or poly dimensions classifications do not apply to our data. The RBF kernel takes the data to infinite dimensions in order to separate the classes properly.

iii) Gamma: It is a function of the RBF kernel. Gamma is the inverse of the standard deviation of the RBF kernel. It affects the variance inversely. Thus, if gamma increases, the variance decreases, and vice versa.

By inputting a single subject's data into the algorithm and applying the tuned parameters according to that user's data, the execution of the algorithm outputs a model file, which is then used to predict the upcoming data as authorized or unauthorized.

5.2. Training and testing

There are two ways in which our data set is divided for the evaluation of two different group of detectors: unary classification and multiclass classification. The same criteria are followed for both Windows and Android data set evaluations.

5.3. Multiclass classification

The main data set contains 5 subjects and 600 rows of every subject. The training set for multiclass classification comprises 400 rows of each user, which means $400 \times 5 = 2000$ rows of the training set. The validation set is used for tuning the parameters on the training set. Thus, the validation set comprises 100 rows of each user, which leads to $100 \times 5 = 500$ rows of the validation set. The testing set is used for evaluating the accuracy measures on the trained model. Thus, the testing set comprises 100 rows of each user, which leads to a total of $5 \times 100 = 500$ rows of the testing set.

Apart from the rows, there are 28 features that comprise the columns of the CyberSleep data set plus one feature for the label of each user as 1, 2, 3, 4, and 5, respectively. Thus, overall, we have 29 features in our sets. Each multiclass detector is trained on the training set. The related parameters are then tuned using the validation set. Finally, the accuracy measures are calculated using the testing set.

5.4. Unary-class classification

The data set criteria for unary-class classification are a bit different in the sense that it only requires a single subject's data during the training phase, because realistically we will only have the authorized data when we implement the authentication system and the detector must be trained on only the positive data and be able to detect if new data have any outliers. For this purpose of unary-classification, the training set is changed five times, and each time, 400 rows of a single user are used for the training set. The same user's 100 rows plus 25 rows of other 4 users are used as the validation set for parameter tuning. Finally, the testing set comprises 100 rows of the authorized user and 25 rows of each of the remaining 4 unauthorized users, hence $4 \times 25 = 100$ rows of unauthorized users. Thus, the testing set is of 200 rows, which consists of 100 rows of the authorized user and 100 rows of 4 unauthorized users.

The feature column for the training set and validation set only contains 28 features. The label column is not added since there is only one label. However, the testing set contains a test data set and a separate corresponding label data set which has only two labels: $+1$ for the authorized user and -1 for all the unauthorized users, to help in accuracy measures.

6. Experimental evaluation

6.1. Performance measurements

We use accuracy, precision, recall, and F1 score (harmonic mean of precision and recall) to judge the performance of the multiclass classification algorithms. For the case of unary classification, metric evaluation of false acceptance rate (FAR) and false rejection rate (FRR) has also been carried out.

High precision means that the algorithm returns more relevant results than irrelevant ones. Precision (P) is defined as the number of true positives (T_p) over the number of true positives plus the number of false positives (F_p). High recall means that the algorithm returns most of the relevant results. Recall (R) is defined as the number of true positives (T_p) over the number of true positives plus the number of false negatives (F_n).

Harmonic mean of precision and recall yield the F1 score.

$$P = \frac{T_P}{T_P + F_P}$$

$$R = \frac{T_P}{T_P + F_n}$$

$$F1Score = 2 * (\frac{P * R}{P + R})$$

The FAR is the fraction of unauthorized samples that were incorrectly labeled as authorized users. A low FAR means that the system is good at rejecting unauthorized users. The FRR is the fraction of authorized data samples that were incorrectly labeled as an unauthorized sample. A low FRR means that the system is good at accepting authorized users.

6.2. Windows performance measure

The accuracy measures for each of the detectors are evaluated on the CyberSleep Windows data set and the results are explained in Tables 3 and 4.

Table 3. The accuracy measures evaluated on Windows data set by multiclass classifiers.

Algorithm	Parameters	Accuracy	Precision	Recall	F1 score
Multiclass SVM	C = 20, kernel = 'rbf', gamma = 6e-06	0.904	0.9125	0.904	0.902
Random forest	n_estimators = 70	0.892	0.9057	0.892	0.885
Logistic regression	C = 22	0.87	0.87	0.87	0.8689
K-Nearest neighbor	n_neighbors = 4	0.844	0.852	0.844	0.8421
MLP classifier of neural networks	hidden_layer_sizes = 232	0.816	0.839	0.816	0.8116
Decision trees	-	0.802	0.8255	0.802	0.7947
GaussianNB of naïve Bayes	-	0.65	0.640	0.65	0.6086

Table 4. The accuracy measures evaluated on Windows data set by unary-class classifier (OCSVM).

One-class support vector machine							
Subject	Parameters	Accuracy	Precision	Recall	F1 score	FAR	FRR
Subject 1	Nu = 0.26, kernel = 'rbf', gamma = 6e-06	0.85	0.8591	0.85	0.849	0.07	0.23
Subject 2	Nu = 0.27, kernel = 'rbf', gamma = 5e-06	0.885	0.9027	0.885	0.8837	0.01	0.22
Subject 3	Nu = 0.124, kernel = 'rbf',gamma = 1e-05	0.825	0.8276	0.825	0.8246	0.13	0.22
Subject 4	Nu = 0.06, kernel = 'rbf', gamma = 2e-06	0.9	0.9014	0.9	0.8999	0.07	0.13
Subject 5	Nu = 0.14, kernel = 'rbf', gamma = 9e-06	0.71	0.7107	0.71	0.7097	0.32	0.26

Table 3 represents the accuracy, precision, recall, and F1 score evaluated on multiclass classifiers. From the evaluation results, we conclude that the multiclass support vector machine (one-against-one approach) proved to be the best classifier with an accuracy of 90.4%, followed by the random forest classifier with an accuracy of 89.2% and so on. The evaluation results reported in Table 3 are for the Windows data set.

Table 4 represents the accuracy, precision, recall, and F1 score evaluated on the unary-class classifier called OCSVM. This model was trained five times, once for each subject's training data, and then the model was tuned according to the particular subject's data, setting different values of parameters every time. Subject 4's data proved to be the most accurate, giving an accuracy of 90%. Although multiclass classifiers have given good results on our Windows data set, our proposed model requires novelty detection, so our chosen detector for this reason is OCSVM for our system.

6.3. Android performance measures

The accuracy measures for each of the detectors were evaluated on the CyberSleep Android data set and the results are explained in Tables 5 and 6.

Table 5. The accuracy measures evaluated on Android data set by multiclass classifiers.

Algorithm	Parameters	Accuracy	Precision	Recall	F1 score
Random forest	n_estimators = 182	0.982	0.9828	0.982	0.9819
Decision trees	-	0.968	0.9692	0.968	0.9679
Multiclass SVM	C = 11, kernel = 'rbf', gamma = 2e-05	0.942	0.9436	0.942	0.9417
K-Nearest neighbor	n_neighbors = 5	0.932	0.9327	0.932	0.9316
Logistic regression	C = 134	0.9	0.901	0.9	0.8987
GaussianNB of naïve Bayes	-	0.896	0.921	0.896	0.8924
MLP classifier of neural networks	hidden_layer_sizes = 266	0.856	0.8658	0.856	0.8560

Table 6. The accuracy measures evaluated on Android data set by unary-class classifier (OCSVM).

One-class support vector machine							
Subject	Parameters	Accuracy	Precision	Recall	F1 score	FAR	FRR
Subject 1	Nu = 0.14, kernel = 'rbf', gamma = 5e-06	0.92	0.9242	0.92	0.9197	0.03	0.13
Subject 2	Nu = 0.34, kernel = 'rbf', gamma = 1e-05	0.69	0.7863	0.69	0.6615	0.02	0.6
Subject 3	Nu = 0.05, kernel = 'rbf', gamma = 1e-06	0.89	0.8906	0.89	0.8899	0.09	0.13
Subject 4	Nu = 0.06, kernel = 'rbf', gamma = 2e-06	0.99	0.99	0.99	0.99	0.02	0.01
Subject 5	Nu = 0.06, kernel = 'rbf', gamma = 8e-06	0.76	0.7758	0.76	0.7564	0.36	0.12

Table 5 represents the accuracy, precision, recall, and F1 score evaluated on multiclass classifiers. From the evaluation results, we conclude that random forest proved to be the best classifier with an accuracy of 98.2%, followed by the decision tree classifier with an accuracy of 96.8% and so on.

Table 6 represents the accuracy, precision, recall, and F1 score evaluated on the unary-class classifier called OCSVM. This model was trained five times, once for each subject's training data, and then the model was tuned according to the particular subject's data, setting different values of parameters every time. Subject 4's data proved to be the most accurate, giving an accuracy of 99%.

6.4. Graphical evaluation

The data points of keystroke dynamics of different subjects from the Windows data set were graphically plotted using scatter plots [13] for the analysis of correlation between the data, analyzing how far or close points are when plotted in space.

Figure 3 represents a matrix of scatter plots between different features of two subjects' data of the Windows data set. Subject 4 is used as the authorized user, represented by blue points, while Subject 2 is used as the unauthorized user, represented by red points.

Figure 4 represents the same concept as Figure 3 except that there is an additional feature, called finger area, that denotes the area of the finger in contact with the smart screen as the user enters the password data. Subject 4 is used as the authorized user, represented by blue points, while Subject 1 is used as the unauthorized user, represented by red points.

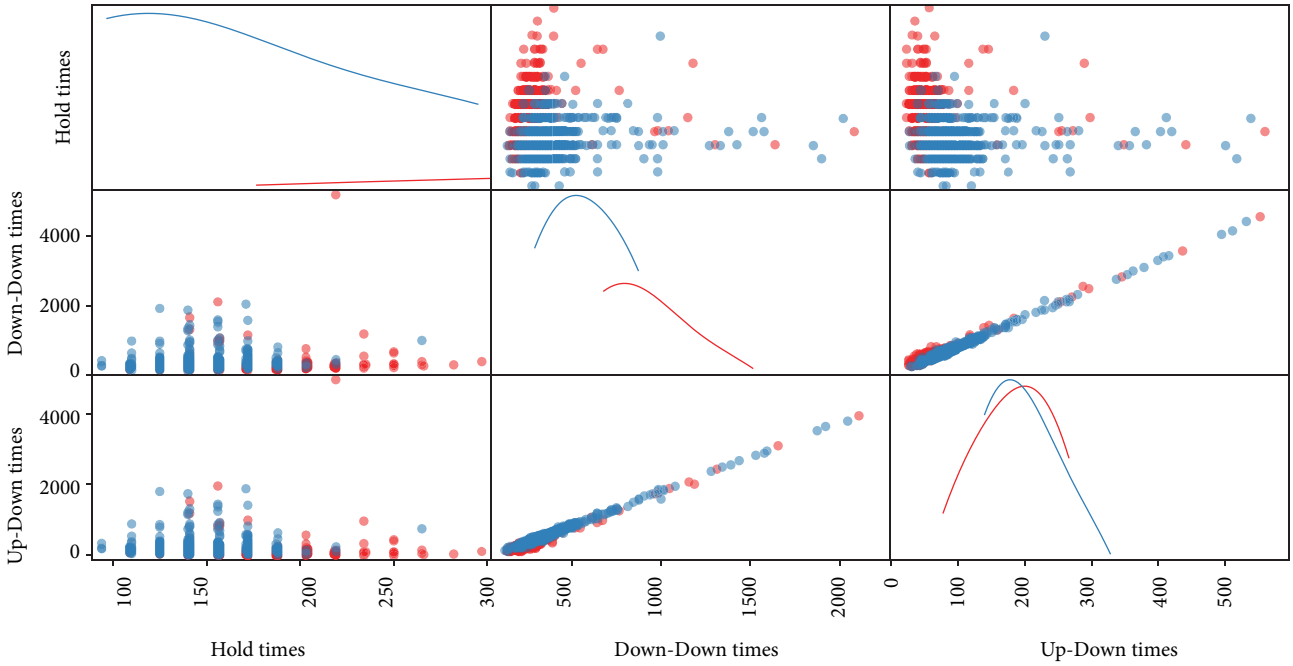

Figure 3. Plotting of keystroke dynamics of Windows data set of two subjects.

Figures 5 and 6 represent the accuracy measures of each subject's Windows and Android data sets when trained on the OCSVM anomaly detector as explained in Section 4.3.2.

Although multiclass classifiers have given good results on our Android data set, our proposed model requires novelty detection, so our chosen detector for this reason is OCSVM for our system.

7. Conclusion

In light of the proposed model and evaluation metrics, we can say that a user can implement our system on his desktop PC or Android device with the expectation that it will provide him with an authentication system and an additional level of security, where the system studies the typing pattern of the user as well as the correctness of the password. The user can expect his desktop system to be as accurate as 90% or even more, while his Android device can provide accuracy of up to 99% depending on the typing rhythm of the subject/user. There are two possible applications of our proposed model. It can be integrated with the Windows or Android authentication system or it can be applied to any application that the user wishes to secure. In the second case, our system could be integrated with the specific application as a service.

Future work can include expanding the study of behavioral biometrics to study the user's mouse dynamics. Further work can include studying the user's file pattern usage to recognize if unusual activity has occurred.

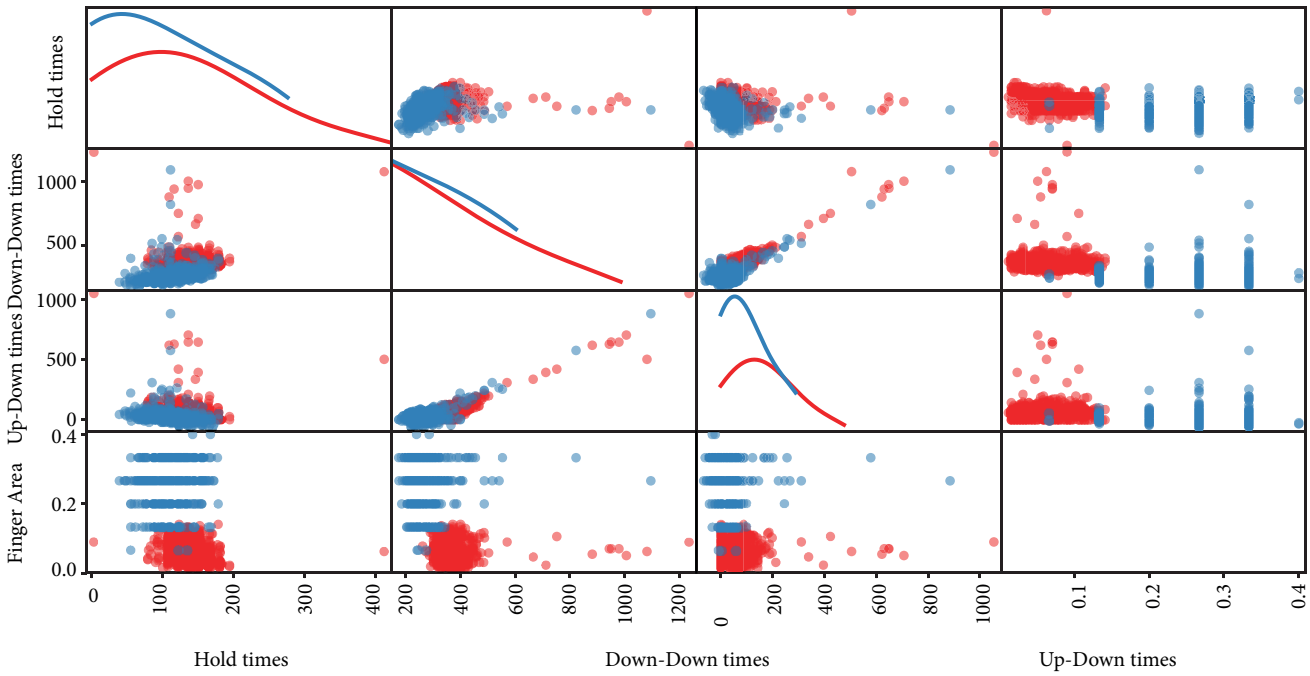

Figure 4. Plotting of keystroke dynamics of Android data set of two subjects.

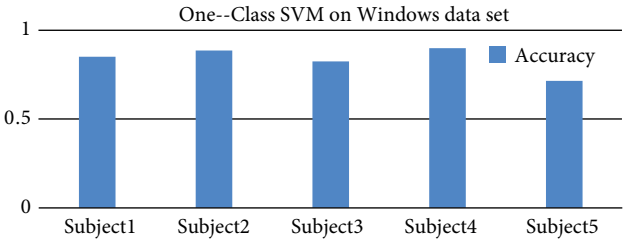

Figure 5. The accuracy each subject for Windows data set when trained on OCSVM.

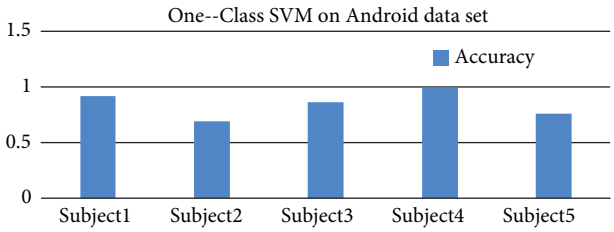

Figure 6. The accuracy of each subject for Android data set when trained on OCSVM.

This technique can be used to detect a hacker's malicious attempts to break into a system.

Acknowledgments

The Department of Computer Science and Software Engineering, NED University of Engineering and Technology, Karachi, Pakistan, supported this research project. We would also like to express our gratitude to Sadaf Iqbal Behlim, who shared her expertise during the course of this research.

References

[1] Forsen G, Nelson M, Staron R, Personal Attributes Authentication Techniques. Technical Report RADC-TR 77-333. Rome, NY, USA: Rome Air Development Center, 1977.

[2] The PS, Teoh ABJ, Yue S. A survey of keystroke dynamics biometrics. Sci World J 2013; 24: 408280.

[3] Killourhy KS, Maxion RA. Comparing anomaly-detection algorithms for keystroke dynamics. In: IEEE/IFIP 2009 International Conference on Dependable Systems & Networks; 29 June–2 July 2009; Lisbon, Portugal. New York, NY, USA: IEEE. pp. 125-134.

[4] Douhou S, Magnus JR. The reliability of user authentication through keystroke dynamics. Stat Neerl 2009; 63:432-449.

[5] Crawford H. Keystroke dynamics: characteristics and opportunities. In: 2010 Eighth International Conference on Privacy, Security and Trust; 17–19 August 2010; Ottawa, Canada. New York, NY, USA: IEEE. pp. 205-212.

[6] Maxion RA, Killourhy KS. Keystroke biometrics with number-pad input. In: IEEE/IFIP 2010 International Conference on Dependable Systems & Networks; 28 June–1 July 2010; Chicago, IL, USA. New York, NY, USA: IEEE. pp. 201-210.

[7] Rundhaug FEN. Keystroke dynamics-Can attackers learn someone's typing characteristics? MSc, Gjøvik University College, Gjøvik, Norway, 2007.

[8] Antal M, Szabó LZ, László I. Keystroke dynamics on Android platform. In: 8th International Conference of Interdisciplinarity in Engineering, INTER-ENG; 9–10 October 2014; Tirgu-Mures, Romania. Amsterdam, the Netherlands: Elsevier. pp. 820-826.

[9] Alghamdi SJ, Elrefaei LA. Dynamic user verification using touch keystroke based on medians vector proximity. In: IEEE 7th International Conference on Computational Intelligence, Communication Systems and Networks; 3–5 June 2015; Riga, Latvia. New York, NY, USA: IEEE. pp. 121-126.

[10] Sun L, Wang Y, Cao B, Yu PS, Srisa-an W, Leow AD. Sequential keystroke behavioral biometrics for mobile user identification via multi-view deep learning. In: European Conference on Machine Learning and Knowledge Discovery in Databases; 18–22 September 2017; Skopje, Macedonia. pp. 228-240.

[11] Sahs J, Khan L. A machine learning approach to Android malware detection. In: 2012 European Intelligence and Security Informatics Conference; 22–24 August 2012; Odense, Denmark. New York, NY, USA: IEEE. pp. 141-147.

[12] Grant H. TapDynamics: Strengthening User Authentication on Mobile Phones with Keystroke Dynamics. Technical Report. Stanford, CA, USA: Stanford University, 2014.

[13] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V et al. Scikit-learn: Machine learning in Python. J Mach Learn Res 2011; 12: 2825-2830.

[14] Schölkopf B, Platt JC, Shawe-Taylor JC, Smola AJ, Williamson RC. Estimating the support of a high-dimensional distribution. Neural Comput 2001; 13: 1443-1471.