One-Class Models for Continuous Authentication Based on Keystroke Dynamics

Maria Kazachuk, Alexander Kovalchuk, Igor Mashechkin, Igor Orpanen, Mikhail Petrovskiy(✉), Ivan Popov, and Roman Zakliakov

Computer Science Department, Lomonosov Moscow State University, MSU, Vorobjovy Gory, Moscow 119899, Russia
{kazachuk,kovalchuk,orpanen,ivan,zakliakov}@mlab.cs.msu.su,
{mash,michael}@cs.msu.su

Abstract. In this paper we discuss an applied problem of continuous user authentication based on keystroke dynamics. It is important for a user model to discover new intruders. That means we don't have the keystroke samples of such intruders on the training phase. It leads us to the necessity of using one-class models. In the paper we review some popular feature extraction, preprocessing and one-class classification methods for this problem. We propose a new approach to reduce dimensionality of a feature space based on two-sample Kolmogorov-Smirnov test and investigate how the quantile-based discretization technique can improve the one-class models' performance. We present two algorithms, which have not been used for keystroke dynamics before: Fuzzy kernel-based classifier and Random Forest Regression classifier. We conduct experimental evaluation of the proposed approach.

Keywords: Keystroke dynamics · User authentication · Kolmogorov-Smirnov test · Quantile discretization · Fuzzy classification · Random forest regression classification

1 Introduction

Nowadays computer systems play a very important role in people's life. These systems are used to store, search and process information. Therefore, it is essential to employ a high level of protection against unauthorized access on these systems.

One of the best known security mechanisms is authentication. It allows a system to confirm that the user is who he claims to be. The reliability of this process depends on the information used for user authentication. Authentication employs several factors, which are usually one of the following: knowledge, ownership, physiological biometrics and behavioral biometrics. Using the knowledge and the ownership factors for authentication has a major drawback, that information can be lost, stolen or divulged.

Physiological biometric images, such as fingerprints, retina, the geometry of the face and hands, are given to people by birth and can not be changed by the will of its owner. Their use for authentication features are highly reliable, but they need additional equipment, and the theft of data samples can cause irreparable damage to the security.

Behavioral biometric samples include those human characteristics and traits that appear when the user performs a certain set of actions. Such data systems include voice, handwriting and gait authentication. Behavioral characteristics can be easily changed by the will of the owner, and it is almost impossible to betray your secret behavioral pattern. Today these systems are widely developed, but their quality is inferior to other authentication systems.

Use of continuous authentication systems allows us to detect intruders when they try to interact with a machine. It's possible that an intruder isn't known, so the construction of a multi-class model, that recognizes a specific user from a closed set, can cause an impostor to remain undetected. Therefore, it is necessary to research the methods that build one-class models for each legitimate user. When performing continuous authentication, model of the legitimate user is compared to the behavior of the current user. Legitimacy of the current user is determined based on this comparison.

This paper features an applied problem of continuous user authentication based on keystroke dynamics, given an assumption of inavailability of illegitimate users' data. This paper is structured into the following sections. Section 2 describes existing keystroke dynamics research. Section 3 describes our approach to enhance keystroke dynamics with better accuracy. Section 4 includes experiments. Section 5 ends this paper with final conclusions on keystroke dynamics.

2 Survey

User's keyboard interaction can be described as key press and release timing information. To collect this data OS tools [6,11–13,15] and web browser tools [1,4] can be used.

Collected data is split into windows, which contain events to be processed into a single feature vector. We have determined several events, which force a new window to be created: exceeding the maximum size of a window or the maximum pause between windows, active process change. In [15] the authors suggest to ignore windows with the number of events below the threshold.

Existing feature extraction approaches include analysis of single key presses and consecutive key presses, called n-graphs. The most frequently used n-graphs are digraphs ($n = 2$) and trigraphs ($n = 3$).

Hold time ($t_i^{up} - t_i^{down}$) and latency ($t_{i+1}^{down} - t_i^{up}$) are calculated for every key (Fig. 1). For n-graphs a subset of the following features is used: $\{(t_{k+n-1}^{up} - t_k^{down})$, $(t_{k+n-1}^{down} - t_k^{up})$, $(t_{k+n-1}^{down} - t_k^{down})$, $(t_{k+n-1}^{up} - t_k^{up})\}$, here k is an index of n-graph in the current window [2–5,11–13,15,16].

Each key and each n-graph produces one or more features in the feature vector, which are equal to the mean or variance of the corresponding statistic (hold time, latency time). Features of keys and key sequences, which are not present in a window, are filled with zeros.

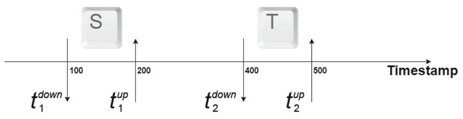

Fig. 1. Key presses and releases

The constructed feature space is high-dimensional, but not all of the extracted features have a significant impact on the classification results. The most common techniques for dimensionality reduction used in related problems are principal component analysis (PCA) [3] and random search approaches such as genetic algorithm (GA) [6,13], particle swarm optimization (PSO) [6,13] and gravitational search algorithm (GSA) [5]. However, our experiments showed that the use of PCA in most cases causes even worse accuracy of the classification, which can be explained by the presence of non-linear correlation between variables. Random search feature selection algorithms are rather computationally difficult.

Some classification algorithms might work considerably better, if features are standardized. This is done by subtracting feature's expected value Ex and dividing the difference by its standard deviation $\sqrt{Dx}$. Both expected value and standard deviation are determined on the training set.

The most commonly used classification algorithms in the field of arbitrary text keystroke dynamics are: metric (one-class KNN [16]) and probabilistic (one-class SVM [1,16], Gaussian mixture models [4], Bayes networks [2]) methods. The best results are achieved in the reviewed papers by using the one-class KNN and one-class SVM classifiers.

3 Proposed Approach

Our approach uses the most popular methods to define a feature space, suggests some new ways of feature selection and preprocessing, and introduces several new classifiers, which have not been used in this task before.

3.1 Feature Space

To construct a feature space, which will describe user's interaction with the keyboard, we propose to combine the analysis of single keys and digraphs. The sequence of collected events is split into windows. Window size is required to be between the fixed minimum and maximum size. If a user is inactive for a specified maximum pause between sequential events, then a window split is forced. These parameters are chosen experimentally.

In order to reduce the dimension of the feature space, we propose to keep only the most frequent digraphs and keys, where frequencies are determined on the training set.

For each of the remaining keys we propose to calculate hold time; for each of the remaining digraphs — the up-up and down-up latency. Also, we divide

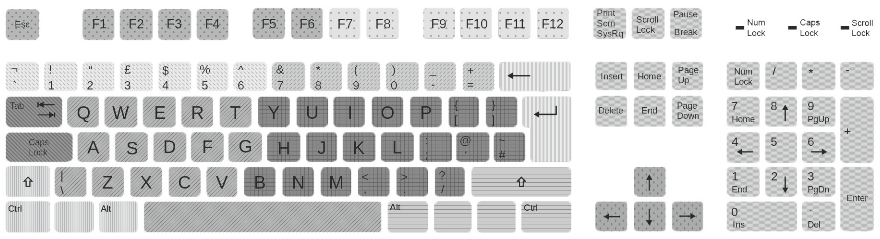

Fig. 2. The proposed 12 key groups

keys into 12 groups (Fig. 2), for each of them hold times are calculated. Some special features, used in this work, are frequency of key presses, and the ratio of the number of left Shift key presses to the number of combined left and right Shift key presses.

3.2 Feature Preprocessing

After feature extraction we examine several feature preprocessing methods: dimensionality reduction of the feature space through selection of the most stable features, and quantile-based feature discretization. To reduce dimensionality of the extracted feature space we select only those features, which have insignificant variance over time. To achieve this we propose two-sample Kolmogorov-Smirnov test. It is used to evaluate a hypothesis, that two samples have the same distribution. The first sample contains unaveraged feature values for each window and the second sample contains congregated unaveraged values for all windows in the training set.

Let $F_{1,n}$ be the distribution function of a feature in the current window, where n is the number of occurrences of the feature in the current window. Let $F_{2,m}$ be the distribution function of the feature for all windows in the training set, where m is the number of occurrences of the feature in the training set. The main goal is to find such λ, that the following inequality is true:

$$\sqrt{\frac{nm}{n+m}}D_{n,m} \geq \lambda, \tag{1}$$

where $D_{n,m} = sup_x \left|F_{1,n}(x) - F_{2,m}(x)\right|$.

Next we use the quantile table of the Kolmogorov distribution to find a p-value, that corresponds to λ. This p-value represents the probability of rejecting the hypothesis, that these two samples have the same distribution.

To accumulate p-values across all windows we suggest Fisher's method, which combines all p-values into single test statistic:

$${\chi_{2k}}^2 \approx -2\sum_{i=1}^{k} \ln(p_i), \tag{2}$$

where k — number of p-values, p_i — i-th p-value.

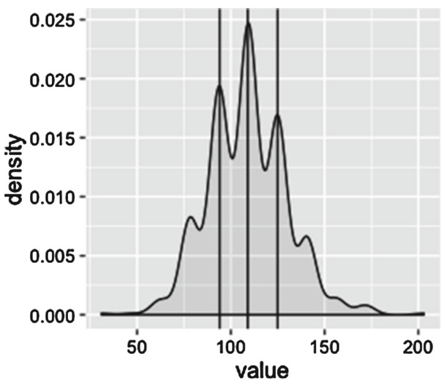

Fig. 3. Quantile-based discretization on 4 intervals

Subsequently the chi-squared distribution table is used to find a corresponding p-value with $2k$ degrees of freedom. The resulting p-value may be used to perform feature selection. We have studied selection of N features with the highest p-values and selection of all features, which exceed threshold p-value. After performing feature selection all unaveraged features are substituted with their mean value.

One of the most popular feature preprocessing techniques used for multimodal distributions is the quantile discretization. This approach has not been applied to the keystroke data before. For each feature in the training set we calculate k quantiles of order $\frac{1}{k}, \frac{2}{k}, ..., \frac{k-1}{k}$, which form several intervals (bins): $(-\infty, \frac{1}{k}), (\frac{1}{k}, \frac{2}{k}), \ldots, (\frac{k-1}{k}, +\infty)$. Then each feature value is replaced with the corresponding bin index, which it falls into.

Unlike the approach in [11], where it is stated, that digraph hold time has normal distribution, we assume most of the features, associated with key presses have multimodal distribution (Fig. 3). Quantile discretization method takes this into account, and subsequent experiments confirm our hypothesis.

3.3 Building User Model

As mentioned before, we consider only one-class classification methods. Fuzzy Kernel-based Method for Outliers Detection and feature values estimator based on Random Forest Regression were introduced for our problem.

We have developed Fuzzy kernel-based method [14] earlier for the task of outliers mining for intrusion detection. It inherits the ideas of SVM, but instead of looking for a crisp sphere in the RKHS feature space, we suggest to search for a fuzzy sphere including all RKHS data images. This problem can be viewed as calculating single fuzzy cluster in the RKHS feature space using possibilistic fuzzy clustering approach. In this case the fuzzy membership can be described as a measure of "typicalness" of data instances. Keystrokes with low "typicalness" are considered outliers. Changing the threshold does not lead to the recalculation of the models, as it was done for SVM and kernelized distance-based algorithms. Mathematically, the problem is to find $\min\limits_{U,a,\eta} J(U,a,\eta)$:

$$J(U,a,\eta) = \sum_{i=1}^{N} u_i^m (\phi(x_i) - a)^2 - \eta \sum_{i=1}^{N} (1 - u_i)^m, \tag{3}$$

where a is a center of the fuzzy cluster in the RKHS feature space; N is a number of instances in the initial feature space X; U is a membership vector, where $u_i \in [0,1]$ is membership of the image $\phi(x_i)$ and besides the "typicalness" of datum x_i; m is fuzzyfier and η – parameter, that controls the size of cluster.

After the minimization of the functional, for each training sample $x_i \in X$ the measure of its "typicalness" u_i is calculated. The calculation of this value for a new item x is done as follows:

$$u(x) = \left[1 + \left(\frac{\sum\limits_{j=1}^{N} u_j^m \sum\limits_{i=1}^{N} u_i^m K(x_i, x_j)}{\eta \left(\sum\limits_{i=1}^{N} u_i^m \right)^2} - 2 \frac{\sum\limits_{i=1}^{N} u_i^m K(x, x_i)}{\eta \sum\limits_{i=1}^{N} u_i^m} + \frac{K(x,x)}{\eta} \right)^{\frac{1}{m-1}} \right]^{-1}, \tag{4}$$

where N is a number of records in training set, $K(x,y)$ is a kernel for x and y.

Another algorithm that we have developed is a one-class classifier, based on the Random Forest Regression [8] used for the approximation of values of all features. The degree of normality of the data is calculated in accordance to how well it was approximated by the model. The idea is similar to Replicator Neural Networks [9], but instead of neural nets the Random Forest is used as an approximator. Regression trees are built as follows. Suppose, that we have p inputs and N observations with response $(x_i, y_i); x_i = (x_{i1}, \ldots, x_{ip}); i = \overline{1, N}$. The algorithm chooses splitting variables, split points and trees topology and we get M regions $R_1, \ldots, R_M$. We model a response (where c_m is a constant in each region, its best approximation is $\hat{c}_m = average(y_i \mid x_i \in R_m)$):

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m). \tag{5}$$

When we use Random Forest for regression, we build an ensemble of trees $\{T(x;\theta_b)\}_1^B$, using the subset of the training data by recursively splitting the nodes on the best split-point among a subset of m random features until the minimum node size is reached. θ_B characterizes the b-th tree in terms of split variables, cutpoints at each node and terminal-node values. After this step, the random forest predictor is:

$$\hat{f}_{rf}^B(x) = \frac{1}{B} \sum_{b=1}^{B} T(x;\theta_b). \tag{6}$$

A one-class classifier can be built by creating a set of p regressors, each for a separate variable, where other variables are used as predictor values:

$$\hat{f}^B_{x_i}(x) = \frac{1}{B}\sum_{b=1}^{B} T(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_p; \theta_b). \tag{7}$$

When we get a new observation, it can be estimated using the built predictors:

$$(x_1, \ldots, x_p) \to (\hat{f}^B_{x_1}, \ldots, \hat{f}^B_{x_p}). \tag{8}$$

The resulting decision function is defined as a reconstruction error:

$$DF(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{f}^B_{x_i})^2. \tag{9}$$

Note that all features must be standardized. The main advantage of this algorithm is the ability to detect non-linear correlation between features and to ignore irrelevant features.

4 Experiments

The most suitable dataset we have found is the Villani keystroke public dataset [12,15], which consists of 144 users, who were instructed to respond to open-ended essay questions and produced 1345 samples overall. The following data was collected for each user: platform (desktop or laptop), gender, age group, handedness and awareness of data collection. In our work only the following data was used: keycode, keystroke press and release times. We only use 53 of 144 users because they provide 20 or more feature vectors when using parameters discussed below. For each of the 53 users half of their data was used as the training set and other half of the data was combined with the data of all the remaining users to be the test set.

Area under the ROC-curve (AUC) was used to evaluate classification results. The AUC is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one [7]. The distinguishing feature of the AUC is the invariance with respect to the proportion of positive and negative samples in the test set. We have used average AUC among all users to evaluate final results.

The essential part of conducting experiments is to select optimized parameters for the feature extraction, feature preprocessing and classification algorithms. We have conducted preliminary experiments on a dataset collected by ourselves to choose some hyperparameters. 20 users had been working for 10 days, 1 h per day, on 3 different configurations of computers doing their usual work: programming, text and presentation preparation, Internet browsing. For feature extraction algorithms optimal parameters were: minimum window size — 300 events; maximum window size — 500 events; maximum pause between events

— 40 s; amount of the most popular keys — 50; amount of the most popular digraphs — 100. Next we have chosen optimal parameters for the feature preprocessing algorithms. Feature selection based on stability depends either on the amount of the most stable features or on the threshold level of significance. Best results in our experiments were achieved by using the level of significance more than 0.1. Quantiles discretization has only one parameter: number of quantiles, which has the default value of 10.

Subsequent task is to choose optimal parameters for classification algorithms, which we have done for each of the following classifiers:

1. One-class KNN [16] is based on evaluating distances between samples in the training set. Threshold distance, which determines the maximum distance of an original class element, is chosen as the distance to the neighbor with index $\lfloor p * N \rfloor$ in the sorted array of k neighbors, where $p \in [0, 1]$.
2. One-class SVM classifier [1,16] is based on the construction of a hypersphere, which encompasses most of the user's data. In the course of classification elements, which fall inside this hypersphere are considered to belong to the original class. The parameters of the algorithm are: kernel type, kernel coefficient γ, an upper bound on the fraction of training errors and a lower bound of the fraction of support vectors ν.
3. Kernel principal component analysis (KPCA) [10] extends standard PCA to non-linear data distributions, which can also be used for one-class classification. The parameters of the algorithm are: kernel type, kernel coefficient γ, number of principal components n.
4. Replicator Neural Network (RNN) [9] is a forward propagation neural network with the same number of input and output neurons, which reconstructs

Table 1. Optimal values for the classification algorithms

	Parameter	Standardization	Selection using stability	Quantile discretization
KNN	k	4		
SVM	*kernel*	radial basis function (RBF)		
	γ	$1/N_{features}$		
	ν	0.1		
KPCA	kernel	radial basis function (RBF)		
	γ	$1/max\|x_i - x_j\|$, where x_i, x_j are in training set, $i \neq j$		
	n	10	10	5
RNN	*hidden*	three hidden layers (16 – 4 – 16)		
	iterations	500		
Fuzzy	*kernel*	radial basis function (RBF)		
	γ	$1/max\|x_i - x_j\|$, where x_i, x_j are in training set, $i \neq j$		
	m	1.5		
RFR	*nTrees*	3	3	20
	minSize	3	10	10

Table 2. Results on the Villani dataset

	Standardization	Selection using stability	Quantile discretization
KNN	0,5392	0,5596	0,6656
SVM	0,7665	**0,7933**	0,8861
KPCA	0,7770	0,7898	0,9081
RNN	0,7641	0,7918	0,8833
Fuzzy	0,7737	0,7888	**0,9122**
RFR	**0,7805**	0,7898	0,8991

original class elements better than outliers. The parameters of the algorithm are: a number of hidden layers of the neural network, a number of neurons on hidden layers, a number of iterations.

5. Fuzzy classification method is described in the previous section. The parameters of the algorithm are: kernel type, kernel coefficient γ, affiliation level's decrease rate m based on the distance to the center of the cluster.
6. Random Forest Regressor (RFR) classification method is described in the previous section. The parameters of the algorithm are: number of trees ($nTrees$) and minimum tree's leaf size ($minSize$).

Optimal parameters were selected on our dataset (Table 1), and then each of the classification algorithms with obtained hyperparameters was applied to Villani dataset (Table 2).

5 Conclusion

In this paper, the task of continuous user authentication using keystroke dynamics was considered. We have proposed a method to reduce dimensionality of a feature space based on two-sample Kolmogorov-Smirnov test and a quantile-based discretization technique to preprocess features with multimodal distribution. We have introduced two one-class classifiers, which haven't been applied to this problem before. We have conducted experiments on a benchmark dataset, which have confirmed that stability-based feature selection improves quality of authentication when using SVM, KNN, Fuzzy, KPCA and RNN classifiers, but almost no effect on RFR classifier. Furthermore, quantile-based discretization improves the results of all classifiers.

Acknowledgments. The research is financially supported by the Ministry of Education and Science of the Russian Federation (the subsidy agreement #14.604.21.0056, unique project identifier RFMEFI60414X0056).

References

1. Al Solami, E., Boyd, C., Clark, A., Ahmed, I.: User-representative feature selection for keystroke dynamics. In: 2011 5th International Conference on Network and System Security (NSS), pp. 229–233. IEEE (2011)

2. Alsultan, A., Warwick, K.: Keystroke dynamics authentication: a survey of free-text methods. Int. J. Comput. Sci. Issues **10**(4), 1–10 (2013)
3. Bailey, K.O., Okolica, J.S., Peterson, G.L.: User identification and authentication using multi-modal behavioral biometrics. Comput. Secur. **43**, 77–89 (2014)
4. Ceker, H., Upadhyaya, S.: Enhanced recognition of keystroke dynamics using Gaussian mixture models. In: Military Communications Conference, MILCOM 2015-2015 IEEE, pp. 1305–1310. IEEE (2015)
5. Chandrasekar, V., Akila, M., Maheswari, T.: Gravitional search optimization for the user authentication in biometrics. Middle-East J. Sci. Res. **23**(8), 1626–1631 (2015)
6. Everitt, R.A., McOwan, P.W.: Java-based internet biometric authentication system. IEEE Trans. Pattern Anal. Mach. Intell. **9**, 1166–1172 (2003)
7. Fawcett, T.: An introduction to ROC analysis. Pattern Recogn. Lett. **27**(8), 861–874 (2006)
8. Hastie, T., Tibshirani, R., Friedman, J., Franklin, J.: The elements of statistical learning: data mining, inference and prediction. Math. Intelligencer **27**(2), 83–85 (2005)
9. Hawkins, S., He, H., Williams, G., Baxter, R.: Outlier detection using replicator neural networks. In: Kambayashi, Y., Winiwarter, W., Arikawa, M. (eds.) DaWaK 2002. LNCS, vol. 2454, pp. 170–180. Springer, Heidelberg (2002). doi:10.1007/3-540-46145-0_17
10. Hoffmann, H.: Kernel PCA for novelty detection. Pattern Recogn. **40**(3), 863–874 (2007)
11. Kang, P., Cho, S.: Keystroke dynamics-based user authentication using long and free text strings from various input devices. Inf. Sci. **308**, 72–93 (2015)
12. Monaco, J.V., Bakelman, N., Cha, S.H., Tappert, C.C.: Developing a keystroke biometric system for continual authentication of computer users. In: 2012 European Intelligence and Security Informatics Conference (EISIC), pp. 210–216. IEEE (2012)
13. Namin, A.S.: Cyberspace security use keystroke dynamics. Ph.D. thesis, Texas Tech University (2015)
14. Petrovskiy, M.: A fuzzy kernel-based method for real-time network intrusion detection. In: Böhme, T., Heyer, G., Unger, H. (eds.) IICS 2003. LNCS, vol. 2877, pp. 189–200. Springer, Heidelberg (2003). doi:10.1007/978-3-540-39884-4_16
15. Tappert, C.C., Cha, S., Villani, M., Zack, R.S.: Keystroke biometric identification and authentication on long-text input. Int. J. Inf. Secur. Priv. (IJISP) **4**, 32–60 (2010)
16. Teh, P.S., Teoh, A.B.J., Yue, S.: A survey of keystroke dynamics biometrics. Sci. World J., 1–24 (2013). doi:10.1155/2013/408280