



Keystroke Dynamics Authentication Using Small Datasets

Nataasha Raul, Royston D'mello, and Mandar Bhalerao^(✉)

Sardar Patel Institute of Technology, Mumbai, India
`mandar.bhalerao@spit.ac.in`

Abstract. Keystroke dynamics is the analysis of timing information which describes exactly when each key was pressed or released by a user while typing. There have been various attempts to use this timing information to identify user's typing pattern and authenticate the user in a system, but mostly from a research perspective. We present a new methodology which focuses on solving the practical problems associated with deploying a keystroke dynamics authentication system. In our proposed methodology, a user's keystroke features are separated into two sets, namely high frequency and low frequency, based on the fraction of the total typing time the key takes. These two feature sets are then trained using OneClassSVM classifiers. Also, our proposed methodology requires minimal data and it is easily deployable on any computer.

Keywords: Keystroke dynamics · One class classification · Authentication

1 Introduction

Identification of a user as a credible user is a necessary condition, in order to allow access to any private or confidential information. Authentication works on three factors namely knowledge, possession and inherence where each factor gives partial information regarding validity of the user. Knowledge constitutes a token that a user knows prior to logging in (PIN, password, lock pattern). Possession includes something that a user has (mobile phone, card, license). Inherence includes what an individual has integrated in itself (Biometric). Biometrics is further classified into physiological characteristics and behavioural characteristics.

Success of biometric is determined by the factors such as universality, uniqueness, variance, measurability, performance, acceptability, circumvention [1].

Keystroke dynamics refers to the behavioural patterns found in a person's typing detected through their keystroke timestamps. Data needed to analyze keystroke dynamics is obtained by keystroke logging. These patterns are used to develop a unique biometric template of the user for future authentication [2]. These patterns are shown to possess cognitive qualities which are unique to each user and can be used as personal identifiers [3].

Keystroke dynamics can be used for both verification and identification. Verification refers to the process of proving a validity of claimed identity. Identification is classifying a user to be to be among those in a database or not. Identification is generally more time consuming, slower in responsiveness, and require higher processing capacity [4]. In our experiment, we have dealt with verification of user identity.

In this paper we try to solve some of the problems associated with keystroke dynamics based authentication from a practical point of view. Most of the current literature focuses on classification on medium to large size datasets. However such datasets won't be available in a production system. The publicly available datasets are created over long durations of time at regular intervals. However, practically we can't wait to accumulate that much data for the system to work. Hence, we aim to solve the following objectives in this paper:

1. Authentication in an unsupervised way.
2. Training the model to work with a dataset as small as 10 data points.
3. Decrease the false rejection rate of the system.

In this paper, we thus present a system for authentication of users in an unsupervised way. The user logs in 10 times during the onboarding process using which we create the user's authentication model which is used thereafter to authenticate the user.

2 Related Work

Keystroke dynamics can either be applied to free (dynamic) text or a fixed (static) text. In our work, we focus on static text. User recognition using keystroke dynamics can be perceived as a pattern recognition problem and most of the methods can be categorized as machine learning approaches (37%), statistical (61%) and others (2%) [4].

In statistics, the most popular method is using a distance measure. In distance measure, the pattern of the user logging in is calculated. This pattern is compared to a precomputed reference pattern in the database to check for any similarity. Zhong et al. [5] give a new distance metric exploiting the advantages of Mahalanobis and Manhattan distance metrics, namely robustness to outliers (for Manhattan distance) and to feature correlation and scale variation (Mahalanobis distance). They treat user authentication as a one class classification problem and use a Nearest Neighbour classifier with their new distance metric to classify a sample as originating from a genuine or a false user. Using this algorithm on the CMU dataset, they have obtained an EER of 8.7% and ZMFAR of 42.3% across all 51 subjects. The error rate is reduced to 8.4% for EER and 40.5% for ZMFAR if outliers are removed from training data.

Similarly, [6] present their distance metric which addresses the problem of popularly used distance metrics such as Euclidean, Manhattan; which assume that each feature of data point is equally important and independent from others. With their new metric, they obtained an EER of 0.062 compared to EER

of 0.110 obtained using Manhattan. Morales et al. [7] compares various score normalization techniques. Their work suggests that using score normalization techniques improves the overall performance of the system. The improvement, although, depends on the classifier and on dataset used.

Other than using distance measurement techniques, other statistical approaches include cluster analysis and probabilistic modelling.

Machine learning is widely used in the pattern recognition. The main idea is to identify a commonly occurring pattern and further classify data based on the pattern learned. This category includes various sub-domains like Neural Networks, Fuzzy Logic and Evolutionary Computing.

Among the listed sub-domains, neural network is claimed to give better results than statistical methods [8]. However, the classifiers require the genuine as well as the intruder's typing data to train. It is infeasible to specifically define and thus to obtain intruder's typing pattern, since any pattern other than the user's pattern is potentially an intruder's pattern. Furthermore, if the user's pattern changes, it would require the whole network to be retrained.

Another approach is by using anomaly detectors. Killourhy et al. [9] compared 14 anomaly detectors on a common dataset synthesized by them. Their experiments showed that Manhattan distance had the least EER of 0.096 and Nearest Neighbor (Mahalanobis) gave the least ZMFAR of 0.468.

Yu et al. [10] identified user authentication as a one-class classification problem, since getting the data of the imposter class is both infeasible and impractical. They used a GA-SVM based wrapper approach for feature selection and an SVM novelty detector, which fetched similar results as compared to neural networks. SVM maps the input vector into a high-dimensional feature space via the kernel function (e.g. linear, polynomial, sigmoid, or radial basis function). The algorithm will then search for a function that encapsulates the majority of patterns contained in the input vector and vector outside this region. As a result, the separating function is able to create more complex boundaries and to better determine which side of feature space a new pattern belongs.

3 Methodology

3.1 Data Collection

We have avoided the use of any public dataset which are usually created over predefined periods of time. We focus on creating our own dataset because our models need to work in an environment in which the user creates their dataset in one sitting during the registration process unlike the available public datasets. And the public datasets in which the users create the dataset in one sitting suffer from the short-time variance in a user's typing rhythm/speed. Using our preprocessing techniques described in the further sections, we solve this problem.

We used a Django web app for collecting keystroke data of each user. The keystroke values are recorded using Javascript. The password given is India@2018, which uses alphabets, numbers and special characters. Even though

in this system the password choice is unconstrained, we have set a password for our experimental setup to maintain a uniformity across the dataset. This ensures that the choice of passwords doesn't influence the results. Each user is required to type this password 10 times. From a system point of view, we can't expect a user's onboarding process to last more than 10 password entries. So in this paper we focus on making the system work for 10 password entries during the model creation phase.

The timestamps are recorded in milliseconds, up to five decimal places. Using the press and release times, various other features (described below) are calculated and stored in a Pandas dataframe, through which it is exported to a CSV file. Along with the timestamp values, the files also contains a column for username.

3.2 Feature Extraction

We record the press and release time of each key for each sample. Using these two timestamps, we calculate the following features:

- Holdtime (Dwell time) = (KeyPress - KeyRelease)
- Flight UU = KeyUp - KeyUp(Previous key released)
- Flight DD = KeyDown - KeyDown(Previous key pressed)
- Flight UD = KeyDown - KeyUp(Previous key released)

These features (except hold time, press and release times) are calculated for a set of two consecutive keys (digraph), for each sample.

3.3 Preprocessing

Consistency Checking. We focus on creating our own dataset because often times a user doesn't enter their password consistently. One time the speed can be fast and the very next time it can slow down. This is the kind of short-time variance that can occur in a user's typing rhythm.

Since we have less amount of training data, we cannot afford any outliers in it. Yu et al. [10] find that preprocessing the data and removal of outliers improves the overall performance of the classifier. Thus we need to ensure that the 10 samples we collect are representative of the user's typing pattern. Hence, while recording each training sample, we consider it only if it matches the first 5 samples. The first 5 entries are assumed to be authentic (not consisting of outliers) which serves as a base for further detection of outliers.

The matching is done using Nearest Neighbor algorithm. First, the Kth nearest distance of each previously recorded training example is calculated. Euclidean distance is used as a distance measure. For two n dimensional points x and y , Euclidean distance (d) between them is defined as $d = \sqrt{(x - y)^2}$. For our purposes, we have chosen $k = 3$. Concretely, what Kth (3rd) nearest distance means is that we consider the nearest distances of a training example with other examples. Each example will differ from others by a certain value (i.e. the distance

between them). Kth nearest distance is the distance between the example under consideration and the Kth most similar (Kth nearest) example to it. Thus, we will get n ($n \leq 10$) such distances. The average value of ' n ' distances recorded is chosen as a threshold value for incoming training example. Thus only if the Kth nearest distance of incoming training example is less than or equal to the threshold value, it is regarded as genuine training example. So the data is collected until there are 10 genuine training examples.

3.4 Separation of Keystrokes into High Frequency and Low Frequency

Instead of using all the features on one classifier, in this paper we have separated the feature set into two parts. Based on how close or further a keystroke is from its neighbours, in the time domain, we classify the keystrokes as high frequency and low frequency keystrokes respectively. And their features as high and low frequency features. For most users, the set of characters which they can type quickly will be very different. Hence we examine how separating the two kinds of features of a user can help in better identifying user patterns.

We consider the release timestamp only for all calculations of this module. We first normalise the dataset in the following way:

1. We first shift all the timings by the timing of the first key release. By doing this we make all the key release timings relative to the first one. This is to normalize the changes due to the user starting typing very late or early.
2. The relative timings are then converted into the fraction of the timing of the total duration of typing i.e. all the values are divided by the total time of typing which is found by the last release timestamp. This gives an indication of at what fraction of the typing time does that keystroke appear.

After this preprocessing the value of the first keystroke will always be 0 and that of the last keystroke will always be 1.

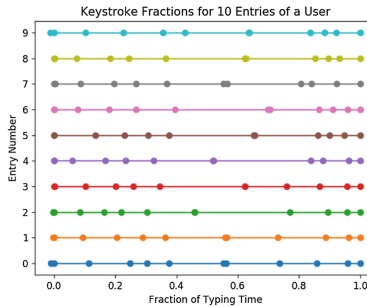


Fig. 1. Fractions at which a user's keystrokes occur for 10 entries

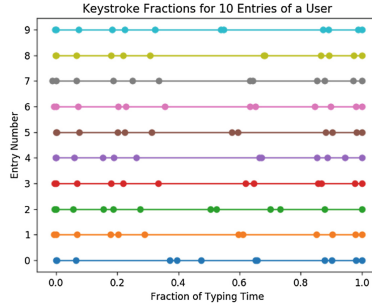


Fig. 2. The same plot for another user

From Figs. 1 and 2 it is evident that there are some regions that are distinct to the user. These regions may be low frequency regions or high frequency regions. There are clusters of low and high frequency regions in the plots.

The occurrence of these low and high frequency clusters may or may not be distinct to a user. In the graphs, most of the time bands look similar for both the users, except the time band between ‘a’ and ‘shift’, and ‘@’ and ‘2’. For these two time bands it can be seen that the second user has wider time bands for these set of keys. Hence they can be used to distinguish the two users visually and show that it is also possible to identify them mathematically.

We then create a threshold to decide high and low frequency as $1/n$. This gives the average timing difference fraction. For every keystroke pair the timing difference is calculated and if that difference is lower than the average timing difference it is considered as high frequency region, otherwise it is considered as a low frequency region.

3.5 Training

Once we have collected 10 samples for a user, the next time the user logs in, the training is initiated. The data in the saved csv files is preprocessed as described in section preprocessing. The features are split into two groups and two classifiers are trained. It should be noted that the preprocessing is applied only for the calculation of high and low frequency features. This preprocessing is not used for the data that goes into training, since this kind of normalisation generally does not affect results of a learning algorithm. We then store the accuracies of the individual classifiers along with the list of high and low frequency features to disk with the two models as well.

The training is done in an unsupervised fashion. To do supervised learning we would need training samples from the intruder. We could create a negative training class with a handful of different users acting as intruders, but since the intruder’s pattern could be anything we cannot accurately represent an intruder’s typing pattern with such a dataset. Hence we have to train the data in an unsupervised way.

The data that we have for each user, although explicitly unlabeled, does have an implicit label of belonging to a genuine user or the positive class. Therefore this kind of learning is a kind of semi-supervised learning. The algorithm that we use for the same is a One Class SVM.

The Support Vector Method For Novelty Detection by Schölkopf et al. [11] separates all the data points from the origin and maximizes the distance from current hyperplane to the origin. This results in a binary function which captures regions in the input space where the probability density of the data is high. Thus the function returns +1 in a region where there are training data points and 1 elsewhere.

3.6 Authentication

When a new entry comes in for a particular user, we first load all their data use the low and high frequency lists to separate the incoming features accordingly. These features are then passed to the respective models and we get their individual predictions. We then take a weighted average of the individual predictions with the stored accuracies as weights. If the final result is positive we authenticate the user, else the user is rejected.

4 Results

To evaluate the results, we collect data from 7 users and keep one of them as an intruder profile. To evaluate the comparative performance of our proposed methodology we evaluate two kinds of results. For the first set of results we use a single feature set and train the model using a single OneClassSVM classifier. For the second set of results we evaluate the proposed methodology by splitting the features and training two classifiers.

For every profile we use the user’s model and find prediction on the intruder’s profile and find FAR. Using each of the user’s profiles we find the FRR using all the incorrect predictions. We find the average FAR has decreased but the FRR has increased on splitting the feature set and creating weighted models. The results are shown in Table 1.

Table 1. Results

Type	FAR	FRR
Single classifier	0.3354	0.1200
High + low freq classifier	0.1032	0.2999

5 Conclusion

In this paper, we proposed a new authentication methodology using keystroke dynamics. Once deployed on a system, it would start authenticating users after 10 trials which are required for training purposes.

A new way of preprocessing keystroke timings was discussed, which separates the user's high frequency and low frequency sets. An FAR of 0.1032 and FRR of 0.2999 was obtained using this method.

In our approach, we ignore a change in user's typing pattern over a long time, which may invalidate the training data. Future efforts related to keystroke dynamics can be targeted to resolve this issue. Also, for the threshold used to define high and low frequency keystrokes we use the average time. This could be improved by trying multiple thresholds in combination with an optimisation algorithm to minimise FAR and FRR.

References

1. Jain, A.K., Bolle, R.M., Pankanti, S. (eds.): *Biometrics: Personal Identification in Networked Society*. Springer, New York (2006). <https://doi.org/10.1007/978-0-387-32659-7>
2. Deng, Y., Zhong, Y.: Keystroke dynamics user authentication based on gaussian mixture model and deep belief nets. <https://www.hindawi.com/journals/isrn/2013/565183/>
3. Obaidat, M.S.: A verification methodology for computer systems users. In: *Proceedings of the 1995 ACM Symposium on Applied Computing*, pp. 258–262. ACM, New York (1995)
4. Teh, P.S., Teoh, A.B.J., Yue, S.: A survey of keystroke dynamics biometrics. <https://www.hindawi.com/journals/tswj/2013/408280/>
5. Zhong, Y., Deng, Y., Jain, A.K.: Keystroke dynamics for user authentication. In: *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 117–123 (2012)
6. Ngoc, H.N., Nguyen, N.T.: An enhanced distance metric for keystroke dynamics classification. In: *2016 Eighth International Conference on Knowledge and Systems Engineering (KSE)*, pp. 285–290. IEEE, Hanoi (2016)
7. Morales, A., Luna-Garcia, E., Fierrez, J., Ortega-Garcia, J.: Score normalization for keystroke dynamics biometrics. In: *2015 International Carnahan Conference on Security Technology (ICCST)*, pp. 223–228 (2015)
8. Crawford, H.: Keystroke dynamics: characteristics and opportunities. In: *2010 Eighth International Conference on Privacy, Security and Trust*, pp. 205–212 (2010)
9. Killourhy, K.S., Maxion, R.A.: Comparing anomaly-detection algorithms for keystroke dynamics. In: *2009 IEEE/IFIP International Conference on Dependable Systems Networks*, pp. 125–134 (2009)
10. Yu, E., Cho, S.: Keystroke dynamics identity verification-its problems and practical solutions. *Comput. Secur.* **23**, 428–440 (2004)
11. Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. *Neural Comput.* **13**, 1443–1471 (2001)