Mining Keystroke Timing Pattern for User Authentication

Saket Maheshwary and Vikram Pudi

Centre for Data Engineering and Kohli Center on Intelligent Systems
International Institute of Information Technology-Hyderabad
Hyderabad, India
saket.maheshwary@research.iiit.ac.in, vikram@iiit.ac.in

Abstract. In this paper we investigate the problem of user authentication based on keystroke timing pattern. We propose a simple, robust and non parameterized nearest neighbor based regression algorithm for anomaly detection. Our approach successfully handle drawbacks like outlier detection and scale variation. Apart from using existing keystroke timing features from the dataset like dwell time and flight time, other features namely bigram time and inversion ratio time are engineered as well. The efficiency and effectiveness of our method is demonstrated through extensive comparisons with other methods using CMU keystroke dynamics benchmark dataset and has shown better results in terms of average equal error rate (EER) than other proposed techniques.

Keywords: Regression, Time series, Prediction, Security, Nearest neighbors, Pattern classification

1 Introduction

In this era where everyone wants secure, faster, reliable and easy to use means of communication, there are many instances where user information such as personal details and passwords get compromised thus posing a threat to system security. In order to tackle the challenges posed on the system security biometrics [4] prove to be a vital asset. Biometric systems are divided into two classes namely physiology based ones and the ones based on behavior. Physiology based approach allows authentication via use of retina, voice and fingerprint touch. In contrast, behavior based approach includes keystroke dynamics on keyboard or touch screens and mouse click patterns.

In this paper we propose an algorithm to deal with *keystroke dynamics* – a behavior based habitual rhythm which is used as a protective measure. Based on the analysis of the keystroke timing patterns, it is possible to differentiate between actual user and an intruder. By keystroke dynamics we refer to any feature related to the keys that a user presses such as key down time, key up time, flight time etc. In this paper, we concentrate on classifying users based on static text such as user password. The mechanism of keystroke dynamics can be integrated easily into existing computer systems as it does not require

any additional hardware like sensors thus making it a cost effective and user friendly technique for authenticating users with high accuracy. It is appropriate to use keystroke dynamics for user authentication as studies [7][9] have shown that users have unique typing patterns and style. Moreover, Syed et al. [7][9] has proven some interesting results in their research work as well. First hypothesis Syed et al. [7][9] proved is that the users present significantly dissimilar typing patterns. Second they have shown details about the relationship between users occurrence of sequence of events and their typing style and ability. Then they explained sequence of key up and key down events on the actual set of keys. They have also shown that there is no correlation between users typing skills and the sequence of events. Hence all these factors make it difficult for intruders to match with the actual users typing patterns. Keystroke dynamics is concerned with users timing details of typing data and hence various features could be generated from these timing patterns. In this paper we are using timing features only on static text.

The rest of the paper is organized as follows. In section 2 we discuss related work and our contribution. In section 3 we discuss the details of how features are engineered from the dataset and in section 4 we discuss the concept of optimal fitting line. In section 5 we present our proposed algorithm for keystroke timing pattern which is divided into two sub sections where first subsection discusses proposed approach and second subsection discusses the scaled Manhattan distance metric for anomaly detection. In section 6 we experimentally evaluate our algorithm and show the results. Finally, we conclude our study and identify future work in Section 7.

2 Related Work

Classifying users based on keystroke timing patterns has been in limelight when Forsen et al.[26] first investigated whether users could be distinguished by the way they type on keyboard. Researchers have been studying the user typing patterns and behavior for identification. Gaines et al. [27] investigated the possibility of using keystroke timings as to whether typists could be identified by analyzing keystroke times as they type long passages of text. Monrose and Rubin [11] later extracted keystroke features using the mean and variance of digraphs and trigraphs. Peacock et al. [28] conducted a detailed survey on the keystroke dynamics literature using the Euclidean distance metric with Bayesian like classifiers. Bergadano et al. [29] and later Gunetti et al. [30] proposed to use the relative order of duration times for different n-graphs to extract keystroke features that was found to be more robust to the intra-class variations than absolute timing. Some neural network based techniques have also been undertaken in the last few years. While the back-propagation models used yield favorable results on small databases, neural networks have a fundamental limitation that every time a new user comes into the database the network needs to be retrained. Gunetti and Picardi [30] published great results for text-free keystroke dynamics identification where they merge relative and absolute timing information on

features. Zhong et al. [5] proposed a new distance metric by combining Mahalanobis and Manhattan distance metrics. Many machine learning techniques have been proposed as well for keystroke dynamics as an authentication system. Keystroke dynamics can be applied with variety of machine learning algorithms like Decision Trees [22], Support Vector Machines [10], Neural Networks [23], Nearest Neighbor Algorithms [24] and Ensemble Algorithms [25] among others.

One problem faced by researchers working on these type of problems is that majority of the researchers are preparing their own dataset by collecting data via different techniques and the performance criteria is not uniform as well hence comparison on similar grounds among the proposed algorithms becomes a difficult task. To address this issue, keystroke dynamics benchmark dataset is publicly provided with performance values of popular keystroke dynamics algorithms [3] to provide a standard universal experimental platform. Killourhy et al. [3] collected and published a keystroke dynamics benchmark dataset containing 51 subjects with 400 keystroke timing patterns collected for each subject. Besides this they also evaluated fourteen available keystroke dynamics algorithms on this dataset, including Neural Networks [24], KNNs, Outlier Elimination [13], SVMs [10] etc. Various distance metrics including Euclidean distance [3], Manhattan distance [20] and Mahalanobis [3] distance were used. This keystroke timing pattern dataset along with the evaluation criteria and performance values stated provides a benchmark to compare the progresses of new proposed keystroke timing pattern algorithms on same grounds.

2.1 Our Contribution

The performance study of the fourteen existing keystroke dynamics algorithms implemented by Killourhy et al. [3] indicated that the top performers are the classifiers using scaled Manhattan distance and the nearest neighbor classifier. In this paper we present a new nearest neighbor regression based algorithm for anomaly detection that assigns weight to the feature vector. Finally we used scaled Manhattan distance metric for anomaly detection. Our algorithm has the following desirable features:

- **Parameterless:** We first design our nearest neighbor based regression algorithm and then show how the parameter can be automatically set, thereby resulting in a parameterless algorithm. This removes the burden from the user of having to set parameter values – a process that typically involves repeated trial-and-error for every application domain and dataset.
- **Accurate:** Our experimental study in Section 6 shows that our algorithm provides more accurate estimates than its competitors. We compare our approach with 14 other algorithm using the same evaluation criteria for objective comparison.
- **Robust/Outlier Resilient:** Another problem with the statistical approaches is outlier sensitivity. Outliers (extreme cases) can seriously bias the results by pulling or pushing the regression curve in a particular direction, leading to biased regression coefficients. Often, excluding just a single extreme

case can yield a completely different set. The output of our algorithm for a particular input record is dependent only on its nearest neighbors hence insensitive to far-away outliers.
- **Simple:** The design of our algorithm is simple, as it is based on the nearest neighbor regression. This makes it easy to implement, maintain, embed and modify as the situation demands.

Apart from our proposed algorithm we have engineered two new features namely *Bigram time* and *Inversion Ratio time* as discussed in Section 3.

3 Feature Engineering

What are good timing features that classifies a user correctly? This is still an open research problem. Though keystroke up, keystroke down and latency timing are the commonly used features, in this paper we have generated two new features from the given dataset besides the existing features. The dataset [3] provides three types of timing information namely the hold time, key down-key down time and key up-key down time. Besides these three existing features, two new features namely Bigram time and Inversion ratio time are engineered. Following are the details of five categories of timing features which is used to generate 51 features using keystroke timing dataset [3]. Figure 1 illustrates various timing features where up arrow indicates key press and down arrow indicates key release.

- **Hold Time** also known as dwell time, is the duration of time for which the key is held down i.e. the amount of time between pressing and releasing a single key. In figure 1, H_{i} represents the hold time. Hold time contributes to *eleven* features (where *ten* features are corresponding to the ten characters of static text and *one* feature corresponds to the return key).
- **Down-Down Time** key down key down time is the time from when key1 was pressed to when key2 was pressed. In figure 1, the times DD_{i} depicts the down down time. It contributes to *ten* features in our feature space.
- **Up-Down Time** key up key down time is the time from when key1 was released to when key2 was pressed. This time can be negative as well. In figure 1, the times UD_i depicts the up down time. It contributes to *ten* features in our feature space.
- **Bigram Time** is the combined time taken by two adjacent keystrokes i.e. the time from pressing down of key1 to releasing to key2. In figure 1, the times B_{i} depicts the bigram time. It contributes to *ten* features in our feature space.
- **Inversion Ratio Time** it is the timing ratio of hold time of key1 and key2 where key1 and key2 are the two continuous keystrokes. In figure 1, $H_{\mathrm{i}+1}/H_{\mathrm{i}}$ is the inversion ratio time. It contributes to *ten* features in our feature space.

Hence these five categories of timing features combines to give a *51* dimensional feature vector.

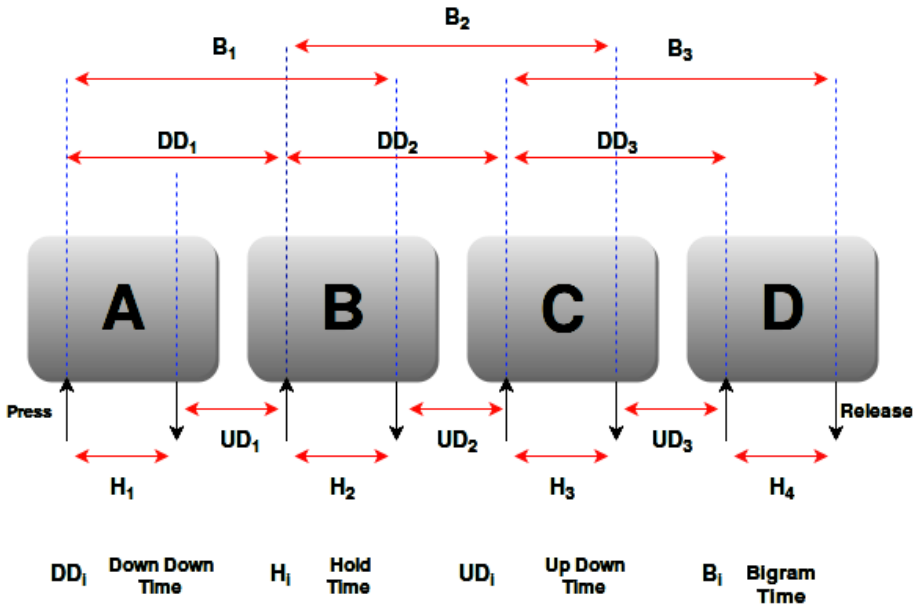

Fig. 1: Illustration of generated keystroke timing features where A,B,C,D are the keys

4 Optimal Fitting Line

Regression algorithms are used for predicting (time series data, forecasting), testing hypothesis, investigating relationship between variables etc. Here in this section we discuss how the optimal fitting line [2] attempts to predict the relationship between one variable from one or more other variables by fitting a linear equation to observed data.

Here in this paper we assume that to construct the line of best fit, with increase or decrease in each independent variable value the dependent variable changes smoothly. Thus this helps us in achieving almost linear relationship between dependent and independent variables thus allowing us to optimally fit a line onto the points in a small neighborhood. The line which minimizes the mean squared error is referred to as optimal fitting line. A low value of error indicates that the line is optimally fitted to the neighborhood and has captured the linearity of the locality. Let the k points have values $\{(x_1, y_1), ..., (x_k, y_k)\}$ in dimension x and y and let the variable to be predicted be y. Let the equation of line be of the form $y = ax + b$. Hence, dependent variable will take the value $ax_i + b$ corresponding to tuple i. Let the error in prediction for tuple i be denoted as e_i and is equal to $|y - ax_i - b|$. Hence the local mean error (LME) is denoted as,

$$LME(a, b) = \frac{\sum_{i=1}^{k} e_i}{k} = \frac{\sum_{i=1}^{k} (y - ax_i - b)^2}{k} \quad (1)$$

By minimizing LME where a and b are the parameters denoted by,

$$a = \frac{\sum_{j=1}^{k} y_{\mathrm{j}} \sum_{i=1}^{k} x_{\mathrm{i}} - k \sum_{i=1}^{k} x_{\mathrm{i}} y_{\mathrm{i}}}{\sum_{j=1}^{k} x_{\mathrm{j}} \sum_{i=1}^{k} x_{\mathrm{i}} - k \sum_{i=1}^{k} {x_{\mathrm{i}}}^2} \tag{2}$$

$$b = \frac{\sum_{i=1}^{k} y_{\mathrm{i}} - a \sum_{i=1}^{k} x_{\mathrm{i}}}{k} \tag{3}$$

Thus, we get the equation of the optimal fitting line. Now after constructing the line of best fit, we are able to predict the dependent values for test tuple. Then we compare the actual and the predicted values of dependent variable to calculate least mean error for the given test tuple. Now based on the mean error, we are assigning weights to our feature vector in inverse proportion which is discussed in section 5.

5 Our Proposed Approach

5.1 Proposed Algorithm

In this section we discuss our proposed nearest neighbor regression algorithm in detail. Our algorithm successfully eliminates nearest neighbor algorithm problems like choice of number of neighbors k by choosing the optimal k value corresponding to minimum error thus making our algorithm to be non parametric in nature. Our algorithm uses a unique weighing criteria (Algorithm 2) to assign weights to the feature vector hence enabling us to determine the relative importance of dimensions. The notation used for the algorithm is as follows: The training data has d dimensions with feature variables $(A_1, A_2, ..., A_{\mathrm{d}})$ and the value of the feature variable for the j^{th} feature variable A_{j} corresponding to the i^{th} tuple can be accessed as $data[i][j]$. The value of the dependent variable of the training tuple corresponding to id value i can be accessed as $y[i]$. The value of the dependent variable is calculated using the cosine similarity and k represents the number of nearest neighbors. For a given test tuple T the value of its k nearest neighbors is determined using an iterative procedure (line 4 of algorithm 1) hence making our algorithm to be non parametric in nature. The range for value k is from *low* to *high* where *low* is set to value 5 (sufficiently small value) an *high* is set to *size of training data data/2* (sufficiently large value). Now we describe our algorithm using the pseudo code below shown in Algorithm 1-2.

We iterate for k in range *5* to *size of training data set/2* and calculate the k nearest neighbors for test data. The k evaluated neighbors are stored in list *ClosestNeighbors* (line 6 of Algorithm 1). Now Algorithm 2 constructs an optimal fitting line $Line_i$ for each dimension of our feature vector (the dataset used by us has 51 features) by fitting a linear equation to observed *ClosestNeighbors* list, in the plane of feature variable and the dependent variable. The regression line is constructed as discussed in section 4. Using the parameters from the equation of the line a and b (equation 2 and 3) we predict the dependent value of test data (line 4 of Algorithm 2). Based on the predicted and actual values of the dependent variable squared error E_{i} is calculated (line 5 of Algorithm 2).

Algorithm 1

1: **procedure** KNN BASED DIMENSIONAL REGRESSION
2: $MinimumError \leftarrow \infty$, $ErrorforK \leftarrow \infty$
3: $OutputWeights \leftarrow 1$ // All d dimensions have same weight initially
4: **for** each k= low to high **do**
5: $ErrorforK \leftarrow 0$
6: $ClosestNeighbors \leftarrow GetNeighbors(data, k, T)$
7: $DimensionalRegressor(T)$ // Algorithm 2
8: **if** $MinimumError > Errorfork$ **then**
9: $MinimumError \leftarrow Errorfork$
10: $OutputWeight \leftarrow W_{\mathrm{T}}$
11: **end if**
12: **end for**
13: **return** $OutputWeight$
14: **end procedure**

Algorithm 2

1: **procedure** DIMENSIONAL REGRESSION
2: **for** each i = 1 to d **do** // d is the number of dimensions
3: $Line_{\mathrm{i}} \leftarrow ConstructLine(ClosestNeighbors, i)$ // As discussed in Secton 4
4: $PredictedTestVal_{\mathrm{i}} \leftarrow T_{\mathrm{i}} * a + b$
5: $E_{\mathrm{i}} \leftarrow (PredictedTestVal_{\mathrm{i}} - ActualTestVal_{\mathrm{i}})^2$
6: **end for**
7: **if** $\forall\ i$ E_{i} is equal **then**
8: $W_{\mathrm{T}} \leftarrow 1$
9: **else**
10: **for** each i = 1 to d **do**
11: $weight_{\mathrm{i}} \leftarrow max(E_{\mathrm{i}})/E_{\mathrm{i}}$
12: $W_{\mathrm{T}} \leftarrow weight_{\mathrm{i}}$
13: **end for**
14: **end if**
15: $Errorfork \leftarrow \sum_{j=1}^{d} E_{\mathrm{j}}$
16: **return** W_{T}
17: **end procedure**

It would be appropriate to state that a lower error value in predicting the line indicates that the constructed regression line is optimal in nature and fits the neighborhood of test data. Hence we conclude that the value of dependent variable predicted via the line of best fit is approximately correct and thus a higher weight should be assigned for a more optimal line or we can say a line with lower squared error. This intuition is captured by assigning weights in inverse proportion to the error in prediction for this dimension, hence a feature with high error value is assigned lower weight and and the feature with lower error value is assigned higher weight. The squared error in prediction of neighbors (line 15 of Algorithm 2) is computed and stored in $Errorfork$. A lower value

of the squared error indicates that the weight values chosen using the nearest neighbors are appropriate. We then select the value of the parameter k for which the calculated error is minimum and hence assigns the corresponding weight vector W_{T} (line 8-10 of algorithm 1). On this weighted feature vector we evaluate the anomaly score via a scaled Manhattan distance metric as discussed in the section below. The approach demonstrated in Algorithm 2 is a completely novel idea for dimension wise assigning weights in inverse proportion to error.

5.2 Scaled Manhattan Distance Metric

After the weights have been assigned to the feature vector via our proposed approach, we calculate the anomaly scores as described by Killourhy et al. [3] for evaluating our model. For calculating anomaly score we are using a scaled Manhattan distance metric as described by Araujo et al. [20]. The anomaly score is calculated as:

$$\frac{\sum_{i=1}^{n} |x_{\mathrm{i}} - y_{\mathrm{i}}|}{\alpha_{\mathrm{i}}} \tag{4}$$

where for i^{th} feature $\mathrm{x_i}$ is the test vector and $\mathrm{y_i}$ is the mean vector calculated from training phase and α_{i} is the mean absolute deviation from the training phase. From the equation 4 we can see that each dimension is scaled by a factor of α_{i} hence making our algorithm capable to handle scale variation.

6 Experimental Setup and Results

In this section we discuss the evaluation criteria used and the performance of our proposed model. We evaluated our approach on the CMU keystroke dynamics benchmark dataset [3]. *51* users are designated this task and average equal error rate(EER) is used as the performance measure for this dataset.

6.1 Evaluation Criteria

We frame keystroke dynamics based authentication as a one- class classification problem which learns a model for a user, rejects anomalies to the learned model as imposters, and accept inliers as the genuine user. Although the use of negative examples in training could significantly improve the accuracy of the classifier, it is unrealistic to assume prior knowledge about the keystroke features from imposters, let alone the availability of their training data. In order to ensure comparison on same grounds we have used exactly the same evaluation criteria as stated by Killourhy et al. [3] on our proposed approach. The data consist of keystroke timing information of 51 users, where each user is made to type a password 400 times. The ***.tie5Roanl*** password used for typing is a strong ten character static text. All the 51 users enrolled for this data collection task typed the same password in 8 different sessions with 50 repetitions per session thus making each user to type 400 times in total. Following are the four steps that are used to evaluate the algorithm for classification of a single (called as a

genuine user) from the other 50 users (called as the impostors). Repeating four steps for all 51 users is to make sure that each user have been attacked all 50 other users.

- **Model training** Extract 200 initial timing feature vectors for a genuine user from the dataset. Our proposed approach is used to build a model depicting the timing behavior of the users.
- **Testing genuine users** Extract last 200 passwords typed by the genuine user from the dataset. Now this is the testing phase for our proposed approach where these 200 timing feature vectors acts as test data. Scores generated in this step acts as the user scores.
- **Testing impostors** Extract initial 5 passwords typed by each of the 50 impostors (i.e., all subjects other than the genuine user) from the dataset. Based on our proposed algorithm and the model build in step 1 another set of scores are generated. Scores generated in this step acts as the impostor scores.
- **Assessing models performance** Based on the genuine user scores and impostor scores generated in the steps above, the ROC curve is generated for the actual (genuine) user. The equal error rate is calculated from the ROC curve where the equal error rate corresponds to that point on the curve where the false positive rate (false-alarm) and false negative rate (miss) are equal. Repeat the above four steps, in total of 51 times where every time each of the subsequent user is taken as the genuine user from the 51 distinct users in turn, and calculate the equal-error rate for each of the genuine users. Then calculate the mean of all 51 equal-error rates thus giving us the performance value for all users, and calculate the standard deviation which will give us the measure of its variance across subjects.

6.2 Results

Figure 2 shows ROC curve for different users with their Equal Error Rate (EER) value and user number where user number corresponds to the user as stated in CMU dataset[1]. Table 1 shows the comparison of various proposed keystroke timing algorithms with our proposed approach. Comparison is shown with 14 other algorithms which used the same dataset and the same evaluation criteria [3] thus assuring an objective comparison. Our proposed algorithm with and without two new engineered features (as discussed in Section 3) is able to achieve an average equal error rate (EER) of ***6.98%*** and ***7.39%*** respectively with a standard deviation (stddev) of 0.044 and 0.047 across 51 subjects. The average equal error rate (EER) shown in the table below are the fractional rates between 0.0 and 1.0, not the percentages. Clearly from Table 1, our proposed approach performs superior than other proposed techniques in comparison.

[1] Dataset available at `http://www.cs.cmu.edu/~keystroke/`

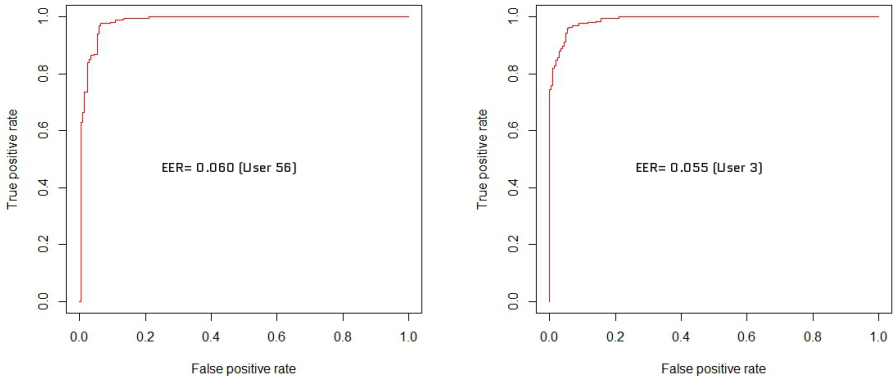

Fig. 2: Shows ROC curve for different users with their Equal Error Rate(EER) value where user number corresponds to the user as labeled in CMU dataset.

Table 1: Comparison of 16 different keystroke timing pattern algorithms that uses the same CMU keystroke timing dataset and evaluation criteria in terms of Average Equal Error Rate(EER)(with standard deviation shown in brackets).

Model/Algorithm	**Average EER(stddev)**	**Source**
Our Proposed Algorithm (with 2 new engineered features)	**0.0698(0.044)**	
Our Proposed Algorithm (without 2 new engineered features)	**0.0739(0.047)**	
Median Vector Proximity	0.080(0.055)	Al-Jarrah [1]
Manhattan-Mahalanobis(No Outlier)	0.084(0.056)	Zhong et al. [5]
Manhattan-Mahalanobis(Outlier)	0.087(0.060)	Zhong et al. [5]
Manhattan(scaled)	0.0962(0.0694)	Killourhy et al. [3]
Nearest Neighbor(Mahalanobis)	0.0996(0.0642)	Killourhy et al. [3]
Outlier Count(z-score)	0.1022(0.0767)	Killourhy et al. [3]
SVM (one-class)	0.1025 (0.0650)	Killourhy et al. [3]
Mahalanobis	0.1101 (0.0645)	Killourhy et al. [3]
Manhattan (Filter)	0.1360 (0.0828)	Killourhy et al. [3]
Neural Network(Auto-assoc)	0.1614 (0.0797)	Killourhy et al. [3]
Euclidean	0.1706 (0.0952)	Killourhy et al. [3]
Fuzzy Logic	0.2213 (0.1051)	Killourhy et al. [3]
k Means	0.3722 (0.1391)	Killourhy et al. [3]
Neural Network(Standard)	0.8283 (0.1483)	Killourhy et al. [3]

7 Conclusion and Future Work

In this paper we investigate the problem of authenticating users based on keystroke timing pattern. We engineered new features namely bigram time and inversion ratio time apart from the features already given in the CMU keystroke timing

dataset. Besides engineering new features we also proposed a simple and robust nearest neighbor based regression algorithm. We evaluated our results and compared it against 14 other algorithms that used the same dataset and evaluation criteria thus providing performance comparison on equal grounds. Although simple, it proved to be effective as it outperformed majority of competing algorithms as shown in Table 1. Future work involves extending our work for soft keys or touch pad keys and in addition to timing pattern features we can use users pressure patterns as well in order to authenticate users. We are planning to experiment with different curve fitting techniques as well.

8 Acknowledgement

This work was supported by *http://metabolomics.iiit.ac.in/* and we would like to thank them for their support.

References

1. Mudhafar, M., Al-Jarrah, An Anomaly Detector for Keystroke Dynamics Based on Medians Vector Proximity. Journal Of Emerging Trends In Computing And Information Sciences VOL3, 2012. 6.
2. A.Desai, H.Singh and V.Pudi , "Gear: Generic,efficient,accurate kNN-based regression",Proc. Int. Conf. KDIR,pp.1 -13
3. Kevin S. Killourhy and Roy A. Maxion. "Comparing Anomaly Detectors for Keystroke Dynamics," in Proceedings of the 39th Annual International Conference on Dependable Systems and Networks (DSN-2009), pages 125-134, Estoril, Lisbon, Portugal, June 29-July 2, 2009. IEEE Computer Society Press,Los Alamitos,California,2009.
4. R. Giot, B. Hemery and C. Rosenberger, "Low Cost and Usable Multimodal Biometric System Based on Keystroke Dynamics and 2D Face Recognition", Int'l Conf. on Pattern Recognition (ICPR), pp. 1128 -1131, 2010.
5. Y.Zhong,Y. Deng,and A. K. Jain, "Keystroke dynamics for user authentication, in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW '12), pp. 1171123, Providence, RI, USA, 2012.
6. Jiacang Ho and Dae-Ki Kang, "Sequence Alignment with Dynamic Divisor Generation for Keystroke Dynamics Based User Authentication, Journal of Sensors, vol. 2015, Article ID 935986, 14 pages, 2015. doi:10.1155/2015/935986
7. Z. Syed, S. Banerjee, Q. Cheng, and B. Cukic, "Effects of User Habituation in Keystroke Dynamics on Password Security Policy," IEEE 13th International Symposium on High-Assurance Systems Engineering, pp. 352-359, 2011
8. S. P. Banerjee and D. Woodard, "Biometric Authentication and Identification Using Keystroke Dynamics: A Survey," Journal of Pattern Recognition Research, pp. 116-139, 2012.
9. Z. Syed, S. Banerjee, and B. Cukic, "Leveraging variations in event sequences in keystroke-dynamics authentication systems, in Proceedings of the IEEE 15th International Symposium on High-Assurance Systems Engineering (HASE '14), pp. 916, IEEE, January 2014.

10. Chu, W., Keerthi, S.S.: New approaches to support vector ordinal regression. In:ICML (2005)
11. A. Messerman, T. Mustafic, S. Camtepe and S. Albayrak, Continuous and non-intrusive identity verification in real- time environments based on free-text keystroke dynamics, Intl Joint Conf. on Biometrics (IJCB), 2011.
12. K. Revett, "A bioinformatics based approach to user authentication via keystroke dynamics, International Journal of Control, Automation and Systems, vol. 7, no. 1, pp. 715, 2009.
13. S. Haider, A. Abbas, and A. K. Zaidi. "A multi-technique approach for user identification through keystroke dynamics, IEEE Intl Conf. on Systems, Man and Cybernetics, pp. 13361341, 2000.
14. Hilbe, Joseph, M.: Logistic Regression Models. Chapman and Hall/CRC Press (2009)
15. I.H. Witten and E. Frank, Data Mining: Practical machine learning tools and techniques, Morgan Kaufmann, 2 edition, 2005.
16. Hall, M., Ian, H.: The Weka Data Mining Software: An update. SIGKDD Explorations (2009)
17. E. Yu and S. Cho. "GA-SVM wrapper approach for feature subset selection in keystroke dynamics identity verification, In Proc. Intl Joint Conf. on Neural Networks (IJCNN), pp. 22532257, 2003.
18. R. Joyce and G. Gupta. "Identity authentication based on keystroke latencies,Communications of the ACM, 33(2):168176, 1990.
19. D. Gunetti and C. Picardi. "Keystroke analysis of free text, ACM Transactions on Information and System Security, 8(3):312347, 2005.
20. L. C. F. Ara ujo, L. H. R. Sucupira, M. G. Liz arraga, L. L. Ling, and J. B. T. Yabu-uti. User authentication through typing biometrics features. In Proceedings of the 1st In- ternational Conference on Biometric Authentication (ICBA), volume 3071 of Lecture Notes in Computer Science, pages 694700. Springer-Verlag, Berlin, 2004.
21. F. Monrose, A.D. Rubin / Future Generation Computer Systems 16 (2000) 351359
22. Brieman, L., Friedman, J., Olshen, R., Stone, C.: Classfication and Regression Trees. Wadsworth Inc. (1984)
23. Haykin, S.: Neural Networks-A comprehensive foundation. Prentice-Hall (1999)
24. Cover, T.M., Hart, P.E.: Nearest Neighbor Pattern Classfication. IEEE Transactions on Information Theory (1967)
25. Schapire, R.E.: A Brief Introduction to Boosting. In: ICJAI (1999)
26. G. Forsen, M. Nelson, and R. Staron, Jr. Personal attributes authentication techniques.Technical Report RADC-TR-77-333,Rome Air Development Center, October 1977.
27. R. S. Gaines, W. Lisowski, S. J. Press, and N. Shapiro. Authentication by keystroke timing: Some preliminary results. Technical Report R-2526-NSF, RAND Corporation,May 1980.
28. A. Peacock, X. Ke, and M. Wilkerson. Typing patterns: A key to user identification. IEEE Security and Privacy,2(5):4047, 2004.
29. F. Bergadano, D. Gunetti, and C. Picardi, "User Authentication through Keystroke Dynamics, ACM Trans. Information and System Security, 5(4), pp. 367397, 2002.
30. D. Gunetti and C. Picardi. "Keystroke analysis of free text, ACM Transactions on Information and System Security, 8(3):312347, 2005