User Authentication with Keystroke Dynamics in Long-Text Data

Hayreddin Çeker and Shambhu Upadhyaya
University at Buffalo
Buffalo NY 14260, USA
hayreddi@buffalo.edu, shambhu@buffalo.edu

Abstract

Keystroke dynamics is a form of behavioral biometrics that can be used for continuous authentication of users while working at a terminal. In this paper, we extend the use of support vector machine (SVM) for continuous authentication with long-text data, from one-time password based authentication using short text. In result, we show we can authenticate legitimate users and reject impostors with negligible error (close to 0% equal error rate) by setting a one-class SVM for each user using a dataset of 34 users in a controlled environment. Our results show that by standardizing the input and setting the correct kernel scale, one-class SVM can be utilized as a tool to continuously authenticate users, and recognize keystroke dynamics with a high accuracy.

1. Introduction

Conventional authentication mechanisms use a single point of entry for users to log in with a credential in an easily usable way. However, when the session remains active, typical systems fail to verify that the user originally authenticated is the user still at the terminal. The trade-off between security and usability can be addressed by utilizing active authentication based on behavioral biometrics. The active authentication paradigm suggests that a system should be able to recognize and continuously verify the identity of a user [8].

Keystroke dynamics is one of the efficient and inexpensive behavioral biometrics that can be used to authenticate computer users in the background while the user is actively working at the terminal. Typing characteristics have been shown to be distinctive enough to distinguish a computer user from another because of the unique timing of keystrokes that each individual performs during typing. Among the machine learning techniques proposed, support vector machine (SVM) has been only applied on short text [19], namely, password. However, authentication mechanisms using short text can be insufficient in practice, especially those that require all users to type the same passphrase during training for consistent feature extraction. Therefore, a mechanism to process long-text data is required to realize the active authentication paradigm without prompting users with a password screen.

In our experiments, the users transcribe a text in a controlled environment and answer survey questions freely. We propose a new methodology to process the long text and extract features to be used for long-term continuous authentication. The contributions of this paper include processing of long text using SVM and a new feature extraction method for keystroke dynamics. Also, we show that SVM can be utilized to differentiate users with high accuracy and low performance overhead.

The paper is organized as follows. Section 2 provides the related works on the use of SVM methods in keystroke dynamics. Then, the details on data collection and feature extraction are given, and the experiments conducted are described in Section 3. Section 4 presents the results of using SVM with long-text data and its effectiveness in recognizing users over short-text data. Finally Section 5 summarizes our findings and gives an insight into how our technique could be improved further.

2. Related Work

Over the years, many different methods and classifiers have been proposed in the area of keystroke dynamics to distinguish an individual from another. In security, typing pattern of a user is a behavioral biometrics that can be used as a means of authentication. In this section, basically the research on keystroke dynamics involving support vector machine (SVM) is discussed. Also, the way the SVM is integrated in the experiments is explored.

Yu et al. [21] consider the problem of authenticating legitimate users while rejecting impostors as an outlier detection problem that can be solved by using one-class SVM. This way, a hypersphere can be drawn to encompass data points from legitimate users, and disallow any other points that fall off the sphere. In addition, they employ a genetic algorithm to select a subset of features that yields the best results by randomized searching. As a result of the exper-

iments, they report the average false reject rate (FRR) of 21 users when the classifier makes no false positive (FAR reaches 0% error). The experiments in [1, 18] are conducted by using a similar method with faster feature selection processes that enable a more diverse set of features.

Saggio et al. [15] employ a two-class SVM with linear kernel using dwell time (time between press and release of the same key) and flight time (time between key presses) of digraphs. They create a separate SVM with one-vs-all method for each user by accepting the legitimate data as positive, and all other data as negative. The users are enrolled in the experiments typing short passwords, though one-vs-all method can be impractical for large training datasets and long-text data. Sang et al. [16] uses a numeric keypad in the experiments and allow users to select numerical passwords to record user's keystroke dynamics. In addition to one-class SVM, they also employ one-vs-all method by categorizing all impostors data as the negative class. The results show that one-class SVM outperforms two-class SVM, and has a closer ROC curve to the top corner of the error chart. Similarly, the experiments of Paula et al. [14] support the fact that the typing pattern of a user can be better recognized by using one-vs-one (one-class) SVM than using one-vs-all SVM.

Ngugi et al. [13] investigate the changes in PIN entry on a numeric keypad by analyzing how typing patterns behave over time. After filtering out the outliers, the features are scaled and fed to an SVM with radial basis function (RBF). Giot and Rosenberger [6] also use RBF kernel to train keystrokes for gender recognition. They use the publicly available GREYC short-text dataset. The results are reported by using a cross-validation process in which the dataset is split into partitions, and one partition is tested against the others in an order.

The experiments conducted in the aforementioned studies are for verification purposes using only short texts (e.g., name, password). They are used as a secondary/supplementary authentication mechanism rather than as a continuous user monitoring and transparent authentication system. The features are extracted when a user enters the password several times in the training session. The next time the user tries to login, the extracted features are compared and the user is authenticated if enough similarity is found. However, in our experiments each user has long text data (13,461 keystroke records on average) to train the system.

The only experiment with long-text data is performed by Garg et al. [5] as a proof-of-concept using GUI-based activities and mouse movements for 3 users. Two-class SVM is used by combining over 100 features composed of mouse angles, key press/hold time, background processes, etc. The results show that up to 96% detection rate is provided based on various feature combinations. Our work is different in terms of feature extraction and prediction methods, and requires the analysis of only certain digraphs efficiently without additional modalities. Also, we use a much larger dataset in this paper, which will help generalize the results.

In Section 5, we provide a table of summary including our current work that lists reported results and important details about these experiments. In addition, the comparison study by Killourhy and Maxion [10] is a valuable resource to find out how classifiers perform under the same conditions using the same password and feature sets.

3. Methodology

3.1. Data Collection

In the data collection process, a desktop environment is set up to record the keystroke data in a lab at Clarkson University. Thirty nine subjects from the university employees and students are enrolled in two different sessions within a period of 11 months. Each session takes approximately 1 hour on two separate days. The users who didn't take the second session are removed. At the end, 34 users are left for cross-validation.

The first session includes a set of survey questions that the subjects are asked to answer. The survey is carefully designed so that the subjects can respond to questions without long pauses and hesitations. To involve more natural typing effect in the experiment, some questions require subjects to choose their own writing topics. Also, the participants describe a picture of a crowded scene with various human activities. The second session consists of a static long-text typing process in which Steve Jobs' famous commencement speech at Stanford University is required to be transcribed by the subjects.

The key-logger is a browser based Java Script program that collects the character and key's press and release time in millisecond. It records the timing data in real time and transfers them to a PHP web server. The program enforces subjects to type at least 500 characters to answer the questions. In this dataset, the digraphs whose latency is above 200 ms are ignored. Also, if a frequency of occurrence of a digraph in the text is less than 50, we exclude it. At the end, we obtain 13,461 keystrokes on average in Table 1 for both sessions after the filtering. This data set is publicly available for research by contacting the authors at Clarkson University [2, 20] and signing a non-disclosure agreement (NDA).

Statistics	Number of keys
Average	13,461
Standard deviation	2,775
Min	6,597
Max	17,271

Table 1: Keystroke Statistics in the Dataset

3.2. Feature Selection

Most of the papers referred in Section 2 feed SVM classifier with the features extracted from inter-key latencies. Since all of these studies involve short text, alignment of the features is not an issue for different users or in different sessions of the same user as the dimension of the feature space is always the same. In other words, the users type the same password in the experiments; hence comparison is easy and simple to operate without any extra processing. However, in long text data which include both transcribed and free text, the feature extraction process is not an easy task since a session may include various sizes of keystrokes with repeated digraphs. Therefore, we propose a new feature alignment method for long-text keystroke dynamics.

	$digraph_1$			$digraph_2$			$\cdots$	$digraph_d$		
$user_i$	f_1^1	f_2^1	f_3^1				$\cdots$			
	.	.	.				$\cdots$			
	f_1^k	f_2^k	f_3^k				$\cdots$			
				f_1^1	f_2^1	f_3^1	$\cdots$			
				.	.	.	$\cdots$			
				f_1^t	f_2^t	f_3^t	$\cdots$			
							$\cdots$			
							$\cdots$	f_1^1	f_2^1	f_3^1
							$\cdots$	.	.	.
							$\cdots$	f_1^n	f_2^n	f_3^n

$m \times d$

Table 2: Feature matrix of long-text data

In this new approach, we calculate the most commonly-typed digraphs by processing the entire text in advance. We run our experiments by including various number of digraphs (d) from the list. Thus, the size of feature vector depends on the value, d, accordingly. We treat each digraph as a separate record that only includes information about the respective keys (f values under $digraph_1, digraph_2, \cdots, digraph_d$); all other fields are set to 0 (blank cells). Suppose we extract flight time (f_1^*) and dwell times (f_2^* and f_3^*) of a particular digraph that is selected from the list of size d. We initialize a two dimensional feature matrix that has $m \times d$ number of columns, where m is the number of digraph features included ($m = 3$ in this example for flight time of the digraph and dwell times of the individual keys).

During the processing of the long-text data, each time we come across the particular digraph, we create a new row and calculate the flight and dwell times accordingly. We set the first 3 columns with these values and fill the rest of the row with 0s. This way, we can include all instances of this digraph in separate rows. For instance, $digraph_1$ has k rows in Table 2 in which only the first 3 columns are filled with digraph timing information, and the rest is set as 0. Afterwards, we continue with the next digraph in the most commonly-typed digraph list, and apply the same procedure by moving the column cursor by m so that data from different digraphs do not align under the same column. Note that the values for $digraph_2$ shifts by 3 columns and contain t instances.

Finally, we obtain a training set which includes as many records as the number of digraphs from the list within the long text. This way, we can integrate all data points that belong to the most commonly-used digraphs which have been used as a good identifier of an individual [11]. This process is repeated for each user separately. Then every user's data is partitioned into training (80%) and testing (20%) groups. A one-class SVM is trained for each user separately (line 3 in Alg. 1) using only the digraph information that belong to the particular user since it is not required to provide negative data points in one-class SVM. Once all users' data are processed, we obtain 34 separate SVM classifiers.

Algorithm 1 One-class SVM prediction algorithm

Input: The most common D digraphs $L = \{l_1, ..., l_d\}$
Input: Kernel scale s
Output: Prediction scores of N users with cross-validation $R = \{r_{1,1}, r_{1,2}, ..., r_{n,n}\}$
1: Extract features from L for all users and partition into training L_{tr} and test L_{test} dataset
2: **for** i:=1 **to** N **do**
3: Fit $L_{tr}(i)$ to one-class SVM by scaling with s: $OCSVM_i := fit(L_{tr}(i), s)$
4: **for** j:=1 **to** N **do**
5: $R(i, j) := predict(OCSVM_i, L_{test}(j))$
6: **end for**
7: **end for**

Then, an overall score is calculated as to how likely a particular user's test dataset belongs to the user whose SVM classifier is used for prediction. That is, all users are cross-checked with each other to find out how one-class SVM is successful in drawing hyperplanes that separate a user from another. The overall score is simply calculated by summing up the prediction scores given by the SVM for the testing data points. At the end, every classifier contains an overall score for a user (line 5 in Alg. 1). Our expectation is that all SVMs have their user's overall score as the highest among others. This way, we can deduce that one-class SVM can transform standardized keystroke data points to another dimension where the users are obviously distinguishable. We use MATLAB's SVM package as software, and train the system with the dataset by setting the proper parameters described in the training process.

3.3. Training Process

Normalization in SVM can have an impact on accuracy of a system, too. Input values are scaled to have a unit norm and lie between 0 and 1. Graf et al. [7] explore the normalization in input space, feature space and in both spaces, and conclude that feature space normalization with adjusted kernel parameter b results in the lowest error rate. However, the normalization experiment conducted in [7] is useful for linear and polynomial kernel functions. It doesn't have an impact on Gaussian kernels since the division operation by symmetric values in RBF kernels always yields 1 ($K(x, x) = 1$) which doesn't affect the result. Therefore, we omit the normalization process in this study.

In a practical guide for SVM, it is suggested that scaling before applying SVM can yield better results. The main advantage of scaling is to avoid features in greater numeric ranges dominating those in smaller numeric ranges [9]. In Gaussian RBF kernel, scaling the input vectors corresponds to scaling variance (σ). That is, when we scale inputs in Eq. 1 by a factor s, we obtain the scaled kernel as in Eq. 2. Scaling at a feature-space level is also applied by some researchers [3] but since we standardize the input and for the sake of simplicity, our scaling level is kept only at the input-space. In this paper, we iterate over various kernel scale parameters to find out the optimal one.

$$k(x, y) = exp\left(-\frac{||x - y||^2}{2\sigma^2}\right) \quad (1)$$

$$k_s(x, y) = exp\left(-\frac{||x/s - y/s||^2}{2\sigma^2}\right) = exp\left(-\frac{||x - y||^2}{2(\sigma * s)^2}\right) \quad (2)$$

4. Results

The test results are based on two separate long-text sessions from 34 users. Each user is compared against the other, and the SVM prediction score is calculated. Leggett and Williams [11] run experiments with different sets of digraphs, e.g., the most frequent, left hand or right hand digraphs. In this paper, we utilize a similar procedure to find out the set that yields the highest accuracy.

We first begin with the most common 4 digraphs (he, re, th, an) using the flight time and dwell time features. The digraph sets in this study are formed statistically using the text typed by the users. The most common ones are iteratively selected and partitioned into sets as follows:

4 digraphs: hrta/eehn
6 digraphs: hrtate/eehnor
8 digraphs: hnhrtate/adeehnor
12 digraphs: hnhrvtlaiteo/adeeehlnnorr
14 digraphs: hnhrvntalaiteo/adeeeghllnnorr
16 digraphs: hnhrvntalaeiteoo/adeeeghllnnnorru

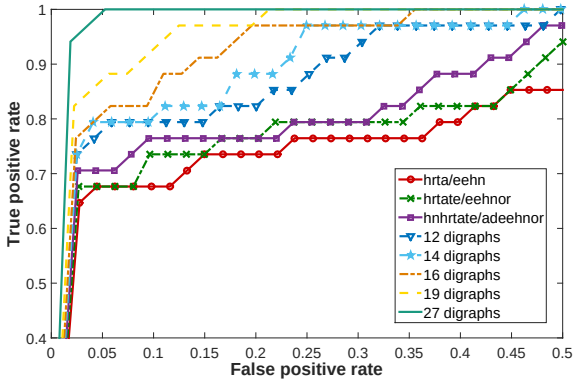

Figure 1: ROC by the most common digraphs

19 digraphs: hnhrtvntalaeiteoiao/adeeeeghllnnnorrstu
27 digraphs: hlnhlmrtvnttalaeioctaeoiaio/aadeeeeeeghillnnnnoorrrsttu

The most common digraphs are displayed in a way that the first letter is on the left side and the second letter is on the right side of the backslash (/), e.g., the most common 6 digraphs are *he, re, th, an, to, er*. Fig. 1 shows how receiver operating characteristics (ROC) change by the number of most common digraphs. ROC is a heavily used statistics in machine learning to illustrate the performance of a classifier as its discrimination threshold changes [17]. The area under the curve (AUC) describes the likelihood that a randomly chosen legitimate data point is classified as positive, is higher than that of a randomly chosen negative example. More information about the ROC analysis can be found in [4]. In short, high AUC value represents a good differentiation capability of a classifier.

Digraph set	Kernel Scale	Training Time	Testing Time	AUC	EER %
hrta/eehn	0.31	0.12	0.0077	0.9947	2.94
hrtate/eehnor	0.36	0.20	0.0128	0.9973	2.58
hnhrtate/adeehnor	0.46	0.28	0.0169	0.9979	2.94
12 digraphs	0.54	0.55	0.0275	1	0
14 digraphs	0.56	0.67	0.0379	1	0
16 digraphs	0.65	0.84	0.0448	1	0

Table 3: Computational cost of the digraph sets

Accordingly, we can observe in Fig. 1 that as we include more digraphs, the trendline gets closer to the corner and AUC approaches to 1, which is the ideal condition. However, the drawback of increasing the number of digraphs is that it has a negative effect on performance because the number of features increments by a factor of 3 as we add another digraph (see Feature Selection in Section 3). For instance, training with 19 digraphs for all users takes 23.9 seconds yielding 0.979 AUC; while 27 digraphs takes 54.5 seconds yielding 0.996 AUC with an only 1.7% improvement.

To overcome the performance degradation caused by the

high number of digraphs, we rather optimize the kernel scale described in Eq. 2. Our experiments show that we can achieve better results with less number of digraphs by setting a proper and fine-tuned kernel scale value. Therefore, we iterate over numerous kernel scale values using different number of digraphs plotted in Fig. 1. In result, we surprisingly find out that each digraph set has its own optimal kernel scale, and achieves perfect accuracy with multiple digraph set and kernel scale. In Fig. 2, we plot the AUC with respect to various kernel sizes. From the figure, we conclude that as we include more digraphs to differentiate users from keystroke dynamics, the optimal kernel scale shifts forward accordingly. The reason being is that as more digraphs are included, more features are extracted, and thus higher variance occurs among the values. Therefore, it is consistent to reach the perfect accuracy (AUC = 1) with higher kernel scales as we increase the number of digraphs.

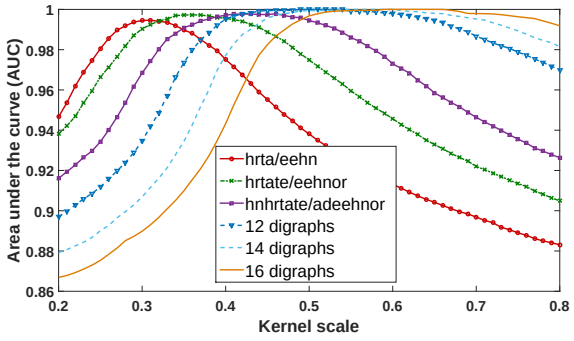

Figure 2: AUC by kernel size

Furthermore, we observe that there is not a single precise value resulting in an AUC value of 1 in the experiments which we run with high number of digraphs. Instead, there is a range of kernel scale values that yields 1.0 AUC. The intervals for 12, 14 and 16 digraphs are [0.49 – 0.54], [0.54 – 0.6] and [0.59 – 0.66], respectively. We interpret this result as follows: A high number of digraphs, and consequently more features in training stage add tolerance to the recognition process. Though it takes more time to generate the separating hypersphere of the legitimate data, and shrink it to a smaller space, at that level, it becomes more achievable to scale and play around with the variance of the surrounding sphere.

Now that we have multiple pairs that have 100% accuracy, it is desirable to have a table that exhibits the computational costs of the experiments for an arbitrary user. In Table 3, we show the time elapsed in the training and testing stages of the digraph sets along with their kernel scales which give the highest AUC value. We run each experiment with the same parameters 100 times and take the average of the resulting durations on a MacBook Pro with 2.5 GHz Intel Core i5 processor and 8GB RAM. For the kernel scale values that yield the same AUC, we pick the one with shortest training time. In result, we see that use of 12 most common digraphs is sufficient to be able to fully recognize 34 users in our experiments with almost no classification error. Also, the testing time is promising and advisable for a continuous authentication mechanism that runs in the background.

Study	FAR%	FRR%	EER%	SVM Type	Text Length	# of users
Yu et al. [21]	0	3.54	-	One-class	Short	21
Sung et al. [18]	3.85	13.10	-	One-class	Short	75
Sang et al. [16]	-	-	20/60	One/Two-class	Short	10
Paula et al. [14]	9/29	7/29	-	One/Two-class	Short	6
Azevedo et al. [1]	9/29	7/29	-	One/Two-class	Short	24
Martono et al. [12]	0.95-14.7	5.7	-	One-class	Short	5
Killourhy et al. [10]	-	-	10.2	One-class	Short	51
Ngugi et al. [13]	-	-	2	Two-class	Short	12
Saggio et al. [15]	4.93	5.10	-	Two-class	Short	16
Giot et al. [6]	-	-	8.45	Two-class	Short	8
Current work	-	-	**0.0-2.94**	**One-class**	**Long**	**34**
Garg et al. [5]	96% detection rate			Two-class	Long	3

Table 4: Comparison of Error Rates

We provide a summary of results in Table 4 including our current research and related works in the literature. The EER in our study varies from 2.94% down to 0%, when 4 and 12 (or more) number of digraphs are used, respectively. Note that $AUC = 1$ in ROC curves indicates 0% EER. Our results outperform the only long-text study in the literature, as well as most of the short-text experiments under different conditions.

5. Conclusion

Active authentication paradigm suggests that a system should be able to recognize and continuously verify identity of a user. For an active authentication mechanism, a system should authenticate a user unobtrusively and transparently in a continuous manner. Support vector machine (SVM) is a great tool to meet these criteria to analyze and classify users continuously in the background.

In addition, keystroke dynamics is one of the efficient and inexpensive behavioral biometrics that can be used to authenticate computer users in the background while the user is actively working at the terminal. In this paper, we show that SVM can be utilized as the classification engine of active authentication mechanism due to its high recognition rate and efficient processing. By using flight and dwell times of 12 most common digraphs, we are able to distinguish all 34 users enrolled in the experiments with an appropriate scale of RBF kernel in one-class SVM. Also, our study demonstrates how kernel scale shifts forward as we include more digraphs along with the computational overhead brought by the new features. As a future work, we plan to run experiments under various space and time conditions to test the robustness of our algorithm. Also, design of an adaptive SVM that can determine the optimal kernel scale value, is in our future plans.

Acknowledgments

This research is supported in part by National Science Foundation Grant No. CNS: 1314803. Usual disclaimers apply. The authors like to thank the Clarkson University research team led by Stephanie Schuckers for providing the dataset for our experiments.

References

[1] G. Azevedo, G. Cavalcanti, and C. Filho. An approach to feature selection for keystroke dynamics systems based on PSO and feature weighting. *2007 IEEE Congress on Evolutionary Computation*, (OCTOBER 2007), 2007.

[2] H. Çeker and S. Upadhyaya. Enhanced Recognition of Keystroke Dynamics using Gaussian Mixture Models. In *Military Communications Conference, MILCOM 2015-2015 IEEE*, number 2015-02, pages 1305–1310, 2015.

[3] Q. Chang, Q. Chen, and X. Wang. Scaling Gaussian RBF kernel width to improve SVM classification. *2005 International Conference on Neural Networks and Brain*, 1:19–22, 2005.

[4] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[5] A. Garg, S. Upadhyaya, and K. Kwiat. A User Behavior Monitoring and Profiling Scheme for Masquerade Detection. In V. Govindaraju and C. R. Rao, editors, *Handbook of Statistics – Machine Learning*, volume 31, pages 353–379. Elsevier, 2013.

[6] R. Giot and C. Rosenberger. A new soft biometric approach for keystroke dynamics based on gender recognition. *International Journal of Information Technology and Management*, 11(1/2):35, 2012.

[7] A. Graf and S. Borer. Normalization in support vector machines. *Lecture notes in computer science*, 2001.

[8] R. P. Guidorizzi. Security: Active authentication. *IT Professional*, (July/August):4–7, 2013.

[9] C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A Practical Guide to Support Vector Classification. *BJU international*, 101(1):1396–400, 2008.

[10] K. S. Killourhy and R. A. Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference on*, pages 125–134. IEEE, 2009.

[11] J. Leggett and G. Williams. Verifying identity via keystroke characterstics. *International Journal of Man-Machine Studies*, 28:67–76, 1988.

[12] W. Martono, H. Ali, and M. J. E. Salami. Keystroke pressure-based typing biometrics authentication system using support vector machines. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4706 LNCS(PART 2):85–93, 2007.

[13] B. Ngugi, B. K. Kahn, and M. Tremaine. Typing Biometrics: Impact of Human Learning on Performance Quality. *Journal of Data and Information Quality*, 2(2):1–21, 2011.

[14] M. V. S. Paula, E. a. Kinto, E. D. M. Hernandez, and T. Carvalho. User Authentication based on Human Typing Pattern with Artificial Neural Networks and Support Vector Machine. pages 484–493, 2005.

[15] G. Saggio, G. Costantini, and M. Todisco. Giovanni Saggio, Giovanni Costantini, Massimiliano Todisco. *Journal of Computer and Information Technology*, 1:2–11, 2011.

[16] Y. Sang, H. Shen, and P. Fan. Novel Impostors Detection in Keystroke Dynamics by Support Vector Machine. *Lecture Notes in Computer Science*, 3320:666–669, 2004.

[17] B. Song, G. Zhang, W. Zhu, and Z. Liang. ROC operating point selection for classification of imbalanced data with application to computer-aided polyp detection in CT colonography. *International journal of computer assisted radiology and surgery*, 9(1):79–89, jan 2014.

[18] K.-s. Sung and S. Cho. GA SVM wrapper ensemble for keystroke dynamics authentication. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3832 LNCS:654–660, 2006.

[19] P. S. Teh, A. B. J. Teoh, and S. Yue. A survey of keystroke dynamics biometrics. *The Scientific World Journal*, 2013, 2013.

[20] E. Vural, J. Huang, D. Hou, and S. Schuckers. Shared research dataset to support development of keystroke authentication. *IJCB 2014 - 2014 IEEE/IAPR International Joint Conference on Biometrics*, 2014.

[21] E. Yu and S. Cho. Novelty Detection Approach for Keystroke Dynamics Identity Verification. *Intelligent Data Engineering and Automated Learning*, 2690:1016–1023, 2003.