Retraining a Novelty Detector with Impostor Patterns for Keystroke Dynamics-Based Authentication

Hyoung-joo Lee and Sungzoon Cho*

Department of Industrial Engineering, Seoul National University,
San 56-1, Shillim-dong, Kwanak-gu, 151-744, Seoul, Korea
{impatton, zoon}@snu.ac.kr

Abstract. In keystroke dynamics-based authentication, novelty detection methods have been used since only the valid user's patterns are available when a classifier is built. After a while, however, impostors' keystroke patterns become also available from failed login attempts. We propose to retrain the novelty detector with the impostor patterns to enhance the performance. In this paper the support vector data description (SVDD) and the one-class learning vector quantization (1-LVQ) are retrained with the impostor patterns. Experiments on 21 keystroke pattern datasets show that the performance improves after retraining and that the one-class learning vector quantization outperforms other widely used novelty detectors.

1 Introduction

While passwords are the most popular in identity verification, they become vulnerable when they have been leaked out or stolen. To make up for the weakness, keystroke dynamics-based authentication has been motivated by the observation that a user's keystroke patterns are repeatable and distinct from those of other users [1]. It can be combined with passwords in almost a user-transparent fashion. Even if an impostor has obtained the password, the account can be protected from the intrusion through the keystroke-based authentication as illustrated in Figure 1. Other biometric methods have been also proposed for complementing or replacing the password method, e.g. fingerprint, iris, voice, etc. [2]. However, these methods need very expensive devices [3] and, more importantly, users may be reluctant to provide their biometric information. On the other hand, the keystroke-based method needs no additional device and keystroke data can be collected relatively easily.

Every time a user types his or her password, a keystroke pattern is defined. The times that each key is stroked and then released are measured by millisecond. A "duration" denotes a time interval during which a key is pressed and an "interval" a latency between a key and the next key. Then, a password of

* Corresponding author.

D. Zhang and A.K. Jain (Eds.): ICB 2006, LNCS 3832, pp. 633–639, 2005.

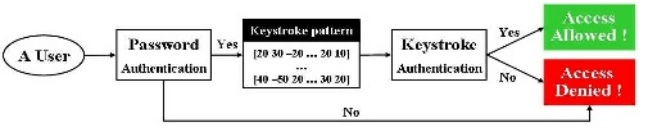

Fig. 1. The framework of a keystroke dynamics-based authentication: If a user types an incorrect password, the access will be immediately denied. Even if the correct password is presented, the keystroke pattern should be more or less accordant to the registered patterns to be allowed.

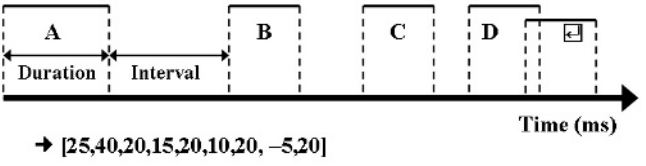

Fig. 2. Transforming a keystroke pattern into a timing vector when a user inputs a string 'ABCD': The duration and interval times are measured by millisecond

m characters can be transformed into a $(2m + 1)$-dimensional timing vector. Figure 2 illustrates how a string 'ABCD' can be represented as a 9D timing vector. The negative value means that the user released the 'D' after stroking the enter key.

After a number of the user's keystroke patterns are collected, a classifier is constructed based on them. One can employ a statistical model [3], [4], [5] or a neural network model [6]. Many of previous researches have used discrimination-based learning techniques, for which one needs to collect patterns from not only the valid user but also impostors. However, when building a classifier in the beginning, impostor patterns are impossible and undesirable to collect. This limitation can be overcome by the novelty detection framework [7], [8]. In novelty detection, the valid user's patterns are denoted as normal and all other individuals' patterns as novel. Then, a model learns characteristics of normal patterns and detects novel patterns that are different from the normal ones. In a geometric sense, the novelty detector defines closed boundaries around the normal patterns in input space [9].

Most of novelty detectors also have limitations that they are unsupervised learners assuming that novel patterns do not exist. So they utilize only normal patterns during training even though in some cases there exist, if few, novel patterns. For instance, intrusions may be attempted and somehow detected. Or even valid users may type their own passwords much differently from their typical keystroke patterns, so those patterns become impostor patterns. Although the impostor patterns are not sufficient to train a classifier, they can help a novelty detector generate more accurate and tighter boundaries. After impostors' keystroke patterns become available from those failed login attempts, the novelty detector which was trained only with the user's patterns can be retrained

with the impostor patterns. A few methods have been proposed to exploit novel patterns [10], [11]. It has been experimentally shown that one can achieve a higher accuracy by utilizing them. Recently, the authors proposed a method to take advantage of the novel patterns [11]. We propose to utilize, if available, the novel patterns for retraining. In particular, so-called one class learning vector quantization (1-LVQ) method [11] is recommended.

This paper is organized as follows. The next section introduces two novelty detectors which can utilize impostor patterns, the support vector data description (SVDD) [10] and the 1-LVQ. In Section 3, the SVDD and the 1-LVQ are applied to keystroke pattern datasets and compared with other novelty detectors. Finally, in Section 4 conclusions and discussion are given.

2 Retraining a Novelty Detector with Impostor Patterns

A valid user's timing vector pattern set constitutes training data $\mathbf{U} = \{(\mathbf{x}_i, y_i)|i = 1, 2, \cdots, N_{\mathbf{U}}\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ is a keystroke pattern represented as a timing vector and $y_i = +1$ is its class label. Then, a novelty detector can be trained with this training dataset. When a person tries to access a system, his keystroke pattern is measured and input to the novelty detector. If the detector recognizes the pattern as normal, he will be allowed access. If the detector rejects the pattern as novel, on the other hand, he will be denied access.

In the beginning, only the user's keystroke patterns are available. However, as time passes, impostor patterns may appear in some cases. First, some impostors may attempt access and be somehow detected and denied access. Second, even the valid user may type inconsistently. When he is rejected as an impostor, that pattern is regarded as an impostor pattern. In these cases, a set of impostor patterns can be denoted as $\mathbf{I} = \{(\mathbf{x}_i, y_i)\}, y_i = -1$. Then, a combined training dataset $\mathbf{X} = \mathbf{U} \cup \mathbf{I}$ can be formed. Usually, the number of the valid user's patterns is much greater than that of impostor patterns, i.e. $|\mathbf{U}| \gg |\mathbf{I}|$, making it impossible to train a binary classifier. Instead, a novelty detector is trained with the user's patterns and then later retrained when the impostor patterns become available. Among novelty detectors that can be retrained with the impostor patterns, the SVDD [10] and the 1-LVQ [11] are considered in this paper.

2.1 Support Vector Data Description (SVDD)

An SVDD tries to define a hypersphere with a minimal volume so that it surrounds as many normal patterns and as few novel patterns as possible [10]. The radius and the center of the hypersphere is denoted respectively as R and $\mathbf{a}$. They can be found by, like a support vector machine (SVM), the standard quadratic programming techniques. In the authentication process, an unknown keystroke pattern $\mathbf{z}$ is accepted as the genuine user's if $\|\mathbf{z} - \mathbf{a}\|^2 \leq R^2$, or rejected as an impostor's otherwise. Using Mercer kernels allows to define boundaries more flexible than just a hypersphere. If a radial basis function (RBF) kernel is employed, the SVDD provides a solution in essentially the same form as the Parzen window or the one-class support vector machine (1-SVM).

2.2 One-Class Learning Vector Quantization (1-LVQ)

The 1-LVQ algorithm is a modified form of the original LVQ [11]. Just as an LVQ, a 1-LVQ is initialized by updating codebooks using a conventional SOM. When updating initial codebooks, only the normal patterns are used. The SOM generates a set of codebooks $\mathbf{W} = \{\mathbf{w}_k | k = 1, 2, \ldots, K\}$, $K \ll N$ to represent the normal data. When codebook update is done, the codebook $\mathbf{m}(\mathbf{x})$ of an input pattern $\mathbf{x}$ and the Voronoi region $\mathbf{S}_k$ of each codebook $\mathbf{w}_k$ are defined.

When the training set includes the novel patterns, a learning rule different from the conventional one can be obtained as follows,

$$\mathbf{w}_k \leftarrow \begin{cases} \mathbf{w}_k, & \text{if } \mathbf{x}_i \notin \mathbf{S}_k, \\ \mathbf{w}_k + \eta(\mathbf{x}_i - \mathbf{w}_k), & \text{if } \mathbf{x}_i \in \mathbf{U}_k, \\ \mathbf{w}_k - \eta(\mathbf{x}_i - \mathbf{w}_k), & \text{if } \mathbf{x}_i \in \mathbf{I}_k, \end{cases} \tag{1}$$

where $\mathbf{U}_k = \mathbf{U} \cap \mathbf{S}_k$ and $\mathbf{I}_k = \mathbf{I} \cap \mathbf{S}_k$. According to this rule, the normal patterns "pull" their codebooks while the novel patterns "push away" theirs.

Since the 1-LVQ, unlike the LVQ, assigns all the codebooks to the normal data, thresholds should be explicitly determined. While some codebooks lie inside dense lumps of input patterns, others lie in regions where patterns are sparsely scattered. For that reason, it is desirable to set different thresholds for different codebooks. For each Voronoi region, a hypersphere with a center at $\mathbf{w}_k$ and a minimal radius can be obtained, so that it surrounds as many normal patterns and as few novel patterns as possible. Now, classification is done as follows. Given a keystroke pattern $\mathbf{z}$, its codebook $\mathbf{m}(\mathbf{z}) = \mathbf{w}_q$ is found. Then, $\mathbf{z}$ is accepted if $\|\mathbf{z} - \mathbf{w}_q\|^2 \leq (r_q^*)^2$, or rejected otherwise.

3 Experimental Results

A program was developed to measure keystroke patterns. The data were collected via the keyboard connected to a workstation from 21 valid users, whose passwords are listed in the first column of Table 1. Many of them would not be understandable, since they are written in Korean alphabet, e.g. '`rhkdwo`', '`dhfpql.`', and '`tjddmswjd`'. They are simply shown in the corresponding English characters, with regard to their positions on the keyboard. The 21 users typed their own passwords with lengths ranging from six to ten characters, generating the normal class of data, and to simulate potential intrusion attempts, 15 "impostors", who had practiced the passwords, typed the 21 users' passwords. In all, 21 datasets were constructed for 21 passwords. For each user's password, 76 to 388 normal patterns were collected for training and 75 for test and 75 novel patterns were also collected. Since we assumed that the training set should be highly imbalanced, 50 user's patterns and 5 impostor patterns were randomly sampled for training. The 75 normal patterns and the rest of 70 novel patterns constituted the test set. A total of 30 different training and test sets were randomly sampled for each password to reduce a sampling bias.

We applied a total of six novelty detectors including the SVDD and the 1-LVQ. The other models are the Gaussian (Gauss) and the Parzen (Parzen) density estimator, the auto-associative neural network (AANN) and the 1-SVM, all of which cannot utilize impostor patterns even when they are available. The keystroke authentication has two types of error, i.e. false acceptance rate (FAR) and false rejection rate (FRR) [12]. Since one type of error can be reduced at the expense of the other, choosing an appropriate trade-off point should be critical to authentication accuracy. In order to avoid the biases introduced by possibly arbitrary parameters and threshold selection, we compared these models in terms of the ingetrated error [10] which was obtained from an ROC curve by integrating the FARs over the FRR from 0 to 50%.

We are interested in the effects of utilizing impostor patterns. For each password, the SVDD and the 1-LVQ were trained with two types of training dataset, one with both the user's and impostors' keystroke patterns and the other with only the user's. In other words, we compared the novelty detectors retrained with impostor patterns and the detectors trained only with the user's patterns. The average integrated errors for 21 passwords are listed in Table 1. For 16 out of 21 passwords, the 1-LVQ trained with the both classes performed better and its integrated errors were statistically lower with a significant level of 10%. For three passwords, '`autumnman`', '`dusru427`', and '`yuhwa1kk`', both models could achieve the minimum error, i.e. 0%. The 1-LVQ trained only with the user's patterns has never produced lower error rates. To sum up, when utilizing impostor patterns, the 1-LVQ produced errors as much as 67% lower and, on average, 27% lower. The SVDD trained with the both classes gave lower, though at best marginally, errors only for 4 passwords. Table 1 shows that by utilizing impostor patterns, the 1-LVQ improved much more than the SVDD did.

The SVDD and the 1-LVQ were compared with other four novelty detectors. It was assumed that the SVDD and the 1-LVQ were retrained with the impostor patterns while the other four models were trained only with 50 user's patterns. It may sound unfair, but in practice there is nothing that the four models can do with the impostor patterns anyway. The integrated errors of six models for 21 passwords are listed in Table 1. The 1-LVQ turns out to be the best, resulting in the lowest errors for 13 out of 21 passwords. Among them, the errors for 11 passwords were statistically lower than those of other models with a significance level of 10%. On average, the 1-LVQ produced 26% lower errors than the second best model, the 1-SVM. The 1-SVM has not produced the lowest error for any of the passwords, but ranked second in most cases. For four passwords, '`autumnman`', '`dltjdgml`', '`dusru427`', and '`yuhwa1kk`', most models achieved a perfect classification. The users of these passwords probably have very unique keystroke patterns. The errors of the SVDD were slightly, but not significantly, higher than those of the 1-SVM, another support vector-based method. The SVDD were as accurate as the other models except the 1-LVQ. The Gauss produced the lowest errors for two passwords and, on average, was comparable to the Parzen, suggesting that keystroke patterns probably are distributed in a unimodal manner and that just a hyperellipsoid can be a decent solution. The

Table 1. The average integrated errors (%) for six novelty detectors: The columns denoted as 'Both' and 'Normal' for the 1-LVQ and the SVDD respectively indicate models trained with the both classes of patterns and with only the user's patterns. The bold faced figures indicate the lowest errors for the corresponding password. An asterisk indicates that the marked model is better than the other models with a significance level of 10%.

Passwords	1-LVQ		SVDD		Gauss	Parzen	AANN	1-SVM
	Both	Normal	Both	Normal				
90200jdg	1.88	2.15	1.97	1.97	**1.58***	2.66	3.43	1.91
ahrfus88	**0.33***	0.46	0.50	0.50	0.73	0.59	0.77	0.48
anehwksu	0.28	0.38	0.33	0.33	**0.16***	0.43	1.46	0.30
autumnman	0.00	0.00	0.00	0.00	0.00	0.00	0.40	0.00
beaupowe	0.02	0.02	0.01	0.01	0.01	0.02	0.81	0.01
c.s.93/ksy	**0.14***	0.19	0.19	0.19	0.82	0.22	0.23	0.19
dhfpql.	**0.71***	0.89	0.86	0.87	0.85	1.12	1.99	0.80
dirdhfmw	**0.37***	0.83	0.96	0.96	1.30	1.37	2.13	0.87
dlfjs wp	**0.42***	0.45	0.45	0.45	0.49	0.50	2.26	0.45
dltjdgml	0.00	0.00	0.00	0.00	0.02	0.00	0.17	0.00
drizzle	**0.06**	0.10	0.09	0.09	0.26	0.18	0.74	0.08
dusru427	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00
i love 3	**0.86**	1.04	1.13	1.14	0.87	1.22	2.68	1.05
love wjd	**0.85***	1.39	1.70	1.72	2.24	2.14	3.38	1.59
loveis.	**0.32***	0.44	0.42	0.42	0.93	0.50	0.98	0.41
manseiii	**0.59***	1.02	1.19	1.20	2.45	1.46	1.94	1.08
rhkdwo	**0.77***	1.32	1.45	1.45	1.58	2.23	2.66	1.38
rla sua	0.01	0.03	0.01	0.01	0.06	0.06	0.38	0.01
tjddmswjd	**0.24***	0.38	0.46	0.46	1.23	0.96	2.31	0.40
tmdwnsl1	**1.13***	1.18	1.22	1.22	1.36	1.32	2.07	1.20
yuhwa1kk	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TotalAvg	**0.43**	0.59	0.62	0.62	0.81	0.81	1.47	0.58

AANN might have difficulties in training, since a lot of training patterns are necessary to train an AANN in a high-dimensional space.

4 Conclusions and Discussion

We applied two novelty detectors to keystroke dynamics authentication. Even though impostor patterns are not available in the beginning, they become available later from failed login attempts. We proposed to retrain the novelty detectors using them. Experiments on 21 keystroke pattern datasets have demonstrated that the 1-LVQ can take advantage of the impostor patterns, though the improvements of the SVDD were no more than marginal. Compared with other widely used novelty detectors, the 1-LVQ has shown its competence as an authenticator, resulting in significantly lower integrated errors.

A few limitations and future directions should be addressed. First, while we have not considered the issue of parameter selection, in practice a particular set of

parameters should be specified in advance. It is tricky to select proper parameters for both the SVDD and the 1-LVQ. Second, while we have arbitrarily sampled 5 impostor patterns, it demands an investigation on how many impostor patterns are needed for the SVDD or the 1-LVQ. Third, we have included all the features for training. However, some features are more important than others while some features may be useless or even be harmful. So, a good feture selection scheme can improve the accuracy. Fourth, one might want to know in advance whether retraining with impostor patterns will be useful. Certain quality measures may be employed to determine the utility of retraining.

Acknowledgement

This work was supported by grant No. R01-2005-000-103900-0 from the Basic Research Program of the Korea Science and Engineering Foundation.

References

1. Gaines, R., Lisowski, W., Press, S., Shapiro, N.: Authentication by keystroke timing: some preliminary results. Rand Report R-256-NSF. Rand Corporation. (1980)
2. Jain, A.K., Bolle, R., Pankanti, S.: Biometrics: Personal Identification in Networked Society. Kluwer, Norwell (1999)
3. Monrose, F., Rubin, A.D.: Keystroke Dynamics as a Biometric for Authentication. Future Generation Computer System 16(4) (2000) 351-359
4. Araújo, L.C.F., Sucupira Jr., L.H.R., Lizárraga, M.G., Ling, L.L., Yabu-Uti, J.B.T.: User Authentication through Typing Biometrics Features. IEEE Transactions on Signal Processing 52(2) (2005) 851-855
5. Bleha, S., Slivinsky, C., Jussein, B.: Computer-access Security Systems using Keystroke Dynamics. IEEE Transactions on Pattern Analysis and Machine Intelligence 12(12) (1990) 1217-1222
6. Obaidat, M.S.. Sadoun, B.: Verification of Computer Users using Keystroke Dynamics. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 27(2) (1997) 261-269
7. Cho, S., Han, C., Han, D., Kim, H.: Web Based Keystroke Dynamics Identity Verification using Neural Networks. Journal of Organizational Computing and Electronic Commerce 10(4) (2000) 295-307
8. Yu, E., Cho, S.: Keystroke Dynamics Identity Verification - Its Problems and Practical Solutions. Computer and Security 23(5) (2004) 428-440
9. Schölkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., Williamson, R.C.: Estimating the Support of a High-dimensional Distribution. Neural Computation 13 (2001) 1443-1471
10. Tax, D.M.J., Duin, R.P.W.: Support Vector Data Description. Machine Learning 54 (2004) 45-66
11. Lee, H., Cho, S.: SOM-based Novelty Detection Using Novel Data. In: Proceedings of Sixth International Conference on Intelligent Data Engineering and Automated Learning, Lecture Notes in Computer Science 3578 (2005) 359-366
12. Golfarelli, M., Maio, D., Maltoni, D.: On the Error-Reject Trade-off in Biometric Verification Systems. IEEE Transactions on Pattern Analysis and Machine Intelligence 19(7) (1997) 786-796