Continual Retraining of Keystroke Dynamics Based Authenticator

Pilsung Kang, Seong-seob Hwang, and Sungzoon Cho*

Seoul National University, San 56-1, Shillim-dong, Kwanak-gu, 151-744, Seoul, Korea
{xfeel80,hss9414,zoon}@snu.ac.kr

Abstract. Keystroke dynamics based authentication (KDA) verifies a user based on the typing pattern. During enroll, a few typing patterns are provided, which are then used to train a classifier. The typing style of a user is not expected to change. However, sometimes it does change, resulting in a high false reject. In order to achieve a better authentication performance, we propose to continually retrain classifiers with recent login typing patterns by updating the training data set. There are two ways to update it. The *moving window* uses a fixed number of most recent patterns while the *growing window* uses all the new patterns as well as the original enroll patterns. We applied the proposed method to the real data set involving 21 users. The experimental results show that both the moving window and the growing window approach outperform the fixed window approach, which does not retrain a classifier.

1 Introduction

Password based user authentication has been widely used in security systems where a keyboard or a keypad is the main input device. When a user exposes his password or a third-party acquires the password, however, the authentication system becomes vulnerable to improper access trials. In order to cope with the vulnerability of password based user authentication, keystroke dynamics based authentication (KDA) has been proposed[1][2]. KDA considers keystroke dynamics together with password characters when authenticating an access attempt, i.e., it utilize a user's behavior as well as knowledge.

KDA involves three main processes as shown in Fig. 1. The enrollment process involves a user enrolling a certain number of keystroke typing patterns. The classifier building process involves constructing a classifier with the typing patterns collected in the enrollment process. The authentication process evaluates an access attempt with the classifier and takes an appropriate action: grant or deny access.

Recently, a number of methods for building high performance classifiers have been reported[3][4][5]. According to Peacock et al.[6], however, high performance classifiers reported in the literature were built with a large number of typing patterns. Most classifiers that they reviewed used more than 100 typing patterns

* Corresponding author.

S.-W. Lee and S.Z. Li (Eds.): ICB 2007, LNCS 4642, pp. 1203–1211, 2007.

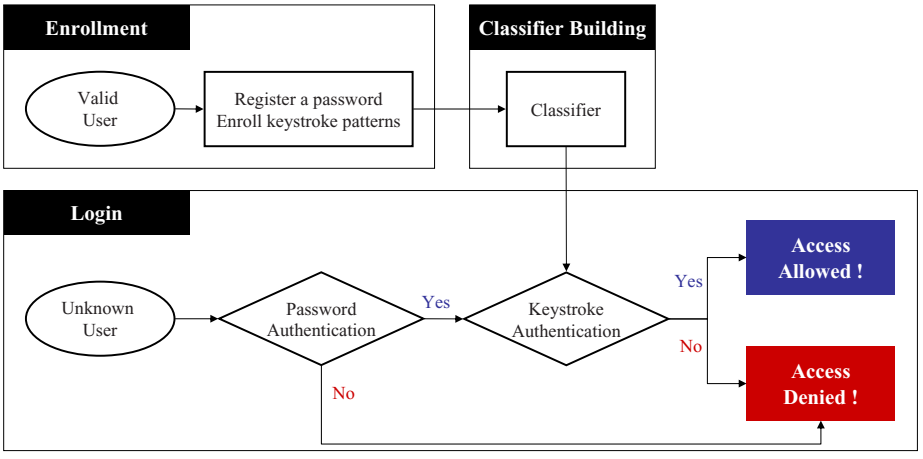
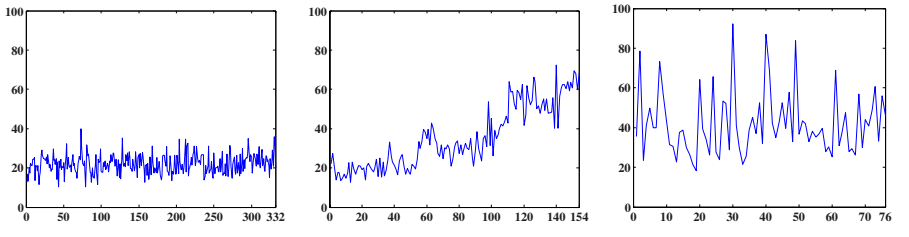


Fig. 1. Keystroke dynamics based authentication process

from each user to build the classifiers. Some even used more than 1,000 typing patterns. In practice, however, the number of patterns collected in the enrollment process should be no more than 10. Otherwise, a user may not choose to use the system at all. With such a small number of typing patterns, it is essential that the typing style of a user will not change in the future. It is often the case, however, that the typing style of a user changes. In this case, the performance of the classifier may degenerate. Fig. 2 shows the distances between the login patterns and the mean vector of the enroll patterns of three users involved in our experiment. The x-axis represents the index of login typing patterns. The larger the index is, the more recent the pattern is. The y-axis represents the Euclidian distance of the login pattern to the mean vector. The mean vector was calculated using 10 enroll patterns. User 9's typing style did not change over time since it shows little fluctuation (Fig. 2(a)). User 1 (Fig. 2(b)) is a typical user whose typing style changed gradually over time. The distance of the login patterns to the mean vector increased steadily. User 16 (Fig. 2(c)), on the other hand, surely changed his typing style, but no trend can be extracted since the



(a) User 9 (b) User 1 (c) User 16

Fig. 2. Distance between login pattern and mean enroll pattern

distance neither increased nor decreased. The fluctuation is larger than the other users and many spikes are presented.

A remedy for changing typing style is to keep retraining the classifier. In this paper, we propose to update the training pattern set dynamically in order to accommodate the change of a user's typing behavior. In particular, we add "successful" login patterns to the training set and retrain the classifier. We propose two different ways of updating the training pattern set. The first is *"Moving window"*, where the number of training patterns is fixed. A recent pattern is added to the training set, while the least recent training pattern is removed. The second is *"Growing window"*, where a recent pattern keeps being added without any training pattern removed. Thus, the number of training patterns keeps increasing.

The rest of this paper is structured as follows. In section 2, we explain moving window and growing window followed by the experimental settings including authentication algorithm and data description in section 3. In section 4, we compare the performance of the classifiers based on two proposed methods and the *"Fixed window"*.[1] In section 5, with a conclusion, we discuss future work.

2 Updating Training Set

When KDA is employed in real systems, only a few number of typing patterns can be collected in the enrollment process. A small number of training patterns have a limitation that they can only represent a user's typing behavior at a particular time. If a user's typing style changes over time, newly available login patterns need to become a part of the training data set to retrain the classifier. We propose two ways of updating the training pattern set as follows.

2.1 Moving Window

In *moving window* method, the number of training patterns is fixed. When a new pattern is added to the training set, the oldest training pattern is removed. Let $TD_i = \{x_{m+1}, x_{m+2}, ..., x_{m+n}\}$ denote the training set at time period i, where n is the size of the training set. x_{m+1} is the oldest training pattern while x_{m+n} is the newest training pattern. With a new typing pattern x' available, TD_{i+1} is updated as follows:

$$\begin{aligned} &if \quad x' \quad is \quad granted \quad access \\ &\qquad TD_{i+1} = TD_i - \{x_{m+1}\} \cup \{x'\} \\ &else \\ &\qquad TD_{i+1} = TD_i \end{aligned} \tag{1}$$

If x' is granted access, it is added to TD_{i+1} while the oldest training pattern is removed. Otherwise, it is not added to TD_{i+1} so that the training data set

[1] *"Fixed window"* refers to the invariant training set consisting of the typing patterns collected in the enrollment process.

remains the same as TD_i. When a user's typing style changes, moving window can accommodate the change immediately. If the change between successive typing patterns is large but has no direction, however, moving window becomes unstable.

2.2 Growing Window

In *growing window* method, the number of training patterns is not fixed but increases. Ever when a new pattern is added to the training set, the oldest training pattern stays so that the growing window stores all the typing patterns that have been granted access as well as the original enroll patterns. The number of training patterns increases by one every time a new typing pattern is added. Let $TD_i = \{x_1, x_2, ..., x_n\}$ denote the training set at time i, where n is the size of the current training set. x_1 is the oldest training pattern, while x_n is the newest training pattern. With a new typing pattern x' available, TD_{i+1} is updated as follows:

$$\begin{aligned} &if \quad x' \quad is \quad granted \quad access \\ &\qquad TD_{i+1} = TD_i \cup \{x'\} \\ &else \\ &\qquad TD_{i+1} = TD_i \end{aligned} \tag{2}$$

If x' is granted access, it is added to TD_{i+1} while the oldest training pattern is not removed. So the number of training patterns becomes $n + 1$. If x' is not granted access, on the other hand, it is not added to TD_{i+1} so that the training data set remains the same as TD_i. Since growing window method stores all possible training patterns, it reflects the change less than moving window when a user changes his typing style. If the change of typing patterns is not progressive and the scope of change is large, it can be more stable than moving window.

3 Experimental Settings

The data set used in our experiment originally appeared in Cho et al[4]. The data set consists of 21 users, each of whom has different number of training patterns from 76 to 388 as shown in Table 1. We used two types of features: duration (time between a key press and release) and interval (time between a key release and another key press). If a user uses a password with n characters, the length of the features becomes $2n - 1$: n durations and $n - 1$ intervals. Each user has 75 typing patterns of the valid user and 75 typing patterns of impostors. It is not easy to collect impostors' patterns in practice, thus, the subjects changed their roles such that a user tried to log in with other users' passwords. Initially, the earliest 10 patterns in the training set were used to build a classifier for all methods.

We employed K-Means algorithm based on Euclidian distance as the authentication classifier. The authentication algorithm is demonstrated in Fig. 3. First, K-means algorithm is performed using the training patterns so that the cluster membership of each training pattern is determined. Second, when the test

Table 1. Data description

User ID	Password	No. of features	No. of patterns	User ID	Password	No. of features	No. of patterns
1	90200jdg	15	164	12	dusru427	15	365
2	ahrfus88	15	260	13	i love 3	15	330
3	anehwksu	15	319	14	loveis.	13	207
4	autumnman	17	111	15	love wjd	15	101
5	beaupowe	15	76	16	manseiii	15	86
6	c.s.93/ksy	19	200	17	rhkdwo	11	205
7	dhfpql.	13	232	18	rla sua	15	101
8	dirdhfmw	15	309	19	tjddmswjd	17	337
9	dlfjs wp	15	342	20	tmdwnsl1	15	108
10	ditjdgml	15	151	21	yuhwa1kk	15	388
11	drizzle	13	299				

Step 1: Perform K-Means clustering with the training patterns

$[C_1, ..., C_K]$ = K-Means(X,K)

(C_K: the members belonging to K-th cluster)

(X: training patterns, K: the number of clusters)

Step 2: Find the closest cluster prototype of the test pattern y_j

$k = \arg_{i \in 1,...,K} \min dist(y_j, P_i)$

(P_i: the prototype of the cluster C_i)

Step 3: Authentication

If $\text{dist}(y_j, P_i) < M \times \frac{1}{|N_k|} \sum_{x_i \in C_k} dist(x_i, P_k)$

(N_k: the number of patterns in k-th cluster, M: Threshold coefficient)

grant access(y_j is considered as a valid user's typing pattern)

else

deny access(y_j is considered as an impostor's typing pattern)

Fig. 3. KDA authentication process

pattern is given, the nearest prototype to the test pattern is determined. Third, the average distance between training patterns belonging to their nearest prototype is calculated to estimate the threshold. If the distance between the test pattern and the nearest prototype is smaller than the threshold, the test pattern is granted access. Otherwise, it is denied access. In our experiment, we set the number of clusters (K) to 3.

There are two types of errors: false rejection rate (FRR) and false acceptance rate (FAR). FRR is the ratio of the valid user's attempts classified as impostors'. A high FRR indicates that the valid user's attempt is often rejected. A user may feel irritated if FRR is high. FAR is the ratio of impostors' attempts classified as the valid user's. The system is not secure when FAR is high. Since the purpose of KDA is to minimize both FRR and FAR, and one can be reduced at the cost of the other, we adopted equal error rate (EER, the value where FRR and FAR are equal[7]) for the performance measure.

4 Experimental Results

The EERs of all users using fixed window, moving window, and growing window are shown in Table 2. The average EER of fixed window is 4.8%. Both moving window and growing window, however, achieved average EER of 3.8%, which is 1.0% lower than that of fixed window. Since the variation of the users was fairly large, we investigated the effect of moving window and growing window on each user separately. First, we constructed a *win-draw-lose* table in order to estimate the relative dominance of moving window and growing window over fixed window (see Table 4). The EERs of moving window and growing window were compared for each user. If the difference was larger than 0.01, "win" or "lose" resulted. Otherwise, "draw" resulted. Five users lowered their EER with moving window while only one user resulted in a higher EER. Similarly, six users lowered their EER with growing window while none of the users resulted in a higher EER. Thus, we can conclude that both moving window and growing window improved EER over fixed window.

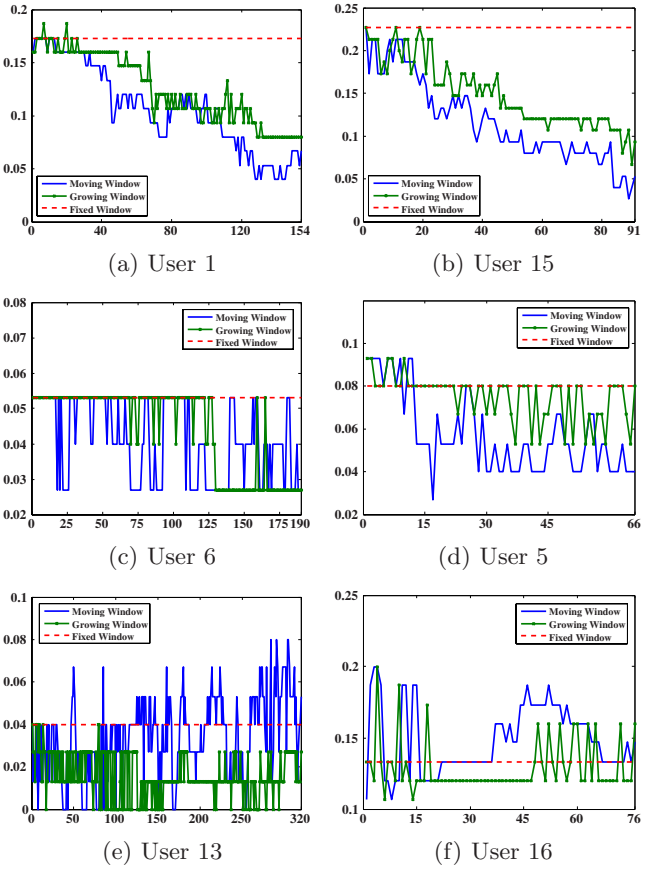
We also plotted the EER over time (Fig. 4) for three methods as the training data set is updated. The x-axis represents the index of typing patterns. A newer

Table 2. The average EER of fixed window, moving window, and growing window

User ID	Fixed window	Moving window	Growing window	User ID	Fixed window	Moving window	Growing window
1	0.173	0.107	0.126	12	0.000	0.000	0.000
2	0.027	0.026	0.027	13	0.040	0.033	0.016
3	0.000	0.000	0.000	14	0.093	0.090	0.081
4	0.000	0.001	0.000	15	0.227	0.117	0.149
5	0.080	0.057	0.075	16	0.133	0.149	0.129
6	0.053	0.041	0.044	17	0.000	0.002	0.001
7	0.013	0.013	0.013	18	0.040	0.035	0.031
8	0.000	0.005	0.003	19	0.000	0.000	0.000
9	0.013	0.013	0.013	20	0.013	0.013	0.013
10	0.000	0.000	0.000	21	0.000	0.000	0.000
11	0.107	0.096	0.082	**Average**	**0.048**	**0.038**	**0.038**

Table 3. Win-draw-lose table of moving window and growing window over fixed window

		Fixed Window		
		Win	Draw	Lose
Moving Window	Total Number	5	15	1
	User ID	1,5,6,11,15	-	16
Growing Window	Total Number	6	15	0
	User ID	1,6,11,13,14,15	-	-



(a) User 1 (b) User 15

(c) User 6 (d) User 5

(e) User 13 (f) User 16

Fig. 4. EER for a sequence of login attempts of some users

pattern has a larger index. The y-axis represents the EER. The users plotted in Fig. 4(a,b,c) had better authentication performance with both moving window and growing window than fixed window. User 1 and User 15 are typical users

whose typing style has changed over time. The EERs continuously decreased as the training patterns were updated. Lower EER was achieved by adjusting training patterns so as to accommodate the user's typing behavior change. Since moving window is more sensitive to typing change than growing window, its EER decreased more rapidly. User 6 has a different change pattern. The training patterns changed irregularly. Moving window accommodated the change as soon as a new pattern was added so that the fluctuation of the EER was rather large. Growing window, on the other hand, did not show much fluctuation of the EER since it used a large number of training patterns. Fig. 4(d) shows User 5 whose EER decreased with moving window but did not with growing window. Note that the training patterns numbered from 10 to 25 seem significantly different from the initial 10 training patterns. Since moving window used newly available patterns only, the EER decreased rapidly. In growing window, on the other hand, the change was accommodated in the retrained classifier with some delay. Fig. 4(e) shows User 13 whose EER decreased with growing window but did not with moving window. The typing style of the user has changed, but it is neither consistent nor has a direction. In this case, moving window *"chased"* after every single change so that it resulted in the unstable EER. Growing window, on the other hand, had enough number of training patterns to absorb the negative effect of a few spurious patterns so that a little fluctuation with low EER resulted. Fig. 4(f) shows User 16 whose EER was larger with moving window than with fixed window. At the beginning, neither moving window nor growing window could catch the change. After 20 patterns, however, both moving window and growing window became stable and the EER decreased. For some reason, moving window could not reflect the typing behavior change between the 35th typing pattern and the 65th typing pattern while growing window could. Therefore, the average EER of moving window became larger than than that of fixed window.

5 Conclusion

In this paper, we proposed to update the training set dynamically so that the classifier can accommodate the change of typing behavior. Based on the experiments involving 21 users, updating training sets by moving window or growing window improved the authentication accuracy over fixed window. There are a couple of limitations in the current work. We fixed the window size to 10 in moving window. The effect of the window size on authentication performance needs to be investigated. In addition, since the typing patterns used in our experiment were collected in a rather short period, data collection with longer period such as 6 months or 1 year should be investigated.

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, the Brain

Korea 21 program in 2006 and 2007, and partially supported by Engineering Research Institute of SNU.

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. Umphress, D., Williams, G.: Identity Verification through Keyboard Characteristics. International Journal of Man-Machine Studies 23, 263–273 (1985)
3. Obaidat, M., Sadoun, S.: Verification of computer users using keystroke dynamics. IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics 27(2), 262–269 (1997)
4. Cho, S., Han, C., Han, D., Kim, H.: Web-based keystroke dynamics identity verification using neural network. J. Organizational computing and electronics commerce 10(4), 295–307 (2000)
5. Brown, M., Rogers, S.J.: User Identification via keystroke characteristics of typed names using neural networks. Int. J. Man-Machine Studies 39, 999–1014 (1993)
6. Peacock, A., Ke, X., Wilkerson, M.: Typing Patterns: A Key to User Identification. IEEE Security & Privacy Magazine 5(2), 40–47 (2004)
7. Hwang, S., Lee, H., Cho, S.: mproving Authentication Accuracy of Unfamiliar Passwords with Pauses and Cues for Keystroke Dynamics-Based Authentication. In: Chen, H., Wang, F.-Y., Yang, C.C., Zeng, D., Chau, M., Chang, K. (eds.) WISI 2006. LNCS, vol. 3917, pp. 73–78. Springer, Heidelberg (2006)