Hindawi Publishing Corporation
Journal of Sensors
Volume 2015, Article ID 935986, 14 pages
http://dx.doi.org/10.1155/2015/935986

Research Article

Sequence Alignment with Dynamic Divisor Generation for Keystroke Dynamics Based User Authentication

Jiacang Ho[1] and Dae-Ki Kang[2]

[1]*Department of Ubiquitous IT, Graduate School, Dongseo University, 47 Jurye-Ro, Sasang-Gu, Busan 617-716, Republic of Korea*
[2]*Department of Computer & Information Engineering, Dongseo University, 47 Jurye-Ro, Sasang-Gu, Busan 617-716, Republic of Korea*

Correspondence should be addressed to Dae-Ki Kang; dkkang@dongseo.ac.kr

Received 28 November 2014; Revised 5 January 2015; Accepted 20 January 2015

Academic Editor: Fei Yu

Keystroke dynamics based authentication is one of the prevention mechanisms used to protect one's account from criminals' illegal access. In this authentication mechanism, keystroke dynamics are used to capture patterns in a user typing behavior. Sequence alignment is shown to be one of effective algorithms for keystroke dynamics based authentication, by comparing the sequences of keystroke data to detect imposter's anomalous sequences. In previous research, static divisor has been used for sequence generation from the keystroke data, which is a number used to divide a time difference of keystroke data into an equal-length subinterval. After the division, the subintervals are mapped to alphabet letters to form sequences. One major drawback of this static divisor is that the amount of data for this subinterval generation is often insufficient, which leads to premature termination of subinterval generation and consequently causes inaccurate sequence alignment. To alleviate this problem, we introduce sequence alignment of dynamic divisor (SADD) in this paper. In SADD, we use mean of Horner's rule technique to generate dynamic divisors and apply them to produce the subintervals with different length. The comparative experimental results with SADD and other existing algorithms indicate that SADD is usually comparable to and often outperforms other existing algorithms.

1. Introduction

In an era which is full of electronic services, people want to have more convenient and faster ways to assist their needs. This includes reading emails, searching information through online communication, transferring files, and paying bills online. For example, online file transfer involves storage mechanisms in a cloud system. To use these services for our private needs, we must register a login ID and password. As for the online bill payment, we must login with our ID and password before we can pay or transfer money to other users. However, it can happen that criminals detect our login ID and password and utilize our credentials to commit crimes such as stealing important files or money. Therefore, stronger and more secure authentication mechanism has to be designed and implemented to prevent these issues.

Considerable amount of authentication mechanisms has been introduced. One of the examples is a biometric system [1]. Biometric system can be divided into two folds, which are physiology-based approach and behavior-based approach. Physiology-based approaches in the authentication system include use of iris, voice, and fingerprint. In contrast, behavior-based approach includes keystroke dynamics on keyboard, mouse, or smart phone. In this paper, we focus more on the behavior-based approach. Behavior-based approach has advantages that it is inexpensive, is easy to implement, and needs no extra hardware to operate. One possible drawback of the behavior-based approach is that its performance can be lower than that of physiology-based approach, but it can reduce the personal risk of a user. For example, in fingerprint-based authentication, an imposter can cut off the genuine user's hand in order to access the security system. However, this kind of attack is ineligible for behavior-based approach. If the imposter wants to break a system, his choice can be limited in either forcing the genuine user to type it or emulating the genuine user's typing by practice. Therefore, there is still a considerable amount of researchers performing behavior-based research because

they believe that keystroke dynamics can improve the security and it will be a common approach to be used in the future [2–10].

Keystroke dynamics concerns timing details of people's typing data [11]. The timing detail means the duration between pressing a key and releasing a key or vice versa. Keystroke dynamics can be applied with the machine learning to discover knowledge about people's typing behavior [8], emotion [12], gender [3], dominant hand [4], and so forth. In this paper, we associate the typing behavior with the authentication system. It is reasonable to use the keystroke dynamics in the authentication system because some users have special typing patterns. For example, some people use only one hand to type whereas the others use both hands [4]. Moreover, Syed et al. [8] have proven three interesting hypotheses in their research. One of the evidence is that users present significantly dissimilar pattern when typing. Another hypothesis is about the relationship between users' typing ability and the event sequence. The event sequence is defined as the sequence of key-up and key-down events with actual key values incorporated. More explanation for event sequence can be reviewed in their paper [8]. In their result, it is shown that there is no correlation between users' typing skill and the event sequence. Last hypothesis discusses the effect of habituation on the event sequences of a user. It reveals that the keystroke dynamics data that is typed later is more representative than the data which is typed earlier. This means that it is difficult for imposters to imitate legitimate users' typing patterns. Besides typing behavior, we can observe that different people have different walking styles (or gait). That is, we can conjecture a person correctly just by hearing the footstep without seeing him/her. It can be possible that, sometimes, we can surmise a person correctly by looking at the appearances and walking style from behind. In summary, we can identify a person easily by using these special characteristics or so-called special habits. These characteristics or habits include typing behavior.

The common timing details that we can obtain from keystroke dynamics are dwell time and flight time. Dwell time, also known as duration time [9], is the length of time between when a key is pressed and when a key is released. Flight time, also known as interval time [9], however, is the length of time between when a key is released and when a next key is pressed. More details are discussed in Section 4.

There is a considerable amount of machine learning algorithms introduced for keystroke dynamics such as naïve Bayes [13], support vector machines (SVM) [10], nearest-neighbor [2], and Euclidean distance measure [14]. In this paper, we choose sequence alignment algorithm. Sequence alignment algorithm is fundamentally concerned about measuring the similarity between multiple sequences. Suppose a user logs in an account in any platform, the system has to check the existence of her identification information before it checks her password. It is well known that this password matching involves complicated encryption of plain text in symmetric password systems [15] or modular arithmetic operations for asymmetric password systems [16]. However, in a more abstract view, this is basically string matching. When a system checks the password, firstly it checks the first letter from the currently inserted password with the first letter from the stored password in the database, followed by the second letter, the third letter, and so forth until all letters are verified. Sequence alignment is a more general and stronger algorithm which measures the similarity between objects. Hence, we consider that sequence alignment algorithm is an appropriate algorithm to be used in this paper.

Of the current research we are aware of, there is still no specific algorithm to be used as the common algorithm in keystroke dynamics research. However, in Revett's research [7], he shows that the sequence alignment algorithm has performed sufficiently when it is applied into the keystroke dynamics. Besides that, some researchers have provided new idea into keystroke dynamics. Giot and Rosenberger's [3] research introduces a new soft biometric for keystroke dynamics based on gender recognition. Idrus et al. [4] also introduce more valuable information such as the type of hand used, age, and the dominant hand. These extra amounts of information can be used as reference in order to help to improve the performance of algorithms when the algorithms are applied into keystroke dynamics. Furthermore, Syed et al. [8] show the concept of event sequences used in the keystroke dynamics. This event sequences help to distinguish the typing behavior of a user.

The next section describes sequence alignment algorithm. Section 3 discusses the proposed method. Sections 4 and 5 explain the experimental method and the results of the experiment, respectively. The final section presents several concluding remarks and future research issues.

2. Sequence Alignment Algorithm

Sequence alignment is an algorithm that calculates the similarity among two or more sequences [17]. This algorithm is widely used in bioinformatics areas such as deoxyribonucleic acid (DNA) sequences, ribonucleic acid (RNA) sequences, or protein sequences. In Revett's [7] research, he has applied the sequence alignment algorithm into the keystroke dynamics and obtained encouraging results. In this paper, we show the performance comparison of our proposed method, Revett's sequence alignment algorithm, and other previous work.

The keystroke dynamics is generated in timestamp format (millisecond). Since values in timestamp format vary and can be infinity, it is inappropriate to apply keystroke dynamics to sequence alignment algorithm directly. Therefore, we have to discretize the timestamp into subintervals. Each subinterval will represent a different category. For example, this process is similar to a questionnaire construction. We usually allow a user to choose few options, such as "strongly disagree," "disagree," "neither disagree nor agree," "agree," and "strongly agree." Sometimes, we also just make it shorter to three options which are "disagree," "neither disagree nor agree," and "agree." But usually we just put maximum options to five or six options. We do not put too many categories into the questionnaire because it is hard to be analyzed later. Revett [7] has used twenty categories (i.e., twenty bins) in his research. These twenty categories are extracted from the letters of amino acid. These letters represent "ACDEFGHIKLMNPQRSTVWY." With the use of twenty bins in

keystroke dynamics, it becomes much suitable for sequence alignment algorithm to be used.

We explain the algorithm design of sequence alignment in the following paragraphs. Firstly, we have to get the difference of the time interval from a feature, for instance, dwell time. This time interval difference is obtained from the difference between maximum time and minimum time. The maximum time of dwell time means the longest time for a user to press a key and release a key. The minimum time of dwell time, on the other hand, is the shortest time for a user to press a key and release a key. The formula is defined by

$$\text{diff}_j = \max_j - \min_j, \tag{1}$$

where j is the number of the column (attribute).

After we obtain the difference of the time interval from an attribute, we have to divide the difference of time intervals into twenty subintervals. The length of subintervals is defined by

$$\text{rangeDiff}_j = \frac{\text{diff}_j}{20}, \tag{2}$$

where j is the number of the column. The reason why we have to get the rangeDiff is because we need to know the length of each category. If a new dwell time is close to the minimum time of dwell time, it will be categorized as letter A. The next category is C, followed by D, and so forth. If a new dwell time is close to the maximum dwell time, then it will be categorized as Y. At the end of calculation, each category has the exact same length, rangeDiff, because static divisor is used. Labeling (mapping) a new time can be formulated by

$$\text{label}_{i,j} = \left\lceil \frac{\text{new\_time}_{i,j} - \min_j}{\text{rangeDiff}_j} \right\rceil, \tag{3}$$

where i is the number of the row (also known as entry) and j is the number of the column. If a new time of a feature is less than the minimum time of the feature or greater than the maximum time of the feature (it mostly happens in the testing phase), it will be categorized as a different alphabet letter and is excluded in the alignment. Equations (1), (2), and (3) are the equations used to map the keystrokes to proper alphabet letters.

Once a row of data (entry) is changed to the corresponding alphabet letters, we run the sequence alignment algorithm. One point is scored if the label is matched in an attribute. Otherwise, no points are scored. The score is described as

$$\text{Score}_j \begin{cases} 1, & \text{if it is matched,} \\ 0, & \text{if it is not matched,} \end{cases} \tag{4}$$

where j is the number of the column. When an entry is completely verified, we sum all the scores. It is then defined by

$$\text{Final\_Score}_i = \sum_{j=1}^{N} \text{Score}_j, \tag{5}$$

where i is the number of the row, j is the number of the column, and N is the total number of the columns. This final score is then compared with the user-specified threshold. The higher the final score is, the higher the possibility that the new data is recognized as the genuine user is. In order to present a clear description for the whole algorithm design, we display the pseudocode of this algorithm in Algorithm 1.

In Algorithm 1, we can see that there are training phase and testing phase. In the training phase, the algorithm starts by calculating the maximum and the minimum for each attribute. It calculates the range difference (rangeDiff) for each attribute. Finally, it calculates labels to construct models.

In the testing phase, the algorithm converts the given testing data to alphabet letter format based on the minimum point obtained at step 3 and the range difference from step 7 in Algorithm 1. After that, it executes the conversion loop described from steps 9 to 14 for once on the test data. This is the conversion process described in step 1 in the testing phase.

After the conversion, for the actual recognition, for each row in the model, the algorithm calculates the score (Final_Score) for the row and the test data. This score is a summation of match scores (score in Algorithm 1) between the attribute in the row of the model and the attribute in the test data. Finally, the statistical summary of information including maximum, minimum, median, mean, and mode of the scores is calculated for the comparison with corresponding thresholds.

3. Sequence Alignment with Dynamic Divisor Generation Algorithm

In this paper, we propose sequence alignment with dynamic divisor generation (SADD) algorithm. SADD checks the degree of sufficiency of the dataset and then provides a proper divisor instead of static divisor as shown in (2) for each attribute. We show the steps to check the degree of sufficiency of the dataset and the reason we used dynamic divisor instead of static divisor in next paragraphs.

As a human, we are unfamiliar with a new thing immediately from the beginning. We have to practice a few times to get accustomed to the new thing. For example, consider an athlete who wants to run a 100-meter track in 10 seconds. However, this is nearly impossible if she is a beginner. She has to train hard and practice regularly. The time record from the first day until the day she manages to run a 100-meter track in 10 seconds could be illustrated as a graph which is shown in Figure 1. It is worth noting that, after few months of training, she will find it very difficult to reduce the time (i.e., less than 10 seconds) to finish the 100-meter track. This is because there is a limitation point of where we can reach even despite how hard we have trained and practiced. We called this phenomenon "realm point." Realm point refers to the temporal or spatial point when the user is accustomed to something or the user has a habit on something. This is also a reason why the world record of 100 meters currently remained at 9.58 seconds.

The phenomenon (i.e., realm point) that we have discussed previously can be applied for most activities including

Input: Training data extracted from a genuine user, A. Testing data extracted from a genuine user or an imposter, B.
In the training phase:
(1) **for** each attribute j **do**
(2) $\max[j] \leftarrow \max(\max[j], A_j)$
(3) $\min[j] \leftarrow \min(\min[j], A_j)$
(4) **end for**
(5) **for** each attribute j **do**
(6) $\text{diff} \leftarrow \max[j] - \min[j]$
(7) $\text{rangeDiff}[j] \leftarrow \text{diff}/20$
(8) **end for**
(9) **for** each row i and each attribute j in A **do**
(10) $\text{label} \leftarrow \text{ceil}((A_{i,j} - \min[j])/\text{rangeDiff}[j])$
(11) **if** label = 0
(12) label ← 1 // If the data is exactly the same value as minimum time, then it should categorize as label A
(13) $\text{model}[i, j] \leftarrow \text{label}$ // We just keep it as 1, 2, 3, . . . , 20 to represent A, C, D, . . . , Y
(14) **end for**
In the testing phase:
(1) Convert the B to alphabet letter format based on the minimum point and the range difference from step 3 and step 7 respectively, from training phase, and then we just run one time from step 9 to step 14 from training phase.
(2) **for** each row i in model **do**
(3) **for** each attribute j in B **do**
(4) **if** $B[j] = \text{model}[i, j]$
(5) $\text{Score}[j] \leftarrow 1$
(6) **end for**
(7) $\text{Final\_Score}[i] \leftarrow \text{sum}(\text{Score})$
(8) **end for**
(9) $\text{Checking\_Score} \leftarrow \text{mean}(\text{Final\_Score})$ // Can be max, min, median, mean, and mode
(10) return Checking_Score
Output: The score then used to compare the threshold

Algorithm 1: The pseudocode to explain the sequence alignment algorithm.

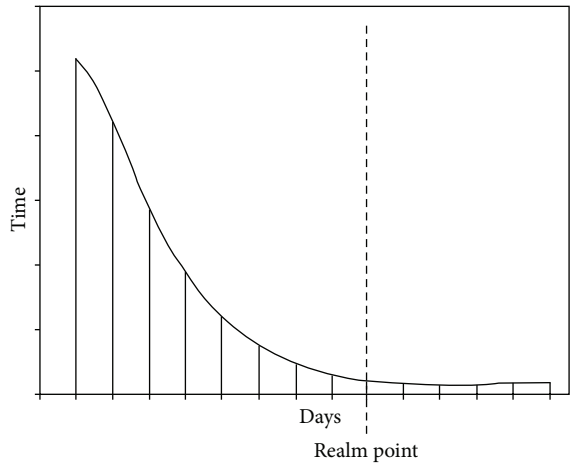

Figure 1: The time of practicing a new action versus the day of practicing it.

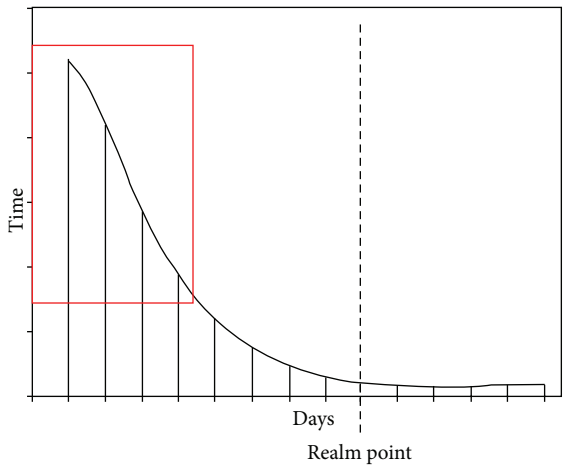

Figure 2: The worst case of the data used in the experiment or in real authentication system.

the typing speed of a password. Unfortunately, it is difficult to know how many hours, days, weeks, months, or years needed to reach the realm point of typing speed. Every user will have different time to reach realm point even with all the other conditions fixed. We do not know if the users are reaching the realm point or not in the beginning of the experiment, and real authentication systems do not know this either. For the best case, the data we collected cover from the beginning of the day to the realm point (or after realm point). For the worst case, it can be from the beginning of the day until the middle of the days, as shown in Figure 2.

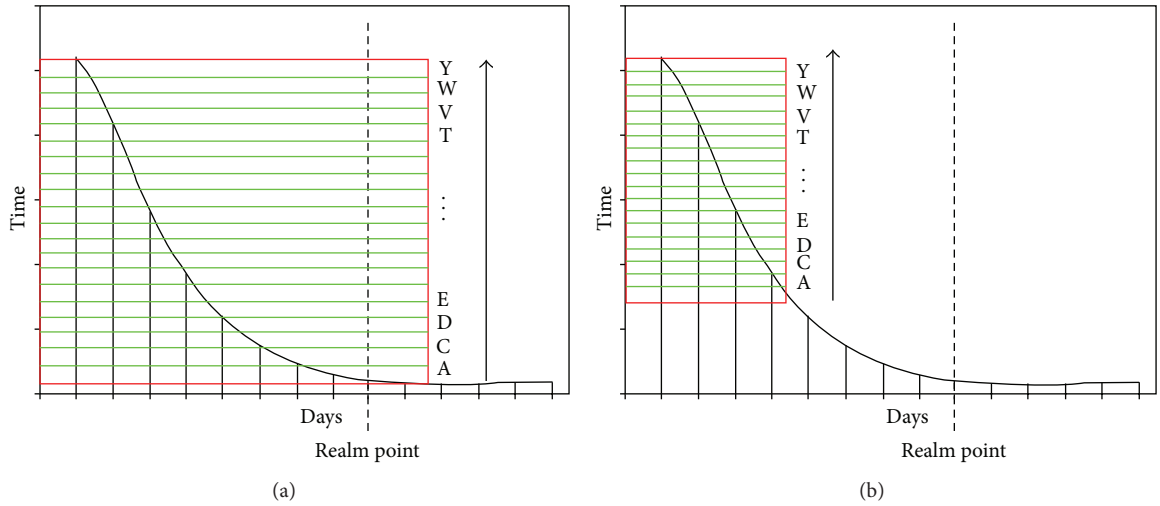

FIGURE 3: (a) The best case after the data is converted into a sequence. (b) The worst case after the data is converted into a sequence.

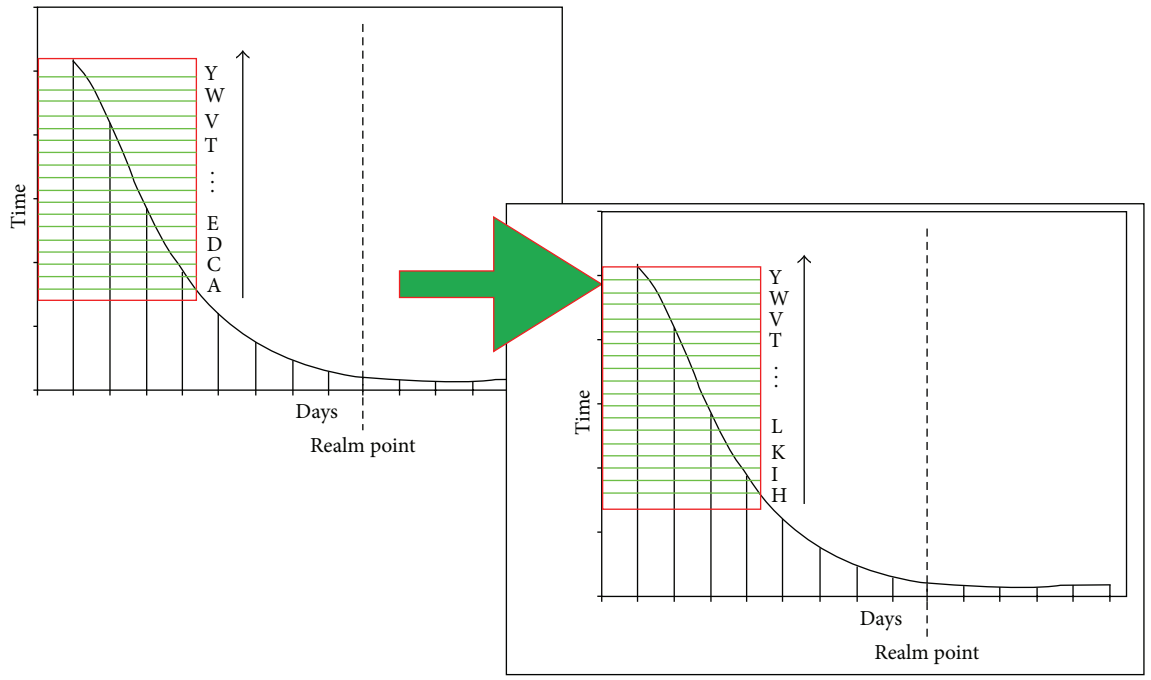

FIGURE 4: Data mapping with the amino acid letters in the worst case. Note that mapping is started from the middle of the letters.

For (1) and (2), we have to find each category in each subinterval. Figures 3(a) and 3(b) present the graph when the best case and worst case are applied with (1) and (2). Note that the intervals highlighted inside the red rectangle box are the ones mapped to the alphabet letters and are stored as a model in the database. This constructed model will be used to verify the new data. In particular, for the worst case, it can be seen that the categories generated do not cover whole information. Hence, if the genuine user gets accustomed to the password typing, then the minimum time of the new data she has inserted is lower than the minimum time in the database. Thus, in turn, this new data has high probability to be rejected as the imposter because it is out of category. In order to avoid this problem, we propose SADD in this paper. We use a subset of alphabet letters (e.g., 15 bins) and apply it to the dataset if its contents are insufficient. The idea is illustrated in Figure 4. We point out this problem to ensure that the model that we used in the database is always faultless and still can be used as a rewind case like when a user is sick and her typing speed becomes slower than usual but her typing behavior is still the same.

Now, we explain the proposed algorithm design. Firstly, (1) remains the same. The main procedure of SADD is to find the correct and suitable divisor used in (2). To find a proper divisor in each attribute, we exploit Horner's rule [18] to get the mean. Figure 5 shows the illustration of the way we use

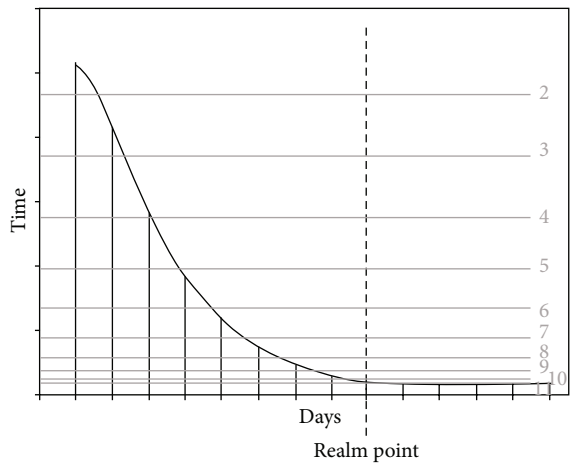

Figure 5: Calculation of mean with Horner's rule.

Horner's rule to calculate the mean. The grey line stated with 2 (known as line 2) on the right hand side is the mean of the first point (maximum point) and the second point. Line 3 is the mean of line 2 and third point. Line 4, however, is the mean of line 3 and the fourth point. The calculation is repeated by summing the previous output and the new data and then halving the sum. The repetition is terminated if it is reached to the end of the data. As shown in Figure 5, line 11 is the mean of line 10 and eleventh point. We also can observe that line 11 is closed to the realm point. By obtaining line 11, we can conclude that this user has reached to the realm point. The calculation for mean of each attribute by using Horner's rule is defined by

$$\text{mean}_j = \frac{((((((x_1 + x_2)/2) + x_3)/2) + \cdots)/\cdots) + x_i}{2}, \tag{6}$$

where i is the number of the row and j is the number of the column.

From line 11 in Figure 5, we observe that the calculated mean is near to the minimum point (i.e., realm point) and is far from the maximum point (i.e., beginning point). To get the proper divisor, we have to find the ratio of the mean from minimum point and the maximum point to the mean. The formula is described as

$$\text{ratio}_j = \frac{\left(\text{mean\_HR}_j - \min_j\right)}{\left(\max_j - \min_j\right)}, \tag{7}$$

where j is the number of the column and $0.5 \leq \text{ratio}_j < 1$.

Once we obtain the ratio, we calculate the divisor by

$$\text{divisor}_j = I - \left(I * \text{ratio}_j\right), \tag{8}$$

where j is the number of the column, I is 20 for twenty bins, and $10 \leq \text{divisor}_j \leq 20$.

After that, we replace the twenty values from (2) to this new divisor. It is calculated as

$$\text{rangeDiff}_j = \frac{\text{diff}_j}{\text{divisor}_j}, \tag{9}$$

where j is the number of the column.

Consider the worst case shown in Figure 3(b). Note that the first letter should not be labeled as "A" but a certain letter in the middle of the alphabet. Therefore, we modify (3) as follows:

$$\text{label}_{i,j} = \left\lceil \frac{\text{new\_time}_{i,j} - \min_j}{\text{rangeDiff}_j} \right\rceil + \left(I - \text{divisor}_j\right), \tag{10}$$

where i is the number of the row, j is the number of the column, and I is for the twenty bins. This means that the new data which is closer to the maximum point will be classified as "Y" and followed by "W," "V," "T," and so forth. The calculation of score and the final score remains the same as (4) and (5). In order to clearly explain our proposed algorithm, we provide Algorithm 2.

In the training phase of Algorithm 2, we calculate the mean by Horner's rule in steps 5 to 9. We calculate ratio in (7) at step 11, divisor in (8) at step 12, and range difference in (9) at step 13. Label assignment described in (10) is implemented at step 16 of Algorithm 2.

4. Experimental Method

4.1. Training and Testing Phase. In the keystroke dynamics, there are six common timing elements that we can use between two different keystrokes. They are as follows.

(i) Hold (H): it is a duration time (or a dwell time) of pressing a key:

  (a) a holding time of the first key, H1;
  (b) a holding time of the second key, H2.

(ii) Up-down (UD): it is a duration time (or a flight time) between key-up of the first key and key-down of the second key.

(iii) Down-down (DD): it is a duration time between key-down of the first key and key-down of the second key; it is the sum of H1 and UD.

(iv) Up-up (UU): it is a duration time between key-up of the first key and key-up of the second key; it is the sum of UD and H2.

(v) Down-up (DU): it is a duration time between key-down of the first key and key-up of the second key; it is the sum of DD and H2.

In our experiment, we use the CMU benchmark dataset [5]. This dataset consists of four elements, which are H1, H2, UD, and DD. However, we doubt that the DD element is not an appropriate element to be used in sequence alignment or similar algorithms. This is because sequence alignment algorithm is comparing attribute by attribute. In this case,

Input: Training data extracted from a genuine user, A. Testing data extracted from a genuine user or an imposter, B.
In the training phase:
(1) **for** each attribute j **do**
(2) $\max[j] \leftarrow \max(\max[j], A_j)$
(3) $\min[j] \leftarrow \min(\min[j], A_j)$
(4) **end for**
(5) **for** each attribute j **do**
(6) $\text{mean} \leftarrow A_1$
(7) **for** each row $i + 1$ **do**// Start from second row
(8) $\text{mean} \leftarrow (\text{mean} + A_i)/2$
(9) **end for**
(10) $\text{diff} \leftarrow \max[j] - \min[j]$
(11) $\text{ratio} \leftarrow (\text{mean} - \min[j])/\text{diff}$
(12) $\text{divisor}[j] \leftarrow 20 - (20 * \text{ratio})$
(13) $\text{rangeDiff}[j] \leftarrow (\text{diff}/\text{divisor}[j])$
(14) **end for**
(15) **for** each row i and each attribute j in A **do**
(16) $\text{label} \leftarrow \text{ceil}((A_{i,j} - \min[j])/\text{rangeDiff}[j]) + (20 - \text{divisor}[j])$
(17) **if** label = 0
(18) $\text{label} \leftarrow 1$ // If the data is exactly the same value as the minimum time, then it should be categorized as label A
(19) $\text{model}[i, j] \leftarrow \text{label}$ // We just preserve it as $1, 2, 3, \ldots, 20$ to represent A, C, D, ..., Y
(20) **end for**
In the testing phase:
(1) Convert the B to the alphabet letter format based on the minimum point the rane difference from step 3 and step 7 respectively, from training phase. And then we run from step 15 to step 20 for just once time.
(2) **for** each row i in model **do**
(3) **for** each attribute j in B **do**
(4) **if** $B[j] = \text{model}[i, j]$
(5) $\text{Score}[j] \leftarrow 1$
(6) **end for**
(7) $\text{Final\_Score}[i] \leftarrow \text{sum}(\text{Score})$
(8) **end for**
(9) $\text{Checking\_Score} \leftarrow \text{mean}(\text{Final\_Score})$ // Can be max, min, median, mean, and mode
(10) return Checking_Score
Output: The score then used to compare the threshold

Algorithm 2: The pseudocode of sequence alignment with dynamic divisor generation (SADD) algorithm.

it is checking element by element. Since DD is the sum of H1 and UD, there is a possibility to obtain the same value of DD with different value of H1 and UD. For instance, consider the following example with two different instances as shown in Figure 7.

Assume that Instance number 1 is the data that we have collected from a genuine user and it is used as model and Instance number 2 is a new data inserted by an imposter. Since it is a short example, we omit the procedure to convert the timestamp format into consensus sequence (label format). As we explain above, the sequence alignment algorithm is checking element by element. The first element it checks is H1. Based on (4), since the H1 of Instance number 2 is mismatched with the H1 of Instance number 1, the zero score is given. Next, the algorithm checks the UD element. They are mismatched too, and so zero score is given. Finally, the algorithm checks the last element, DD. Since they are matched to each other, one score is given. If the threshold given from a system is one, then Instance number 2 will be predicted as a genuine user. Otherwise, it will be rejected as an imposter's. If the system predicts Instance number 2 as a genuine user, then it will cause a lot of loss to the genuine user. Besides that, the example we show is a short example. However, in the real life, there can be a considerable amount of DD elements. If it is too fortunate for an imposter whose entire DD elements are matched (due to DD element being the sum of H1 and UD) with some entries of genuine users in the stored model and the threshold given is the total number of DD elements, then the system will accept this instance as a genuine user. Then, the imposter can access the system.

Therefore, in order to discover the effectiveness of DD elements in the authentication, we create another dataset from benchmark dataset which is having no DD elements (by removing all DD attributes from original dataset). Other than that, we also want to evaluate how effective it is to use all elements in the authentication. Hence, we create another dataset from benchmark dataset which is having extra UU and DU elements (by adding UU and DU attributes into original dataset). Basically, the difference between these three datasets is the number of the attributes used in the experiment. The first dataset (Dataset number 1) consists of 31 attributes, the second dataset (Dataset number 2) consists

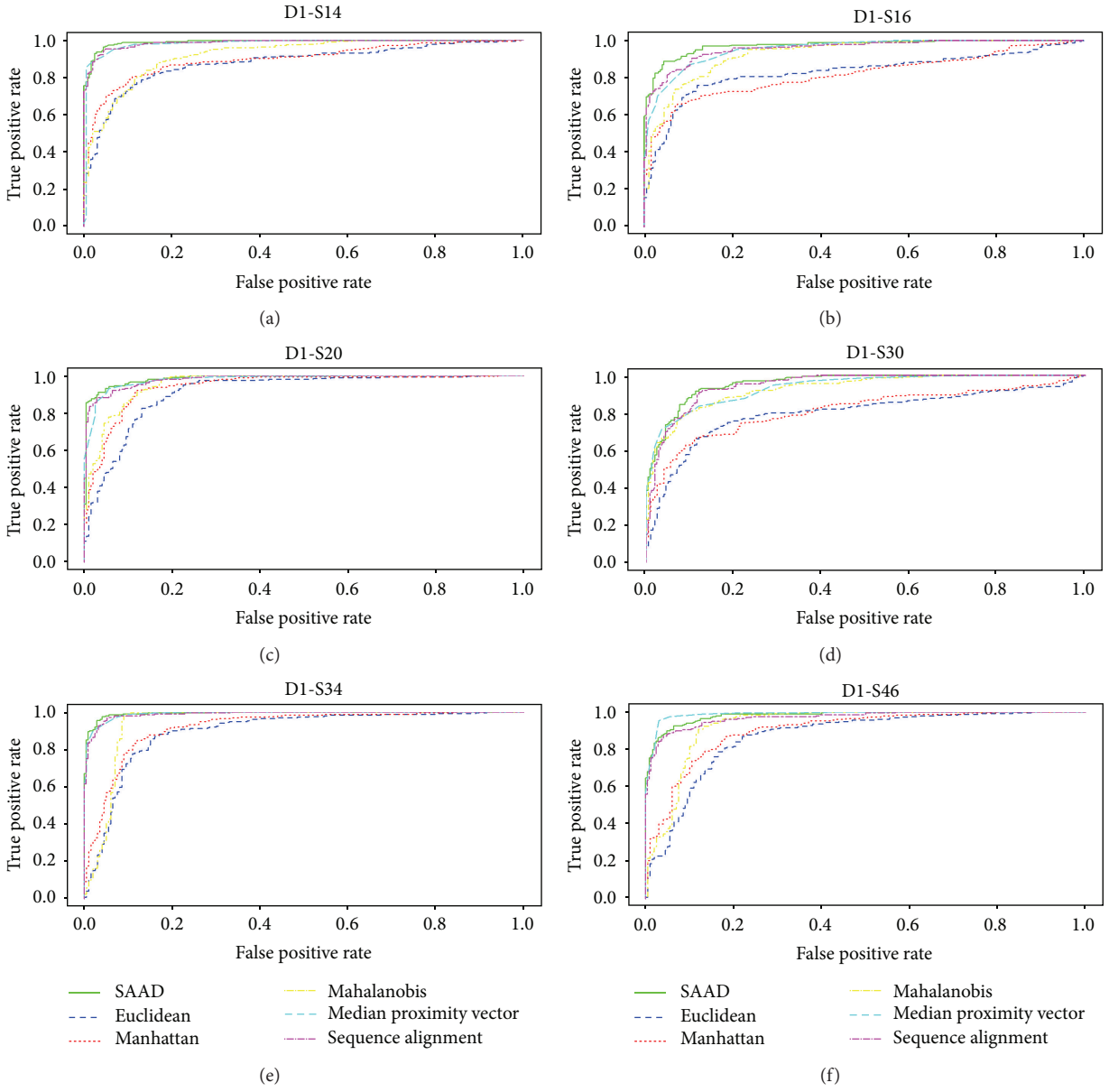

Figure 6: Examples of ROC curves from any 6 subjects in dataset number 1. Note that D stands for dataset and S stands for subject.

Instance#1: H1: 10 ms, UD: 14 ms, DD: 24 ms

Instance#2: H1: 12 ms, UD: 12 ms, DD: 24 ms

Figure 7: Instance number 1 is the data collected from a genuine user and stored as a model in database and Instance number 2 is the data collected from an imposter.

of 21 attributes, and the last dataset (Dataset number 3) has 51 attributes.

As for the benchmark dataset and two extra datasets we have created, each dataset contains 20,400 typing password entries. In each dataset, it consists of 51 subjects involved in this experiment. Each subject has 400 entries. In our experiment, during the training phase, we select one subject as genuine user and the remaining 50 subjects as imposters. First 200 entries of the chosen subject are used as training data. However, the remaining 200 entries of the chosen subject are used as testing data. Besides that, we obtain the first five entries from the remaining 50 subjects as testing data too. In total, there are 450 entries used in the testing phase. Based on Killourhy and Maxion [5] research, the reason they use the top five from the remaining 50 subjects is because they assume the imposter is unfamiliar to the password. In our experiment, we will operate at least 51 times in each dataset, and thus the number of total runs for three datasets will be 153 runs.

4.2. Performance of Evaluation Using ROC Curves. The main performance of evaluation in our experiment is using receiver operating characteristic (ROC) curves. We compare our algorithm and other approaches by showing the graph in

Table 1: The average of equal error rates with the standard deviation of equal error rate (after plus-minus sign) for six algorithms in three datasets. The highest performance of the algorithm is in bold.

Algorithm	EER		
	Dataset number 1	Dataset number 2	Dataset number 3
SADD	**0.076 ± 0.060**	**0.078 ± 0.060**	0.082 ± 0.062
Sequence alignment	0.084 ± 0.061	0.079 ± 0.053	0.087 ± 0.061
Median vector proximity	0.080 ± 0.060	0.094 ± 0.069	**0.081 ± 0.060**
Mahalanobis	0.110 ± 0.065	0.110 ± 0.065	0.110 ± 0.065
Manhattan	0.153 ± 0.092	0.144 ± 0.090	0.154 ± 0.089
Euclidean	0.171 ± 0.095	0.170 ± 0.097	0.169 ± 0.093

ROC curves. One of the examples of ROC curves is shown in Figure 6. The label at y-axis is true positive rate. True positive rate is the rate when an imposter is detected. And the x-axis labels the false positive rate, which is the rate when a genuine user is detected as an imposter. We can calculate the equal error rate (EER) from the ROC curve. EER is the point that is intersected by diagonal line and the curve line. We can conclude that an algorithm's performance is higher than others if the curve line is closer to the 1.0 point y-axis or the area under the curve (AUC) is larger than other algorithms.

5. Experimental Results

As aforementioned, we run three different datasets in our experiment. The first dataset is the CMU benchmark dataset, the second dataset is without the DD element, and the last dataset is with extra UU and DU elements. Table 1 shows the performance of our algorithm (SADD), sequence alignment, median vector proximity [6], Mahalanobis distance, Manhattan distance, and Euclidean distance. Table 1 has shown that our proposed algorithm produced higher performance than other previous work. Although the results for Dataset number 3 are not the best among the six algorithms, our algorithm, SADD, shows only 0.001 difference, in terms of EER, compared with the median vector proximity algorithm. Based on the results, it can be seen that our algorithm is comparable or outperforms in any case, either with or without some elements (i.e., UU, DD, UD, and DU elements).

In order to observe the performance of each algorithm in each subject, we produce the area under curve (AUC) in Tables 2, 3, and 4 with Dataset number 1, Dataset number 2, and Dataset number 3, respectively. If the total of AUC is close to value 1, then it means that the classification of that particular algorithm is desirable. These results are related with those in Table 1. We can view from Tables 2 and 3 that SADD consists of highest value of AUC. Therefore, SADD has shown the highest performance with Dataset number 1 and Dataset number 2 in Table 1. However, Table 4 shows that median vector proximity has highest value of AUC, and this is the reason why median vector proximity has highest performance with Dataset number 3 in Table 1. From Table 3, it is worthwhile to note that although median vector proximity shows overall higher accuracy, SADD has more wins (i.e., 20 wins) than median vector proximity has (i.e., 19 wins) for Dataset number 3.

Besides step 9 in the testing phase from Algorithms 1 and 2, we comment that we can use the statistical summary information such as minimum, maximum, mean, median, or mode when we calculate the checking score. In order to show the effectiveness of the statistical metric in the experiment, we provide the results with different statistical metrics in the form of the average of the EER with its standard deviation in Table 5. Interestingly, the result with mean metric has the higher performance than other statistical metrics such as median or mode. Furthermore, our proposed algorithm shows the highest performance in three datasets.

In addition, we operate an experiment with different number of the bins (different number of the categories) to test the effectiveness of the number of bins toward our proposed algorithm and the sequence alignment algorithm with static divisor. We show the result in Table 6. The statistical metric that we used to operate in this experiment is mean. Table 6 indicates that 20 bins or 30 bins have almost the same performance. However, depending on the dataset, we have to use different number of bins in order to produce optimal performance. As the results shown in Table 6, Dataset number 1 and Dataset number 2 have to use 20 bins to perform the highest results for SADD algorithm, but 30 bins have to be used in Dataset number 3 for SADD algorithm in order to obtain the highest results. Meanwhile, 30 bins are much appropriate to be used with SA algorithm in Dataset number 1 and Dataset number 3 and 20 bins, on the other hand, are much suitable for Dataset number 2.

6. Related Work

In Revett's research [7], he shows that the sequence alignment algorithm has performed effectively when it is applied into the keystroke dynamics. He also provides the steps of applying sequence alignment in the keystroke dynamics. However, there is still much more to improve as we have described in Section 3. Therefore, we have proposed our method named sequence alignment with dynamic divisor generation (SADD) algorithm which checks the degree of sufficiency of the dataset and provides a proper divisor to each attribute. Our result proves that SADD has higher performance than the sequence alignment algorithm using static divisor in every attribute.

Al-Jarrah [6] has proposed the median proximity vector in his paper. He provides a very good performance of this algorithm by using median instead of mean in his experiment. In our proposed algorithm, we apply various statistical metrics in the experiments to find the most appropriate measure.

Giot and Rosenberger's [3] research introduces a new soft biometric for keystroke dynamics based on gender recognition. Besides that, the similar study from Idrus et al. [4] also introduces more valuable information such as the number of hands used (i.e., one hand or both hands),

Table 2: The result of AUC for six algorithms in Dataset number 1. The highest value of AUC result is in bold.

Subject	Algorithm					
	SADD	Sequence alignment	Median vector proximity	Mahalanobis	Manhattan	Euclidean
1	0.91294	**0.92024**	0.91255	0.84844	0.81886	0.79240
2	**0.97029**	0.96491	0.96794	0.88018	0.87374	0.86002
3	**0.98542**	0.97690	0.95917	0.96926	0.89572	0.84620
4	0.99472	0.99249	**0.99634**	0.97216	0.96588	0.93968
5	0.97759	0.97771	**0.97867**	0.96218	0.93860	0.95596
6	0.98896	0.98872	**0.98927**	0.97972	0.94190	0.92976
7	**0.99936**	0.99835	0.99800	0.98944	0.99448	0.99088
8	0.99395	0.99359	**0.99498**	0.96762	0.95878	0.95658
9	0.99918	0.99846	**0.99933**	0.97772	0.97190	0.93394
10	0.99970	**0.99978**	0.99311	0.96102	0.96404	0.94450
11	0.98905	**0.99086**	0.96766	0.93818	0.87728	0.85674
12	0.89676	0.85888	0.87452	**0.91010**	0.81641	0.82154
13	**0.99871**	0.99626	0.99809	0.99184	0.98328	0.97100
14	**0.98766**	0.98363	0.98269	0.92130	0.89508	0.87682
15	0.99442	**0.99460**	0.99430	0.97628	0.95916	0.93830
16	**0.97430**	0.95875	0.95498	0.93130	0.81956	0.83250
17	0.96347	0.96203	**0.96723**	0.94258	0.94638	0.93536
18	0.97977	0.97453	**0.99634**	0.98998	0.85226	0.63850
19	0.96628	0.96097	0.99002	**0.99272**	0.91926	0.86964
20	**0.98631**	0.98169	0.97971	0.95914	0.94242	0.91234
21	0.98708	0.97760	**0.98939**	0.97928	0.95504	0.94054
22	**0.99145**	0.98747	0.99111	0.96448	0.90074	0.85908
23	**0.99391**	0.98614	0.98325	0.98912	0.94938	0.91076
24	0.98912	0.98902	**0.99239**	0.96422	0.96178	0.92958
25	0.90524	0.89650	**0.96463**	0.91256	0.92642	0.88396
26	0.86154	0.85887	**0.87943**	0.84346	0.76234	0.73782
27	0.91178	**0.93407**	0.90287	0.79592	0.83082	0.84098
28	0.88261	0.87920	0.89901	**0.93832**	0.81372	0.71792
29	0.98055	**0.98507**	0.92300	0.89790	0.80930	0.76686
30	**0.95856**	0.94897	0.93700	0.93106	0.81390	0.80098
31	0.99604	0.99028	0.99899	**0.99964**	0.98006	0.96314
32	0.98061	**0.98122**	0.95802	0.94178	0.90500	0.87796
33	0.97399	0.96602	**0.97599**	0.95766	0.93116	0.90708
34	**0.99356**	0.98958	0.99164	0.94522	0.92062	0.89656
35	0.90270	0.86283	**0.92857**	0.87956	0.70992	0.61772
36	0.98176	0.97069	**0.98948**	0.93214	0.90228	0.87042
37	0.99873	0.99707	**0.99911**	0.96974	0.98362	0.96498
38	0.99488	0.98635	**0.99702**	0.99290	0.95578	0.89508
39	0.97015	0.95877	0.95811	**0.97590**	0.92448	0.89638
40	**0.96546**	0.96143	0.95883	0.93078	0.89306	0.90874
41	0.89218	**0.90912**	0.85444	0.80208	0.70720	0.76250
42	**0.99264**	0.99112	0.99190	0.97140	0.95162	0.93066
43	0.92320	0.91166	**0.98007**	0.93964	0.55174	0.52548
44	**0.98860**	0.98114	0.98077	0.95566	0.90678	0.89028
45	0.96613	0.96170	**0.97195**	0.94778	0.94792	0.95978
46	**0.99974**	0.99930	0.99831	0.99286	0.99016	0.97922
47	**0.99926**	0.99910	0.99904	0.99086	0.99130	0.97598
48	**0.97999**	0.96864	0.97639	0.95596	0.96432	0.96334
49	**0.99996**	0.99994	0.99900	0.99504	0.99894	0.99572
50	**0.99533**	0.99344	0.98778	0.94522	0.93800	0.93418
51	**0.98853**	0.98566	0.98125	0.95238	0.92136	0.93976
Total	**49.50412**	49.28132	49.43454	48.25168	46.03375	44.84610
Wins	**20**	8	18	5	0	0

Table 3: The result of AUC for six algorithms in Dataset number 2. The highest value of AUC result is in bold.

Subject	Algorithm					
	SADD	Sequence alignment	Median vector proximity	Mahalanobis	Manhattan	Euclidean
1	0.88624	**0.90732**	0.87691	0.84844	0.82584	0.79356
2	0.95560	**0.95577**	0.93895	0.88018	0.86560	0.85034
3	**0.98590**	0.97993	0.96152	0.96926	0.90524	0.85016
4	0.99222	0.99102	**0.99439**	0.97216	0.96684	0.93098
5	0.94865	**0.96462**	0.94441	0.96218	0.93528	0.95348
6	0.98901	**0.99115**	0.98706	0.97972	0.95880	0.93774
7	**0.99928**	0.99776	0.99636	0.98944	0.99494	0.98914
8	**0.99156**	0.98971	0.98889	0.96762	0.96574	0.96284
9	0.99926	0.99839	**0.99928**	0.97772	0.98964	0.95676
10	0.99645	**0.99884**	0.98651	0.96102	0.96532	0.94166
11	0.98800	**0.98809**	0.95766	0.93818	0.88172	0.85402
12	**0.91010**	0.87188	0.88252	0.91010	0.82332	0.81468
13	**0.99851**	0.99646	0.99836	0.99184	0.98646	0.97188
14	**0.98852**	0.98281	0.98131	0.92130	0.90544	0.88206
15	0.99350	**0.99477**	0.98994	0.97628	0.96714	0.93850
16	**0.98241**	0.97288	0.95340	0.93130	0.85616	0.84848
17	0.95874	**0.96397**	0.94673	0.94258	0.94676	0.93334
18	0.99080	0.98800	**0.99653**	0.98998	0.90300	0.68836
19	0.97484	0.97116	0.99244	**0.99272**	0.94462	0.87930
20	**0.98398**	0.98143	0.97688	0.95914	0.94462	0.91040
21	**0.99048**	0.98306	0.98878	0.97928	0.96214	0.94290
22	**0.98967**	0.98849	0.98830	0.96448	0.92518	0.86614
23	**0.99664**	0.98886	0.97813	0.98912	0.96066	0.91082
24	0.98587	**0.98621**	0.98557	0.96422	0.96584	0.93000
25	0.91843	0.90906	**0.95942**	0.91256	0.92510	0.86990
26	0.84544	0.85266	**0.85673**	0.84346	0.76024	0.73304
27	0.87100	**0.93043**	0.82663	0.79592	0.81164	0.83336
28	0.93937	0.93577	**0.93986**	0.93832	0.82262	0.6955
29	0.97335	**0.98495**	0.91598	0.89790	0.82654	0.77378
30	**0.94988**	0.94329	0.89873	0.93106	0.81544	0.78808
31	0.99886	0.99779	0.99926	**0.99964**	0.98600	0.96808
32	0.97698	**0.98168**	0.94639	0.94178	0.92584	0.88910
33	**0.98258**	0.97630	0.97367	0.95766	0.93922	0.90810
34	**0.99577**	0.99215	0.99290	0.94522	0.93410	0.90136
35	0.95729	0.92842	**0.96261**	0.87956	0.75252	0.61538
36	0.99026	0.98381	**0.99333**	0.93214	0.92598	0.87978
37	**0.99893**	0.99773	0.99888	0.96974	0.98948	0.96940
38	0.99652	0.99269	**0.99730**	0.99290	0.96702	0.90490
39	**0.97834**	0.97180	0.95770	0.97590	0.94078	0.90424
40	**0.97330**	0.96871	0.94501	0.93078	0.91412	0.91472
41	0.83326	**0.88258**	0.78742	0.80208	0.68962	0.74806
42	**0.99272**	0.99254	0.99048	0.97140	0.95918	0.93112
43	0.95977	0.94853	**0.98591**	0.93964	0.60102	0.52690
44	**0.98470**	0.97880	0.97269	0.95566	0.92728	0.89918
45	0.94854	0.95723	0.94442	0.94778	0.94986	**0.95918**
46	**0.99973**	0.99959	0.99874	0.99286	0.99284	0.98012
47	**0.99938**	0.99902	0.99711	0.99086	0.99500	0.97770
48	0.96373	0.96001	0.94336	0.95596	**0.96554**	0.96292
49	**0.99758**	0.99742	0.99736	0.99504	0.99928	0.99722
50	**0.99167**	0.98757	0.97627	0.94522	0.94132	0.93674
51	0.98904	**0.98974**	0.97148	0.95238	0.92646	0.93746
Total	**49.48265**	49.47285	49.02047	48.25168	46.53034	44.94286
Wins	**23**	14	10	2	1	1

Table 4: The result of AUC for six algorithms in Dataset number 3. The highest value of AUC result is in bold.

Subject	Algorithm					
	SADD	Sequence alignment	Median vector proximity	Mahalanobis	Manhattan	Euclidean
1	0.91878	0.91561	**0.92803**	0.84844	0.82134	0.79844
2	0.97017	0.96301	**0.97068**	0.88018	0.88380	0.85948
3	**0.98415**	0.97933	0.95722	0.96926	0.90408	0.85802
4	0.99602	0.99316	**0.99697**	0.97216	0.96790	0.94638
5	0.98845	0.98843	**0.99180**	0.96218	0.95524	0.96498
6	**0.99111**	0.98988	0.99043	0.97972	0.92956	0.92516
7	**0.99928**	0.99884	0.99874	0.98944	0.99544	0.99210
8	0.99434	0.99422	**0.99510**	0.96762	0.95000	0.94710
9	**0.99877**	0.99693	0.99721	0.97772	0.95844	0.92222
10	0.99970	**0.99974**	0.99689	0.96102	0.96640	0.94618
11	0.98862	**0.98864**	0.96965	0.93818	0.87660	0.86392
12	0.89050	0.86445	0.84490	**0.91010**	0.82128	0.83304
13	**0.99877**	0.99702	0.99727	0.99184	0.97992	0.96996
14	**0.98520**	0.98127	0.97807	0.92130	0.88548	0.87208
15	0.99006	0.99071	**0.99299**	0.97628	0.94972	0.93500
16	**0.95655**	0.94369	0.94444	0.93130	0.80648	0.83034
17	0.96680	0.96742	**0.97450**	0.94258	0.95020	0.93990
18	0.96225	0.95484	0.98715	**0.98998**	0.80096	0.60398
19	0.95516	0.95233	0.98263	**0.99272**	0.90200	0.86606
20	**0.98398**	0.98131	0.97650	0.95914	0.94202	0.91498
21	0.97955	0.97219	**0.98599**	0.97928	0.95442	0.94144
22	0.99033	0.98696	**0.99062**	0.96448	0.89646	0.86752
23	**0.99109**	0.98497	0.98092	0.98912	0.94574	0.91436
24	0.99158	0.99109	**0.99410**	0.96422	0.95720	0.92892
25	0.87958	0.87253	**0.96272**	0.91256	0.93030	0.89172
26	0.87085	0.87509	**0.88064**	0.84346	0.77044	0.74542
27	0.92324	**0.93092**	0.92630	0.79592	0.84516	0.84584
28	0.86118	0.87980	0.88506	**0.93832**	0.85035	0.77736
29	0.97665	**0.98056**	0.92037	0.89790	0.80344	0.76698
30	**0.95948**	0.95189	0.95176	0.93106	0.82536	0.81706
31	0.98530	0.97686	0.99673	**0.99964**	0.97476	0.95788
32	**0.97732**	0.97706	0.95275	0.94178	0.88974	0.87134
33	**0.97796**	0.97639	0.97776	0.95766	0.93554	0.91100
34	**0.99097**	0.98795	0.98872	0.94522	0.91766	0.90142
35	0.86485	0.84880	**0.88478**	0.87956	0.73278	0.64826
36	0.9760	0.96774	**0.97859**	0.93214	0.90458	0.88226
37	0.99892	0.99692	**0.99944**	0.96974	0.98150	0.96516
38	0.98589	0.97523	**0.99420**	0.99290	0.94534	0.88950
39	0.97196	0.96555	0.94897	**0.97590**	0.91350	0.89376
40	**0.96815**	0.96767	0.96202	0.93078	0.88326	0.90798
41	0.91268	**0.92235**	0.88391	0.80208	0.72702	0.77560
42	**0.99197**	0.99096	0.99086	0.97140	0.95082	0.93314
43	0.89182	0.89006	**0.96234**	0.93964	0.54252	0.54492
44	**0.98709**	0.98154	0.98108	0.95566	0.89562	0.88752
45	0.97411	0.97339	**0.98341**	0.94778	0.94900	0.96054
46	**0.99971**	0.99942	0.99838	0.99286	0.98820	0.97910
47	0.99952	**0.99957**	0.99873	0.99086	0.98826	0.97440
48	0.98405	0.97524	**0.98639**	0.95596	0.96146	0.96468
49	**0.99996**	0.99994	0.99991	0.99504	0.99734	0.99464
50	**0.99212**	0.99190	0.98780	0.94522	0.93488	0.92860
51	**0.98937**	0.98596	0.98243	0.95238	0.92398	0.94170
Total	49.36191	49.21733	**49.38885**	48.25168	45.96349	44.99934
Wins	**20**	6	19	6	0	0

Table 5: The average of equal error rates with the standard deviation of equal error rate (after plus-minus sign) by using different statistical metrics in both sequence alignment algorithm and SADD. The highest performance of the algorithm and statistical metric is in bold.

Statistical metric	SADD			Sequence alignment		
	Dataset number 1	Dataset number 2	Dataset number 3	Dataset number 1	Dataset number 2	Dataset number 3
Minimum	0.273 ± 0.092	0.335 ± 0.079	0.264 ± 0.096	0.400 ± 0.073	0.472 ± 0.037	0.325 ± 0.106
Maximum	0.083 ± 0.061	0.092 ± 0.059	0.084 ± 0.067	0.088 ± 0.061	0.098 ± 0.056	0.086 ± 0.065
Mean	**0.076 ± 0.060**	**0.078 ± 0.060**	**0.082 ± 0.062**	0.084 ± 0.061	0.079 ± 0.053	0.087 ± 0.061
Median	0.087 ± 0.061	0.095 ± 0.064	0.091 ± 0.063	0.102 ± 0.065	0.115 ± 0.062	0.099 ± 0.064
Mode	0.134 ± 0.076	0.135 ± 0.070	0.146 ± 0.073	0.157 ± 0.078	0.153 ± 0.070	0.160 ± 0.078

Table 6: The average of equal error rates with the standard deviation of equal error rate (after plus-minus sign) by using different number of bins in both sequence alignment algorithm and SADD. The highest performance of the algorithms in appropriate number of the bins is in bold.

Bins	Dataset number 1		Dataset number 2		Dataset number 3	
	SADD	SA	SADD	SA	SADD	SA
10	0.083 ± 0.070	0.089 ± 0.073	0.079 ± 0.066	0.077 ± 0.064	0.095 ± 0.079	0.101 ± 0.082
20	**0.076 ± 0.060**	0.084 ± 0.061	**0.078 ± 0.060**	**0.079 ± 0.053**	0.082 ± 0.062	0.087 ± 0.061
30	0.077 ± 0.059	**0.084 ± 0.059**	0.082 ± 0.068	0.083 ± 0.061	**0.076 ± 0.057**	**0.084 ± 0.060**
40	0.077 ± 0.060	0.086 ± 0.061	0.082 ± 0.067	0.087 ± 0.062	0.079 ± 0.059	0.084 ± 0.061
50	0.079 ± 0.059	0.086 ± 0.062	0.084 ± 0.068	0.087 ± 0.062	0.079 ± 0.060	0.085 ± 0.063

age, and the dominant hand (i.e., left or right). This extra information can be used as reference in order to help to improve the performance of algorithms. Although we do not include this extra information into our experiment due to the limitation information that we can obtain from the CMU benchmark dataset, we believe that this extra information will be beneficial in the future work.

Syed et al. [8] show the concept of event sequences used in the keystroke dynamics. These event sequences help in distinguishing the typing behavior of a user. Most keystroke dynamics use key-down and key-up without actual key values. However, introducing too many dimensions can cause the curse of dimensionality. Therefore, it is interesting to extend our algorithm to accommodate these event sequences in the future work.

7. Conclusion

In this paper, we have proposed sequence alignment with dynamic divisor generation (SADD) for user authentication by using the keystroke dynamics. Based on the experiments we have conducted, our algorithm produces promising results and also mostly outperforms other previous work. We also empirically show that the dynamic divisor generally outperforms static divisor. We believe that the dynamic divisor takes an important role in sequence alignment algorithm because it calculates the degree of sufficiency of the dataset (by using mean of Horner's rule) and then it provides faultless calculation for an appropriate divisor to be used in each attribute. These dynamic divisors help to prevent the genuine user's data digressed from the legal categories.

Based on Giot and Rosenberger's [3] research, they introduce a new soft biometric for keystroke dynamics based on gender recognition. They have done interesting work by introducing this new information in keystroke dynamics. Furthermore, Idrus et al. [4] also introduce more valuable information such as the type of hands used (i.e., one hand or both hands), age, and the dominant hand (i.e., left or right). However, they just study the effectiveness of using the information. Therefore, it will be interesting to apply this extra information to our future study. We also like to discover more valuable information besides the information we have discussed above.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (no. NRF-2013R1A1A2013401).

References

[1] J. C. Poss, D. Boye, and M. W. Mobley, "Biometric voice authentication," Patent and Trademark Office, Washington, DC, USA, U.S. Patent no. 7,386,448, 2008.

[2] S. Cho, C. Han, D. H. Han, and H.-I. Kim, "Web-based keystroke dynamics identity verification using neural network," *Journal of Organizational Computing and Electronic Commerce*, vol. 10, no. 4, pp. 295–307, 2000.

[3] R. Giot and C. Rosenberger, "A new soft biometric approach for keystroke dynamics based on gender recognition," *International Journal of Information Technology and Management*, vol. 11, no. 1-2, pp. 35–49, 2012.

[4] S. Z. S. Idrus, E. Cherrier, C. Rosenberger, and P. Bours, "Soft biometrics for keystroke dynamics," in *Image Analysis and Recognition*, vol. 7950 of *Lecture Notes in Computer Science*, pp. 11–18, Springer, Berlin, Germany, 2013.

[5] K. S. Killourhy and R. A. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics," in *Proceedings of the IEEE/IFIP International Conference on Dependable Systems & Networks (DSN '09)*, pp. 125–134, IEEE, July 2009.

[6] M. M. Al-Jarrah, "An anomaly detector for keystroke dynamics based on medians vector proximity," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 3, no. 6, pp. 988–993, 2012.

[7] K. Revett, "A bioinformatics based approach to user authentication via keystroke dynamics," *International Journal of Control, Automation and Systems*, vol. 7, no. 1, pp. 7–15, 2009.

[8] 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. 9–16, IEEE, January 2014.

[9] X. Wang, F. Guo, and J.-F. Ma, "User authentication via keystroke dynamics based on difference subspace and slope correlation degree," *Digital Signal Processing: A Review Journal*, vol. 22, no. 5, pp. 707–712, 2012.

[10] E. Yu and S. Cho, "Keystroke dynamics identity verification—its problems and practical solutions," *Computers and Security*, vol. 23, no. 5, pp. 428–440, 2004.

[11] R. Moskovitch, C. Feher, A. Messerman et al., "Identity theft, computers and behavioral biometrics," in *Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI '09)*, pp. 155–160, June 2009.

[12] A. F. M. N. H. Nahin, J. M. Alam, H. Mahmud, and K. Hasan, "Identifying emotion by keystroke dynamics and text pattern analysis," *Behaviour & Information Technology*, vol. 33, no. 9, pp. 987–996, 2014.

[13] N. B. Amor, S. Benferhat, and Z. Elouedi, "Naive Bayes vs decision trees in intrusion detection systems," in *Proceedings of the ACM Symposium on Applied Computing*, pp. 420–424, ACM, March 2004.

[14] R. Fabbri, L. D. F. Costa, J. C. Torelli, and O. M. Bruno, "2D Euclidean distance transform algorithms: a comparative survey," *ACM Computing Surveys*, vol. 40, no. 1, article 2, 2008.

[15] R. Koch, M. Golling, and G. D. Rodosek, "Behavior-based intrusion detection in encrypted environments," *IEEE Communications Magazine*, vol. 52, no. 7, pp. 124–131, 2014.

[16] P. O. Asagba and E. O. Nwachukwu, "RSA asymmetric cryptosystem beyond homogeneous transformation," *West African Journal of Industrial and Academic Research*, vol. 9, no. 1, pp. 3–12, 2014.

[17] D. W. Mount, *Bioinformatics: Sequence and Genome Analysis*, Cold Spring Harbour, Cold Spring Harbour Laboratory Press, 2nd edition, 2004.

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, 3rd edition, 2009.