Normalization Technique and Weight Adjustment Analysis for Keystroke Vector Dissimilarity Authentication

TANAPAT ANUSAS-AMORNKUL*, NAPHAT BUSSABONG
Department of Computer and Information Science,
King Mongkut's University of Technology North Bangkok,
1518 Pracharat 1 Road, Wongsawang, Bangsue, Bangkok 10800,
THAILAND

*Corresponding Author

Abstract: - A keystroke dynamics authentication uses keystroke rhythm for each user on a keyboard to verify a real user. The idea is that each user has a unique keystroke rhythm such that it can be determined the identity of a user. To verify a user, a keystroke vector dissimilarity technique was proposed to use keystroke features as a vector and calculate a weight using SoftMax+1 to overcome the Euclidean distance problem. However, the weight has yet to be analyzed in detail. Therefore, this paper aims to find a normalization technique and a weight adjustment to enhance the accuracy of the keystroke vector dissimilarity technique. The normalization techniques and activation functions analyzed in this study are Euclidean norm, Mean normalization, Min-max normalization, Z-score normalization, SoftMax function, and ReLU function. The weight adjustment varies from *w+1000* to *1000-w*. The results show that the Mean and Min-max normalizations with *10-w* as a weight gave the same results at 96.97% accuracy and 3.03% error, which are better than the previous work.

Key-Words: - Keystroke rhythm, keystroke dynamics authentication, keystroke vector dissimilarity, Euclidean distance, normalization technique analysis, weight adjustment analysis.

Received: March 8, 2024. Revised: July 2, 2024. Accepted: August 3, 2024. Published: September 23, 2024.

1 Introduction

A keystroke dynamics authentication is a biometric authentication using keystroke rhythm on a keyboard to verify a real user. Each user has their keystroke rhythm or keystroke dynamics, which can classify a real user from a fake user. There are many techniques to verify a keystroke rhythm. One example is to use a statistical method using a Euclidean distance, which was used to measure the keystroke distance between a real user and a fake user, [1]. However, the Euclidean distance does not consider a sequence of input keys such that a fake user can be a real user even if the keystroke is different. For example, a fake user may type faster than a real user in the beginning but slower in the end, but the Euclidean distance between the real user and the fake user is small, and it is determined as the real user. In [1], a keystroke vector dissimilarity was proposed to enhance the keystroke dynamics authentication (KDA) by using a keystroke vector and a weight to reduce a flaw in Euclidean distance—however, the weight needed to be thoroughly analyzed to improve the authentication performance. The contribution of this study extends the limited research on the normalization techniques and weight adjustments for the keystroke vector dissimilarity technique.

In this study, normalization techniques and weight adjustment are analyzed to improve the accuracy of [1]. A dataset from [1] is compared with the previous work. The normalization techniques and activation functions are Euclidean norm, Mean normalization, Min-max normalization, Z-score normalization, SoftMax function, and ReLU function, and a weight is adjusted from w+1000 to 1000-w. The objective of this paper is to obtain a normalization technique and a weight adjustment such that the accuracy of the keystroke vector dissimilarity technique is improved. Therefore, a fake user has less chance to pass the authentication process, and a system is safe from several mishaps, such as information leaks and modifications, and privilege escalation.

The organization of this paper is as follows. The next section presents the background of keystroke dynamics, normalization techniques, and activation functions. In addition, the literature review is also explained. Section 3 presents the overview of keystroke vector dissimilarity, which is used as a model to analyze the normalization techniques and weight adjustment. Performance metrics and datasets used in this study are presented. Next, the experimental results are shown and analyzed in detail with an example. The last section is the conclusion, which summarizes the study results and future work.

2 Background and Literature Review

This section presents background on keystroke dynamics features and normalized techniques, along with the literature reviews in this research area.

2.1 Background

Fundamental keystroke dynamics features are presented in this section to give basic concepts of standard keystroke dynamics features. Moreover, normalization techniques and activation functions are explained to provide basic knowledge on how to normalize data in the study.

2.1.1 Keystroke Dynamics Features

Keystroke dynamics is a keystroke rhythm of a user on a keyboard. Each user has a unique keystroke rhythm that can be used to verify the user's identity. To collect the keystroke dynamics, keystroke features are explained as follows.

- Interkey time (I) is the time duration from releasing one key to pressing a following key.
- Hold time (H) is the time duration from pressing a key to releasing that key.
- Latency (L) is a combination of Interkey time and Hold time or the time duration from pressing a key to pressing a following key.

The standard keystroke features are illustrated in Figure 1.

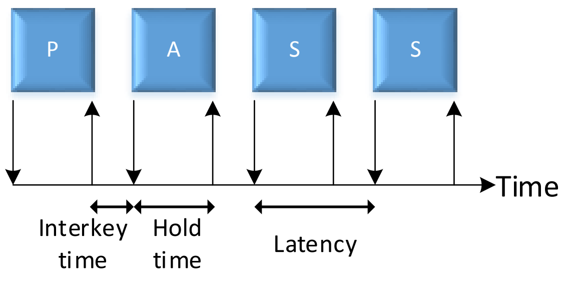

Fig. 1: Keystroke dynamics features

2.1.2 Normalization Techniques and Activation Functions

This study investigates four normalization techniques and two activation functions to enhance the keystroke dynamics authentication performance. The techniques and functions in this research are described in detail as follows.

2.1.2.1 Euclidean Norm

An important component of Euclidean norm, or L2 normalization, is the length or magnitude of a vector in mathematics using (1).

$$||V||_2 = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2} = \sqrt{\sum_{i=1}^{n} v_i^2} \tag{1}$$

where V is an n-dimension vector.

For the Euclidean norm, each value in the vector is divided by its length, as shown in (2).

$$\ell_{2-norm}(V) = \frac{v}{||v||_2} = \left(\frac{v_1}{||V||_2}, \frac{v_2}{||V||_2}, \frac{v_3}{||V||_2}, \ldots, \frac{v_n}{||V||_2}\right) \tag{2}$$

where n is the dimension of the vector. The Euclidean norm outputs are between 0 and 1.

2.1.2.2 Mean Normalization

Mean normalization is calculated using a mean as a reference for data scaling, as shown in (3).

$$\boldsymbol{x'} = \frac{\boldsymbol{x - \mu}}{\boldsymbol{max(x) - min(x)}} \tag{3}$$

where μ is a mean, max(x) is the maximum value in a vector x, and min(x) is the minimum value in a vector x. The outputs from the Mean normalization are between -1 and +1.

2.1.2.3 Min-max Normalization

Min-max normalization is used to scale data using minimum value as a reference, as shown in (4).

$$\boldsymbol{x'} = \frac{\boldsymbol{x - min(x)}}{\boldsymbol{max(x) - min(x)}} \tag{4}$$

The outputs from the Min-max normalization are between 0 and +1.

2.1.2.4 Z-score Normalization

Z-score normalization is applied to adjust data to exhibit a mean of 0 and a standard deviation of 1, as shown in (5).

$$\boldsymbol{x'} = \frac{\boldsymbol{x - \mu}}{\boldsymbol{\sigma}} \tag{5}$$

where σ is a standard deviation. The outputs from the Z-score normalization are between negative and positive values.

2.1.2.5 SoftMax function

SoftMax is a mathematical function commonly employed in Deep learning for classification. This function transforms data into probability values, as shown in (6).

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad (6)$$

where $\sigma(z)_i$ is a SoftMax function, z is an input vector, e^{zi} is a standard exponential function, K is a number of an input vector, and $\sum_{j=1}^{K} e^{z_j}$ is a summation of all standard exponential functions. The total sum of the SoftMax function outputs is 1.

2.1.2.6 ReLU (Rectified Linear Unit) function

ReLU is a mathematical function used in Deep learning for its simplicity with no negative value. It transforms data from 0 to maximum value, as shown in (7).

$$x'(\mathrm{V}) = max(0, v_i) \quad (7)$$

where V is an input vector, and v_i is an ith vector element.

2.2 Literature Review

Keystroke dynamics authentication research can be classified into two main research techniques: statistical and machine learning. The reviews for each technique are as follows.

2.2.1 Statistical Techniques

For keystroke dynamics authentication, statistical techniques were proposed because of less data usage. It means that the amount of collected keystroke data is small and practically useful for real-world applications. [1], proposed a Keystroke Vector Dissimilarity using a vector to represent keystroke dynamics features. The technique proposed to use a Master vector profile and an SD vector profile to verify a user. The SD vector profile served as a weight vector to assign the importance of each feature (or position in a vector) differently. The proposed work gave 96.67% accuracy and 3.33% EER.

[2], proposed using dynamic time warping for keystroke dynamics authentication with the CMU dataset. The result showed that the proposed technique gave a False Accept Rate (FAR) of 39.9%, False Reject Rate (FRR) of 3.3%, and EER of 17.6%. [3], proposed using the Generalized Fuzzy model (GFM), a combination of Mamdani-Larsen and Takagi-Sugeno fuzzy models for keystroke dynamics authentication. It was tested with the CMU dataset and the GFM model gave the best performance at 7.86% EER.

[4], proposed a simple statistical method using an average and a standard deviation to create a profile from crowdsourcing. The best results were obtained in 2 cases. The first case gave FAR at 0.0% and FRR at 2.54%. The second case gave FAR at 0.0% and FRR at 2.87%.

2.2.2 Machine Learning Techniques

Machine learning techniques for KDA are actively used in the research area. The techniques also gave high accuracy, but they require a large amount of data for training.

[5], proposed using Chinese keystroke dynamics for continuous authentication by selecting 5 timing features and using one-class SVM for authentication. The accuracy was 62.3%. [6], proposed to use the Artificial Bee Colony algorithm to classify a user using their dataset. The keystroke dynamics features in this work were pressure, dwell time, and flight time. The best accuracy was 90.63%, using Dwell time and pressure as the features.

[7], studied four machine learning models, i.e. KNN, SVC, Random Forest, and XGBoost using CMU dataset. The XGBoost gave the best performance at 93.59% accuracy. [8], created their application to collect keystroke data and tested with several classifier algorithms. The best algorithm was the tree-based algorithm, and the accuracy was 94%.

[9], studied five machine learning algorithms, i.e., Manhattan Distance, Manhattan Filtered Distance, Manhattan Scaled Distance, Gaussian Mixture Model, and Support Vector Machine, for biometric authentication with CMU dataset. The Manhattan Scaled Detector statistical algorithm was the best algorithm, which gave EER 11.76%. [10], proposed to transform behavioral biometrics (time series) into three-dimensional (3D) images to test 6 deep learning algorithms. The best algorithm was the GoogleNet model, which gave EER 4.49%.

Table 1. Literature work summary

Reference	Dataset	Acc.	EER	FAR	FRR
Stat [1]	own	96.67%	3.33%	3.33%	3.33%
Stat [2]	CMU Dataset	-	17.6%	39.9%	3.3%
Stat [3]	CMU Dataset	-	7.86%	-	-
Stat [4]	own	-	-	0%	2.54%
ML [5]	own	62.3%	-	-	-
ML [6]	own	90.63%	-	-	-
ML [7]	own	93.59%	-	-	-
ML [8]	own	94%	-	-	-
ML [9]	CMU Dataset	-	11.76%	-	-
ML [10]	own	-	4.49%	-	-

Note: Stat is a statistical technique, and ML is a machine learning technique.

Table 1 shows the summary of the literature works using accuracy, EER, FAR, and FRR as performance metrics to compare with other works. Some metrics cannot be determined in their papers. Datasets were from different sources, and each source collected keystroke dynamics features differently. The results are highly challenging to compare against one another. The highest accuracy is 96.67%, with 3.33% error in [1] with their own dataset. However, this study obtains the dataset from [1]. The results can be compared fairly with the same dataset.

3 Keystroke Vector Dissimilarity

The concept of the keystroke vector dissimilarity technique was proposed in [1]. However, this research aims to analyze the normalization techniques and weight adjustment of the keystroke vector dissimilarity technique. An overview of the keystroke vector dissimilarity is presented to explain how the technique works.

3.1 Overview

The keystroke vector dissimilarity was proposed to overcome a weakness of Euclidean distance by using a keystroke vector and a standard deviation (SD) vector as weights to verify a user. The overview diagram for the keystroke vector dissimilarity technique is shown in Figure 2.

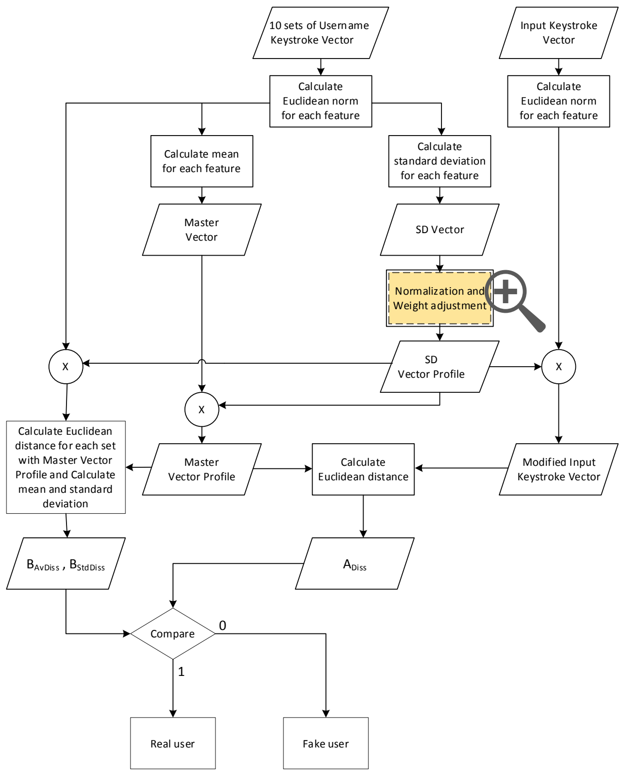
10 sets of Username Keystroke Vector
Input Keystroke Vector
Calculate Euclidean norm for each feature
Calculate Euclidean norm for each feature
Calculate mean for each feature
Calculate standard deviation for each feature
Master Vector
SD Vector
Normalization and Weight adjustment
SD Vector Profile
X
X
X
Calculate Euclidean distance for each set with Master Vector Profile and Calculate mean and standard deviation
Master Vector Profile
Calculate Euclidean distance
Modified Input Keystroke Vector
B_{AvDiss}, $B_{StdDiss}$
A_{Diss}
Compare
0
1
Real user
Fake user

Fig. 2: Keystroke vector dissimilarity diagram

This technique consisted of 5 main components, i.e., 1) a keystroke vector, 2) a vector normalization, 3) a master vector and SD vector profile, 4) a master vector profile and modified input keystroke vector, and 5) a verification process. A keystroke vector is created from keystroke dynamics features: Interkey time and Latency for each key to create a vector using a sequence of input keystrokes. A vector normalization was used to normalize the standard deviation for each feature in a vector, and it is used as a weight for each feature. This vector has to be adjusted so that weights are not zero.

A Master Vector is a vector created from a means of 10 real user inputs to create a master vector profile by multiplying it with an SD vector profile, which weighs each element in a vector to overcome the weakness of the Euclidean distance. When a user inputs his username on a keyboard, an input keystroke vector is created and calculated the Euclidean norm for each feature. Then, the vector is multiplied by the SD vector profile to create a modified input keystroke vector compared with a Master vector profile. A verification process is done using the Euclidean distance and a Six Sigma technique. If the distance between a new input vector and a Master vector profile is in the range of a real user, the verification is passed, and vice versa.

3.2 Performance Metrics

In this study, standard keystroke dynamics performance metrics are False Accept Rate (FAR), False Reject Rate (FRR), and Equal Error Rate (EER). Moreover, the machine learning performance metrics, i.e., Accuracy (Acc.) and Error (Err.), are used in this study when EER cannot be evaluated in some cases. Therefore, Accuracy is the main performance metric of this study for comparing results.

3.2.1 False Accept Rate (FAR)

False Accept Rate (FAR) is the percentage of fake user inputs accepted as a real user. The FAR formula is displayed in (8).

$$FAR = \frac{Number\ of\ Accepted\ Fake\ user}{Total\ Fake\ user\ Attempts} \times 100 \tag{8}$$

3.2.2 False Reject Rate (FRR)

False Reject Rate (FRR) is the percentage of real user inputs rejected as a fake user. The FRR formula is displayed in (9).

$$FRR = \frac{Number\ of\ Rejected\ Real\ User}{Total\ Real\ User\ Attempts} \times 100 \tag{9}$$

3.2.3 Equal Error Rate (EER)
The Equal Error Rate (EER) is a cross point between FAR and FRR: the lower the EER, the higher the accuracy of a biometric system.

3.2.4 Accuracy
Accuracy (Acc.) is a percentage of predictions that match the actual results. The accuracy formula is displayed in (10).

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ Number\ of\ predictions} \times 100 \qquad (10)$$

3.2.5 Error
Error (Err.) is a percentage of predictions that do not match the actual results. It is the opposite of Accuracy. The error formula is displayed in (11).

$$Error = 100 - Accuracy \qquad (11)$$

3.3 Dataset
In this study, a dataset used for the experiments is from [1]. The dataset was collected from 18 users, 15 real users, and 3 fake users. Each real user typed their username (not a password) on a keyboard in 2 sets. For each set, a user typed his username 10 times. The total data for real users is 300 records. A fake user typed the usernames of real users 30 times for each user. The total data for fake users is 1,350 records. The total data is 1,650 records in this dataset.

4 Normalization Technique and Weight Analysis
From the previous section, the overview of the keystroke vector dissimilarity technique is explained. Figure 2 shows the overall components of the technique, and the component that will be analyzed in this study is the normalization and weight adjustment component, as shown in Figure 3. The main reason to examine this component is that its function is to determine weights for creating an SD vector profile to solve the Euclidean distance problem. The previous research in [1] only proposed to use the SoftMax function as a normalization technique and SoftMax outputs (*w*) +1 as weights. The accuracy and EER were 96.67% and 3.33%, respectively.

This study aims to analyze other normalization techniques and weight adjustment values to improve the accuracy of the keystroke vector dissimilarity technique. However, the previous analysis of SD vector values found that the standard deviation value for each keystroke feature was very small. The SD vector must first be normalized so that the values can give significant weight to the keystroke vectors.

In this section, the normalization techniques and weight adjustment are under-investigated to study the performance and how to calculate a proper weight for the keystroke vector dissimilarity technique. The normalization techniques in this study are Euclidean norm, Mean normalization, Min-max normalization, Z-score normalization, SoftMax function, and ReLU function. It is important to note that the output from a normalization technique is represented as “w” in this study.

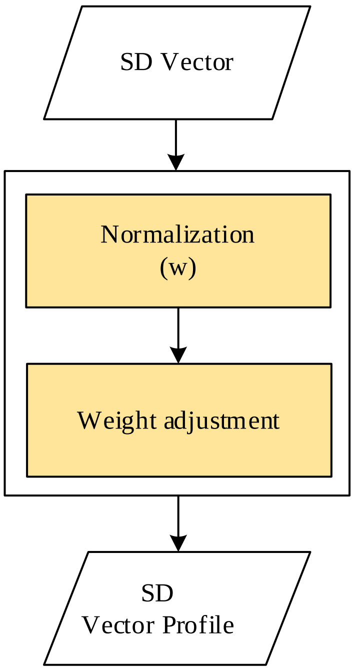

Fig. 3: Normalization and Weight adjustment.

In addition, weight adjustment is an important component, such that the weight is not zero when multiplying with the master profile or a modified input keystroke vector, and the effect of that element is still important for the verification process. The weight adjustment value can be added, subtracted, or multiplied to make a proper weight for creating an SD vector profile. Sometimes, the outputs from a normalization technique are small, such that multiplying by 10, 100, or 1000 is a reasonable weight adjustment, and the addition or subtraction by 1 is crucial to ensure that weights are not zero. Several weight adjustment values are investigated such as *w+1*, *w+1000*, *(1000w)+1*, *1-w*, *1000-w*, and *1-(1000w)*.

The overview of an SD vector as the input to a normalization technique is that all elements in the vector are standard deviations, always positive numbers. Each standard deviation value is calculated from 10 username inputs. The standard deviations for all elements in the SD vector are small. A component of the SD vector with a high standard deviation means that a user is unfamiliar with that feature, and the weight should be small. This hypothesis is shown later in this section.

Table 2, Table 3, Table 4, Table 5, Table 6 and Table 7 show the results from different normalization techniques and activation functions: Euclidean norm, Mean normalization, Min-max normalization, Z-score normalization, SoftMax function, and ReLU function, respectively. The weights are varied, and the performance metrics are accuracy (Acc.), error (Err.), FAR, FRR, and EER. In the Tables, some values of EER cannot be determined because the cross point between FAR and FRR was not found. The bold row in the Tables gives the best performance compared to other table rows.

The results from the Euclidean norm are presented in Table 2. The outputs from the Euclidean norm are ranging between 0 and 1. The best weight for the Euclidean norm is *10-w* with 96.91% accuracy and 3.09% error.

Table 2. Euclidean norm results

Weight Adjustment	Acc.	Err.	FAR	FRR	EER
w+1	95.33%	4.67%	4.67%	4.67%	-
w+10	96.67%	3.33%	3.33%	3.33%	3.33%
w+100	96.67%	3.33%	3.33%	3.33%	3.33%
w+1000	96.67%	3.33%	3.33%	3.33%	3.33%
(10*w)+1	91.09%	8.91%	8.89%	9.00%	-
(100*w)+1	90.00%	10.00%	10.00%	10.00%	10.00%
(1000*w)+1	90.06%	9.94%	9.93%	10.00%	-
1-w	96.67%	3.33%	3.33%	3.33%	3.33%
10-w	**96.91%**	**3.09%**	**3.11%**	**3.00%**	**-**
100-w	96.67%	3.33%	3.33%	3.33%	3.33%
1000-w	96.67%	3.33%	3.33%	3.33%	3.33%
1-(w*10)	88.06%	11.94%	11.93%	12.00%	-
1-(w*100)	89.94%	10.06%	10.07%	10.00%	-
1-(w*1000)	90.06%	9.94%	9.93%	10.00%	-

Table 3. Mean normalization results

Weight Adjustment	Acc.	Err.	FAR	FRR	EER
w+1	93.52%	6.49%	6.52%	6.33%	-
w+10	96.49%	3.52%	3.48%	3.67%	-
w+100	96.67%	3.33%	3.33%	3.33%	3.33%
w+1000	96.67%	3.33%	3.33%	3.33%	3.33%
(10*w)+1	87.39%	12.61%	12.59%	12.67%	-
(100*w)+1	86.24%	13.76%	13.78%	13.67%	-
(1000*w)+1	86.24%	13.76%	13.85%	13.33%	-
1-w	96.67%	3.33%	3.33%	3.33%	3.33%
10-w	**96.97%**	**3.03%**	**3.04%**	**3.00%**	**-**
100-w	96.73%	3.27%	3.26%	3.33%	-
1000-w	96.67%	3.33%	3.33%	3.33%	3.33%
1-(w*10)	87.70%	12.30%	12.30%	12.33%	-
1-(w*100)	86.18%	13.82%	13.85%	13.67%	-
1-(w*1000)	86.30%	13.70%	13.78%	13.33%	-

The outputs from Mean normalization range between -1 and +1, but the outputs from Min-max normalization range between 0 and 1. The best weight for Mean and Min-max normalizations is *10-w* with the same accuracy at 96.97% and error at 3.03%, as shown in Table 3 and Table 4. This is because both normalization techniques have the same denominator, but the numerators differ by changing from a mean to a minimum value. The overall results from Min-max normalization are better than that from Mean normalization.

Table 4. Min-max normalization results

Weight Adjustment	Acc.	Err.	FAR	FRR	EER
w+1	94.49%	5.52%	5.48%	5.67%	-
w+10	96.49%	3.52%	3.48%	3.67%	-
w+100	96.67%	3.33%	3.33%	3.33%	3.33%
w+1000	96.67%	3.33%	3.33%	3.33%	3.33%
(10*w)+1	90.55%	9.46%	9.48%	9.33%	-
(100*w)+1	90.06%	9.949%	9.93%	10.00%	-
(1000*w)+1	90.06%	9.94%	9.93%	10.00%	-
1-w	95.64%	4.36%	4.37%	4.33%	-
10-w	**96.97%**	**3.03%**	**3.04%**	**3.00%**	**-**
100-w	96.73%	3.27%	3.26%	3.33%	-
1000-w	96.67%	3.33%	3.33%	3.33%	3.33%
1-(w*10)	88.91%	11.09%	11.11%	11.00%	-
1-(w*100)	90.06%	9.94%	9.93%	10.00%	-
1-(w*1000)	90.06%	9.94%	9.93%	10.00%	-

For Z-score normalization, the outputs from this normalization range between negative and positive numbers, which can be less than -1 or more than 1. The weights can be large such that the best weight is *100-w* with 96.79% accuracy and 3.21% error, as shown in Table 5.

Table 5. Z-score normalization results

Weight Adjustment	Acc.	Err.	FAR	FRR	EER
w+1	89.46%	10.55%	10.52%	10.67%	-
w+10	95.70%	4.30%	4.30%	4.33%	-
w+100	96.67%	3.33%	3.33%	3.33%	3.33%
w+1000	96.67%	3.33%	3.33%	3.33%	3.33%
(10*w)+1	86.18%	13.82%	13.85%	13.67%	-
(100*w)+1	86.30%	13.70%	13.78%	13.33%	-
(1000*w)+1	86.24%	13.76%	13.85%	13.33%	-
1-w	93.03%	6.97%	6.96%	7.00%	-
10-w	96.67%	3.33%	3.33%	3.33%	3.33%
100-w	**96.79%**	**3.21%**	**3.19%**	**3.33%**	**-**
1000-w	96.67%	3.33%	3.33%	3.33%	3.33%
1-(w*10)	86.36%	13.64%	13.63%	13.67%	-
1-(w*100)	86.30%	13.70%	13.70%	13.67%	-
1-(w*1000)	86.30%	13.70%	13.78%	13.33%	-

The SoftMax function outputs are probabilities ranging between 0 and 1, and the summation of all weights has to be 1. Since the values of SD are small when the SoftMax function calculates weights, the outputs from the function are almost similar for all positions in the vector. That makes the performance for all weights look identical at 96.67% accuracy and 3.33% error, as shown in Table 6.

The last function is the ReLU function, which changes negative values to zeros but keeps the values of positive numbers as they are. This function does not normalize any values, but it eliminates the negative values to solve the vanishing gradients in deep learning. The best weight is *1-(w*10)* at 96.85% accuracy and 3.15% error, as shown in Table 7.

Table 6. SoftMax function results

Weight Adjustment	Acc.	Err.	FAR	FRR	EER
w+1	96.67%	3.33%	3.33%	3.33%	3.33%
w+10	96.67%	3.33%	3.33%	3.33%	3.33%
w+100	96.67%	3.33%	3.33%	3.33%	3.33%
w+1000	96.67%	3.33%	3.33%	3.33%	3.33%
(10*w)+1	96.67%	3.33%	3.33%	3.33%	3.33%
(100*w)+1	96.67%	3.33%	3.33%	3.33%	3.33%
(1000*w)+1	96.67%	3.33%	3.33%	3.33%	3.33%
1-w	96.67%	3.33%	3.33%	3.33%	3.33%
10-w	96.67%	3.33%	3.33%	3.33%	3.33%
100-w	96.67%	3.33%	3.33%	3.33%	3.33%
1000-w	96.67%	3.33%	3.33%	3.33%	3.33%
1-(w*10)	96.67%	3.33%	3.33%	3.33%	3.33%
1-(w*100)	96.67%	3.33%	3.33%	3.33%	3.33%
1-(w*1000)	96.67%	3.33%	3.33%	3.33%	3.33%

Table 7. ReLU function results

Weight Adjustment	Acc.	Err.	FAR	FRR	EER
w+1	96.67%	3.33%	3.33%	3.33%	3.33%
w+10	96.67%	3.33%	3.33%	3.33%	3.33%
w+100	96.67%	3.33%	3.33%	3.33%	3.33%
w+1000	96.67%	3.33%	3.33%	3.33%	3.33%
(10*w)+1	96.67%	3.33%	3.33%	3.33%	3.33%
(100*w)+1	95.70%	4.30%	4.30%	4.33%	-
(1000*w)+1	91.33%	8.67%	8.67%	8.67%	-
1-w	96.67%	3.33%	3.33%	3.33%	3.33%
10-w	96.67%	3.33%	3.33%	3.33%	3.33%
100-w	96.67%	3.33%	3.33%	3.33%	3.33%
1000-w	96.67%	3.33%	3.33%	3.33%	3.33%
1-(w*10)	**96.85%**	**3.15%**	**3.19%**	**3.00%**	**-**
1-(w*100)	96.42%	3.58%	3.56%	3.67%	-
1-(w*1000)	86.30%	13.70%	13.70%	13.67%	-

Table 8. Summary

Technique	Weight Adj.	Acc.	Err.	FAR	FRR	EER
Euclidean Norm	10-w	96.91%	3.09%	3.11%	3.00%	-
Mean	10-w	96.97%	3.03%	3.04%	3.00%	-
Min-Max	10-w	96.97%	3.03%	3.04%	3.00%	-
Z-score	100-w	96.79%	3.21%	3.19%	3.33%	-
SoftMax	All	96.67%	3.33%	3.33%	3.33%	3.33%
ReLU	1-(w*10)	96.85%	3.15%	3.19%	3.00%	-

The summary of the best weight adjustment for each normalization technique and activation function is shown in Table 8. The overall weights are *x-w*, where *x* can be 1, 10, or 100, depending on the technique. The best result is 96.97% accuracy and 3.03% error for Mean and Min-max normalizations with *10-w* as weight adjustment.

4.1 Example Data

From the results in Table 1, Table 2, Table 3, Table 4, Table 5 and Table 6, numerical example data from 1 user is explained in detail. This user has a six-character username, and the SD vector contains 2 keystroke features for each key, and the total is 12 elements in the vector. The SD vector is shown as follow.

SD vector
[0.00000, 0.00234, 0.00229, 0.00255, 0.00177, 0.00230, 0.00118, 0.00086, 0.00085, 0.00133, 0.00109, 0.00000]

The values in the SD vector are all positive numbers with very small magnitudes. The SD vector has to be normalized using one of six normalization techniques. The outputs from all techniques are shown as follows.

<u>Euclidean norm</u>
[**0.00000**, 0.41756, 0.40853, <u>0.45481</u>, 0.31578, 0.40957, 0.21054, 0.15394, 0.15134, 0.23661, 0.19370, **0.00000**]

<u>Mean normalization</u>
[-0.54096, 0.37714, 0.35728, <u>0.45904</u>, 0.15336, 0.35958, -0.07803, -0.20249, -0.20821, -0.02071, -0.11506, **-0.54096**]

<u>Min-max normalization</u>
[**0.00000**, 0.91810, 0.89824, <u>1.00000</u>, 0.69432, 0.90054, 0.46293, 0.33847, 0.33275, 0.52025, 0.42590, **0.00000**]

<u>Z-score- normalization</u>
[**-1.55996**, 1.08757, 1.03029, <u>1.32374</u>, 0.44224, 1.03691, -0.22502, -0.58391, -0.60041, -0.05971, -0.33179, **-1.55996**]

<u>Softmax function</u>
[**0.08322**, 0.08341, 0.08341, <u>0.08343</u>, 0.08337, 0.08341, 0.08332, 0.08329, 0.08329, 0.08333, 0.08331, **0.08322**]

<u>ReLU function</u>
[**0.00000**, 0.00234, 0.00229, <u>0.00255</u>, 0.00177, 0.00230, 0.00118, 0.00086, 0.00085, 0.00133, 0.00109, **0.00000**]

The minimum values from each output are in bold, while the maximum values are underlined for better visualisation. Since the outputs from normalization techniques can be zero, the weights must be manipulated to avoid the weights being zero. In addition, the output values are small, and the outputs or weights from each technique have to be adjusted to give the best accuracy. The best SD vector profiles for each technique with different weight adjustments are presented as follows.

SD vector profile (Best results)

- *at 10-w as weight adjustment*

<u>Euclidean norm</u>
[<u>10.00000</u>, 9.58244, 9.59147, **9.54519**, 9.68422, 9.59043, 9.78946, 9.84606, 9.84866, 9.76339, 9.80630, <u>10.00000</u>]

<u>Mean normalization</u>
[<u>10.54096</u>, 9.62286, 9.64272, **9.54096**, 9.84664, 9.64042, 10.07803, 10.20249, 10.20821, 10.02071, 10.11506, <u>10.54096</u>]

<u>Min-max normalization</u>
[<u>10.00000</u>, 9.08190, 9.10176, **9.00000**, 9.30568, 9.09946, 9.53707, 9.66153, 9.66725, 9.47975, 9.57410, <u>10.00000</u>]

<u>Softmax function</u>
[<u>9.91678</u>, 9.91659, 9.91659, **9.91657**, 9.91663, 9.91659, 9.91668, 9.91671, 9.91671, 9.91667, 9.91669, <u>9.91678</u>]

- *at 100-w as weight adjustment*

<u>Z-score normalization</u>
[<u>101.55996</u>, 98.91243, 98.96971, **98.67626**, 99.55776, 98.96309, 100.22502, 100.58391, 100.60041, 100.05971, 100.33179, <u>101.55996</u>]

- *at 1-10w as weight adjustment*

<u>ReLU function</u>
[<u>1.00000</u>, 0.97658, 0.97708, **0.97449**, 0.98229, 0.97702, 0.98819, 0.99136, 0.99151, 0.98673, 0.98913, <u>1.00000</u>]

The minimum values from each SD vector profile are in bold, while the maximum values are underlined. From all SD vector profiles, the minimum value from the normalization techniques gave maximum weight for all SD vector profiles. The different values among weights should be low. The Euclidean norm, Mean, and Min-max normalizations gave high accuracy at 10-w as weights. It means that the weight values should be around 10. The Euclidean norm showed the least standard variation at 0.158 compared to the Mean and Min-max normalizations at 0.347. Therefore, the standard deviation affects the performance of the technique. From the results, standard deviations of Mean and Min-max normalizations are the same, and the accuracy is identical at 10-w weight adjustment. The results also showed that the SoftMax function is unsuitable for the keystroke vector dissimilarity technique because the weights are almost identical. It means that the importance of each value is similar and cannot finally solve the Euclidean distance problem. The z-score normalization gave the highest standard deviation (1.0) at 100-w as weights, but the accuracy was less than that of the Mean and Min-max normalizations. The ReLU function is not a good technique since the values of the SD vector are all positive, and the output from the ReLU function is the same as the SD vector.

5 Conclusion

This study examines normalization techniques, activation functions, and weight adjustments for the keystroke vector dissimilarity technique proposed in [1]. The study aims to enhance the accuracy of the technique to improve the authentication process. The normalization techniques and activation functions include Euclidean norm, Mean normalization, Min-max normalization, Z-score normalization, SoftMax function, and ReLU function. Weight adjustments were varied during the experiments. The results indicate that Mean normalization and Min-max normalization with 10-w as a weight gave the same highest result, achieving 96.97% accuracy and 3.03% error, outperforming the previous work in [1], which achieved 96.67% accuracy and 3.33% error.

The analysis steps can be applied to other keystroke dynamic datasets for future research to enhance accuracy. In addition, the verification process for keystroke vector dissimilarity can be modified to employ machine learning techniques rather than the Six Sigma technique.

References:

[1] N. Bussabong and T. Anusas-amornkul, Enhanced Keystroke Dynamics Authentication Using Keystroke Vector Dissimilarity, *in 2023 15th International*

Conference on Information Technology and Electrical Engineering (ICITEE), Chiang Mai, Thailand, 2023, pp. 223-228, doi: 10.1109/ICITEE59582.2023.10317643.

[2] M. I. W. Pramana, Suhardi, N. B. Kurniawan and J. Sembiring, Keystroke dynamics for authentication using dynamic time warping, *in 2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, NakhonSiThammarat, 2017, pp. 1-5, doi: 10.1109/JCSSE.2017.8025915.

[3] A. Bhatia, M. Hanmandlu, S. Vasikarla and B. K. Panigrahi, Keystroke Dynamics Based Authentication Using GFM, *in 2018 IEEE International Symposium on Technologies for Homeland Security (HST)*, Woburn, MA, USA, 2018, pp. 1-5, doi: 10.1109/THS.2018.8574195.

[4] A. Foresi and R. Samavi, User Authentication Using Keystroke Dynamics via Crowdsourcing, *in 2019 17th International Conference on Privacy, Security and Trust (PST)*, Fredericton, 2019, pp. 1-3, doi: 10.1109/PST47121.2019.8949031.

[5] T.-L. Lin and Y.-S. Chen, A Chinese Continuous Keystroke Authentication Method Using Cognitive Factors, *in 2019 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW)*, Yilan, 2019, pp. 1-2, doi: 10.1109/ICCE-TW46550.2019.8991902.

[6] P. M. Koh and W. K. Lai, Keystroke Dynamics Identification System using ABC Algorithm, *in 2019 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, Selangor, Malaysia, 2019, pp. 108-113, doi: 10.1109/I2CACIS.2019.8825073.

[7] S. Singh, A. Inamdar, A. Kore and A. Pawar, Analysis of Algorithms for User Authentication using Keystroke Dynamics, *in 2020 International Conference on Communication and Signal Processing (ICCSP)*, Chennai, India, 2020, pp. 0337-0341, doi: 10.1109/ICCSP48568.2020.9182115.

[8] N. Çevik, S. Akleylek and K. Y. Koç, Keystroke Dynamics Based Authentication System, *in 2021 6th International Conference on Computer Science and Engineering (UBMK)*, Ankara, Turkey, 2021, pp. 644-649, doi: 10.1109/UBMK52708.2021.9559008.

[9] R. Chandok, V. Bhoir and S. Chinnaswamy, Behavioural Biometric Authentication using Keystroke Features with Machine Learning, *in 2022 IEEE 19th India Council International Conference (INDICON)*, Kochi, India, 2022, pp. 1-6, doi: 10.1109/INDICON56171.2022.10040168.

[10] Y. B. W. Piugie, J. Di Manno, C. Rosenberger and C. Charrier, Keystroke Dynamics based User Authentication using Deep Learning Neural Networks, *in 2022 International Conference on Cyberworlds (CW)*, Kanazawa, Japan, 2022, pp. 220-227, doi: 10.1109/CW55638.2022.00052.

Contribution of individual authors to the creation of a scientific article (ghostwriting policy)

- Tanapat Anusas-amornkul gave concepts, supervised, analyzed, and wrote this research.
- Naphat Bussabong implemented and conducted the experiments for this research.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself
No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare.

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)
This article is published under the terms of the Creative Commons Attribution License 4.0
https://creativecommons.org/licenses/by/4.0/deed.en_US