PC-based User Continuous Authentication System Using the User's Finger Stroke Characteristics

Hyung-dong Lee[1], KiHyo Nam[2], Heewoong Lee[2], Mun-Kweon Jeong[2]

[1]Institute for National Security Strategy, Seoul, Korea

[2]SD R&D Center, UMLogics Co., Ltd., Daejeon, Korea

Received: September 09, 2023; Accepted: October 29, 2023; Published: November 14, 2023

Abstract

Biometric technology, which performs continuous authentication based on user behavior, has been developed in various ways depending on the type of device, input device, and sensor. Research on continuous authentication technology in PC-based systems with few sensors installed is based on data from 3physical devices that extract and analyze features from keyboard and mouse input patterns. Among these, previous studies on continuous authentication through keyboard input performed continuous authentication using the key hold delay time that occurs when one key is pressed, the key interval delay time that occurs due to the interaction between fingers, and the key press delay time. did However, the keyboard-based continuous authentication model has limitations in increasing accuracy due to a small number of features. Therefore, in this paper, when a user inputs a sentence using a QWERTY keyboard in a PC system, the function is subdivided by reflecting the characteristics of each finger and used for continuous authentication. The features extracted by reflecting the characteristics of the finger were subdivided into a total of 151 latencies, and SVDD, decision tree, and CNN were used as continuous authentication models. Experimental data was collected through the user's input of randomly displayed sentences, and features were created based on this. User keystroke behavior was used to validate the continuous authentication model. Validation metrics included thresholds for classification accuracy (ACC), ROC curves, false rejection rate (FRR), equal error rate (EER), and false acceptance rate (FAR). As a result of the experiment, it was found that continuous authentication including the user's finger input pattern was superior to the existing method.

Keywords: Continuous Authentication, Biometrics, Keyboard Stroke, Support Vector Machine, Convolution Neural Network.

1 Introduction

Most of methods to authenticate user of desktop computer or mobile device place their starting point on entry point into the system. Those methods typically require password-based authentication on the user and permit access only when the correct password is entered. Although these entry-point-based authentication methods have been used in various fields, they have weakness in terms of security that the device is unable to detect intruder after the authentication step is completed successfully.[1] In addition, the recent pandemic caused by the corona virus hastened the transition to a non-face-to-face environment, and entry point-based user authentication techniques have been applied to various non-face-to-face services such as remote business processing and video conferences. As a result, security threats caused by authorized users have further increased.[2] In particular, the session hijacking targeting open sessions caused a problem that both passive and active attacks such as internal

Research Briefs on Information & Communication Technology Evolution (ReBICTE), Vol. 9, Article No. 10 (November 14, 2023)
*DOI:https://doi.org/10.56801/rebicte.v9i.****
This paper is an English version of the paper published in "THE JOURNAL OF KOREAN INSTITUTE OF NEXT GENERATION COMPUTING, Vol. 19, No. 2, pp. 114-130, April 2023

information leakage and backdoor installation are possible. In particular, session hijacking, which targets opened sessions by an attacker, has caused problems in which both passive and active attacks can be performed, such as leakage of internal information and installation of backdoors. As a countermeasure to this, the Zero Trust-based security model has recently been in the limelight again. Zero Trust is a model that is based on the principle of no trust and gives access to the desired information only after going through a legitimate authentication process regardless of the inside or outside of the service.[3] The zero trust security model requires analyzing the current state of a user device accessing a service from the outside or accessing a service using a more strengthened authentication procedure such as multi-factor authentication. In addition, it includes the process of verifying that the user is a legitimate user by allowing authorized users to access only authorized resources and continuously monitoring the user. The technology suitable for this is suitable for the zero trust security model because it can analyze the user's behavior with continuous authentication technology to verify whether it is a legitimate user and detect abnormal behavior.

The process of continuous authentication, called as dynamic authentication or implicit authentication, starts after user obtained permission from system and is used to verify that the current user is the authorized one.[4] The application of this method is studied on two devices broadly. The first is application to mobile devices where continuous authentication is performed by learning user behavior patterns using various sensors such as magnetometers, accelerometers, and gyroscopes installed in the device.[5] Since almost no sensors are installed in PC-based systems, previous studies have performed continuous authentication by using data generated from the keyboard or mouse, or combining data from various devices such as keyboard and mouse or keyboard and camera[6]. Among them, studies on continuous authentication using only keyboard input patterns in PC-based systems mainly adopted methods that combine authentication models or propose new frameworks because the types of functions are not diverse.

Authentication based on keyboard input patterns is one of the most difficult devices to extract user characteristics among PC input devices. It is difficult to extract the user's characteristics because the keyboard is used by simply pressing keys rather than moving in various forms like a mouse. In addition, since the typing speed that occurs when entering a sentence is a difficult factor to specify a user, many studies currently extract user characteristics with latency using each key event. This key latency is used as an input value for user authentication model learning by combining a key down event and a key up event using one or two keys. However, this method has a limitation that the number of features is very small, and the small number of features also affects the accuracy of the user authentication model. Therefore, it is necessary to secure user authentication accuracy through segmentation of features that are combined using latency.

In this paper, continuous authentication is performed using features extracted based on keyboard input stroke and the interaction among user's fingers. When the user enters character using the keyboard, three types of latency can be extracted. The first is the key hold latency referring to time interval between the Down event and the Up event that occurs when a key is pressed. The second is the key interval latency referring to time interval between the Up event of a key and Down event of next key. The third is the key press latency referring to time interval between Down events of two keys. The first latency is measured, for each finger, using the pressure generated by pressing a key. The second and third latencies are measured using the finger-to-finger key event time interval. In this study, the data generated when users input randomly given sentences are collected. The software developed for this study has two functions: data collection by hooking keyboard events and verification using the collected data. SVDD and Decision Tree (DT) and Convolutional Neural Network (CNN) were used as the continuous authentication model. The contributions of this study are as follows:

- To propose a method for performing continuous authentication using finger-based characteristics in an environment where a user inputs text using a keyboard
- The data that can be extracted through keyboard input is subdivided and used for continuous authentication.
- To use the input pattern for each finger and the interaction pattern between each finger as a segmentation criteria
- To use key pressure information for nine fingers (left side four, right side four, two thumbs are considered as one finger) and 72 (9(Total number of fingers)x8(Number of fingers interacting with 1 finger)) interaction information among nine fingers in experiment
- To use SVDD, decision tree, and deep learning models to verify user authentication data including finger-based characteristics
- To compare and analyze the results of studies on newly proposed continuous authentication method using finger-based features and on continuous authentication method using general features.

This paper is organized as follows. In Section II, previous studies and prior knowledge on keyboard-based continuous authentication are described. Section III provides a description of data collection and preprocessing methods. In Section IV, the methodology and certification model for continuous certification are presented. Section V describes the function implementation and experiment results. Finally, the conclusions are in Section VI

2 Related Work

Research on continuous authentication using the keyboard is still in progress, and this section briefly describes the existing research on continuous authentication using the keyboard.

2.1 Continuous Authentication using Key Latency

The data used for latency-based continuous authentication uses the time difference of keyboard events, keyboard input speed, and user usage patterns to authenticate users and verify performance through machine learning and statistical techniques.

S.J. Shepard [7] tried continuous authentication using keyboard for the first time and measured the time point when two keys were pressed for key down and key up. For continuous authentication, a unique pattern that occurs when diphthong is entered was used.

M. S. OPbaidat [8] performed an experiment to collect and classify data using the interkey time, which is the time interval in which the user presses a key consecutively while the experimenter is typing a specific phrase, and the hold time, in which the user presses and releases the key. Authentication was carried out by dividing into experiments for collection and classification, and it was verified with an artificial neural network called HSOP (Hybrid-Sum-Of-Products). Since HSOP is an artificial neural network paradigm announced in 1994, this paper verifies the accuracy of continuous authentication using CNN applied to various fields recently.

R Giot et al. [9] performed continuous authentication using statistical data generated from

keystrokes: duration of the key press of each key(H), the delay between each key press(DD), the delay between a key release and the next key press(UD) were used in generating (H, DD, UD) for each sample. After combining multiple samples into a set called gallery, the average, median, standard deviation, maximum between mean and median, and minimum between mean and median values of the gallery were applied to the equation to calculate the score for each user and update the model.

Jae-Wook Lee et al. [10] used the data, a measurement of the duration and latency while inputting sentences as a timing vector. After determining an appropriate pseudo-volume of the elliptical hypothetical space using a hybrid genetic algorithm, FRR and FAR of user authentication using keyboard were calculated.

Eric Flior et al. [11] explained that the data generated from keystrokes is limited and developed a system that authenticates users by using the Cosine Correlation based on features including Typing speed referring to the maximum keystroke speed per minute, keystroke seek-time referring to the time required for a user to find and press a specific key, Flight-time referring a time interval between two key-up events or key-down events, Characteristic sequences of keystrokes referring to sequences of words typed repeatedly and frequently, and Examination of Characteristic Errors referring to specific errors in a user's typing.

Y. Kaneko et al. [12], after having subjects to input 51 pre-determined themes and collecting time interval data of down-down events of 31 diphthong keys, calculated lower EER through outlier data filtering based on similarity with Hamming-distance.

M. S. Hossain et al. [13] extracted three features from the keyboard: key interval latency, key press latency and key hold latency. As the user authentication method, the previously known Otsu method and Gaussian method were used, and the Order Static rejection method proposed in the paper was compared and analyzed.

M. S. Hossain et al. [14] Data were collected by dividing input style data when the subjects entered creative text and realistic text, and when they typed the keyboard on a hard surface versus a soft surface. The collected data was tested by generating a concordance score between the Relative R scale of the experimental data and the test data with Key Interval Latency, Key Press Latency, and Key Hold Latency.

Stylios, Ioannis, et al. [15] collected keystroke actions for smartphones in Android applications. Create a feature using the duration, latency, pressure, X, Y coordinates of keys generated from the smartphone's virtual keyboard. They generated a Multi Layer Perceptron (MLP) model using the keystroke features and evaluated the performance of continuous authentication using Accuracy and EER indicators. In this paper, an experiment is conducted by generating a model for continuous authentication using CNN rather than MLP.

Lu, Xiaofeng, et al. [16] studied continuous authentication using keystrokes that occur when entering free-text. The keystroke sequence was created by extracting the hold time information that occurs when the key is entered, the Release-Press event, and the Press-Press event, and the user matching rate is calculated using sliding window authentication. Sliding window authentication is a method of extracting keystroke as much as a window of a certain size in a keystroke sequence, entering it into a continuous authentication model, moving the window one step at a time, and calculating a user matching rate. The authentication model used CNN+RNN and the performance of the continuous authentication model was verified using FAR, FRR, and EER.

A. T. Kiyani [17] used user ID, session ID, key down time, and up time as keystroke information.

they conducted experiments by considering keystrokes as sequential series and proposing a Robust-Recurrent Confidence Model (R-RCM) model using LSTM suitable for learning time series data. The LSTM generated hyperparameters representing the user's characteristics and used the final threshold calculated by the R-RCM to determine whether the user should use the system. The results of the study showed better performance than the continuous authentication model using only LSTM. and A.T. Kiyani et al.[18] diversified features into Key Monograph Action and Key Diagraph Action in a later study, and applied them to the R-RCM model to conduct a continuous authentication study.

de-Marcos, Luis, et al. [19] studied the continuous authentication framework using sensor information and Soft-Keyboard events occurring in BYOD environment. The feature extracted from the Soft-Keyboard is Keycode, PressingTime, TimeBetweenPress, TimeReleaseNextPress, and NextKeyCode. Using the extracted feature, Unsupervised Machine Learning (EllipticEnvelope, IsolationForest, OneClassSVM) and Supervised Machine Learning (RFC, SGD, SVC) were used to create models. The generated model was applied to the continuous authentication framework and the performance was verified using learning data, test data, and verification data.

Martín, Alejandro G., et al. [20] studied how to perform continuous authentication using User and Entity Behavior Analysis (UEBA) technology. For keyboard stroke information, HoldKey1, HoldKey2, Release-Press, Press-Press, Release-Release, and Press-Release events were used, and continuous authentication research was conducted with Mouse Dynamic. Continuous authentication used the N-gram method, and the performance of the continuous authentication model was evaluated using FAR, FRR, and EER.

Ayotte, Blaine, et al. [21] studied continuous authentication using Monograph and Digraph features of keystrokes. The dataset used two datasets, Calrkson II and Buffalo, and the feature used Monograph's Key duration and Digraph's Down-Down, Up-Down, Down-Up, and Up-Up events as features. Five evaluation indicators were used: FAR, FRR, EER, Average Number of Genuine Action (ANGA), , Average Number of Imposter Actions (ANIA) and The continuous authentication model used six models: Instance-based Tail Area Density(ITAD) Metric, Kernel Density Estimation(KDE), Manhattan Distance, Scacled Manhattan, Mahalanobis, Transformed Mahalanobis. As a result of the study, Monograph, Up-Up Diagram, and Down-Down Diagram were the features that had the greatest influence on continuous authentication and Using the ITAD Metric proposed in the paper, they improved performance to detect impostors twice as fast.

Chen, Zhaohang, et al. [22] studied user authentication using Hold Time (HT) occurring in one key and Flight Time (FT) occurring in two keys. HT is the time duration between the press and release of a certain key. FT refers to the time interval between two consecutive keys, which has four types, Press-Press, Press-Release, Release-Press, Release-Release FT. In the paper, static and continuous authentication of users were evaluated using Gaussian Model in Edge Computing Architecture. As a result of the evaluation by dividing it into three scenarios: HT single feature, FT single feature, and HT&FT feature, it was confirmed that the most effective EER indicator was found in HT&FT feature and could be applied to the Online Examination environment.

2.2 Continuous Authentication using Keyboard Pressure

In continuous authentication using hardware, the data is collected using a sensor that detects pressure installed through physical modification of the keyboard.

H. Nonaka et al. [23] estimated the moment of key stroke using the timing and waveform of the key-down event and key-up event measured based on the data collected using the pressure sensor mounted on the keyboard.

K. KOTANI et al. [24] measured the pressure, latency time, and hold time when a keystroke occurs using a number input keypad equipped with a pressure sensor, and calculated EER through FRR and FAR in four types of environments.

C. C. Loy et al. [25], using a keyboard with a pressure sensor, collected feature data such as average pressure, noise, and distortion and tested the performance of user authentication by using Multi Layer Perceptron, Logistic Regression, Fuzzy ARTMAP neural networks, and statistical approach.

N.J. Grabham et al. [26] studied user authentication using a pressure sensor mounted on a bank ATM keypad. The features used for authentication were On and Off Times, Maximum Force, Average Force, and Maximu Gradient, and verification was performed using Component-Wise Linear Verification and L1-Norm and L2-Norm Verification.

More related works were compared below in Table 1.

Table 1. Comparison of related works

Paper	Feature Type	Authentication Method	FAR	FRR	EER
[10]	KH, KI	Ellipsoidal Hypothesis Space	10.5%	10.2%	10.3%
[13]	KH, KP, KI	Order Statistic rejection method	N/A	N/A	18.6%
[14]	KH, KP, KI	R Measures	N/A	N/A	6.0%
[16]	KH, KP, KI	CNN+RNN	4.12%	1.95%	3.04%
[20]	KH, KP, KI, KD-KU, KU-KU	N-gram based Risk Model	5.9%	7.4%	5.9%
[21]	KH, KP, KI, KD-KU, KU-KU	ITAD metric	N/A	N/A	7.8%
[22]	KH, KP	Gaussian Model	N/A	N/A	7.46%

(KP: key hold latency, KP: key press latency, KI: key interval latency, KD-KU: key down-key up latency, KU-KU: key up-key up latency)

3 Data Collection & Pre-processing

3.1 Raw Data Collection

· Software for data collection: In this study, a software was developed to collect event data that occurs on the user's keyboard. The software collected data through the pyHook package. The software consists of a function to randomly appear sentences, a function to display an input window where characters can be entered on the screen, and a function to hook events that occur on the keyboard and store them in the database.

· Subjects: The subjects were 10 right-handed staffs of the laboratory in their 20s to 40s and (male: n=8, female: n=2). They had a bachelor's degree and higher and used computers frequently.

· Subjects' behavior: When keyboard input data is collected in an environment where users generally use computers, a phenomenon in which data does not occur may occur because the frequency of keyboard use is irregular. Therefore, in this paper, in order to collect data from participants in a controlled environment, all keyboard events generated through the act of entering randomly appearing sentences during software execution were hooked and saved. The subject's behavioral information was collected with the consent to use it only for research purposes without providing it to the outside.

3.2 Key Information on each Finger

The QWERTY type keyboard used in the experiment has various keys such as characters, special keys, and function keys, but the keys generally used for text input are limited, and most users assign a key to press to each finger. The patterns of each of the four left and four right fingers used when the subjects inputs a sentence were extracted. In addition, since most of the space keys are used with the thumb, and whether the left thumb or the right thumb is different for each participant, it is assumed that there is only one thumb. Since the participants in the experiment are accustomed to using a computer, most of the fingers are determined according to the keyboard keys as shown in Table 2.

Table 2. Keys assigned to each finger

No	Finger	Assigned Keys
1	Left pinky	q, a, z, Left SHIFT
2	Left ring	w, s, x
3	Left middle	e, d, c
4	Left index	r, f, v, t, g, b
5	Right index	y, h, n, u, j, m
6	Right middle	i, k, ‘,’
7	Right ring	o, l, ‘.’
8	Right pinky	p, ‘;’, ‘/’, Right SHIFT, BACKSPACE
9	Thumb	SPACE

Special characters were not included in the study since their input method and patterns are varied considerably among subjects.

3.3 Information extracted by using Keyboard Events

As a key event that occurs when a sentence is input using a keyboard, it is possible to measure the key interval latency and key press latency that occur through two key combinations, and the key hold latency that occurs with one key.

- Key interval latency(KI): Time interval between the Up event of a key and the Down event of a key pushed next
- Key press latency(KP): Time interval between the Down event of a key and the Down event of a key pushed next
- Key hold latency(KH): Time interval between Down and Up events of a key

3.4 Fingers-based Features

Since key hold latency is an event that occurs when one finger presses and releases one key, we use a total of 9 fingers are used when reflecting finger characteristics, so 9 features can be extracted for each finger. Since key interval latency measures how long it takes for one finger to generate a key up event and another finger to generate a key down event, so you can generate 8 features per finger. Therefore, by extracting the key interval latencies from 9 fingers, a total of 72 features can be generated. Like key interval latency, key press latency measures the time interval between one finger's key-down event (9 types) and another finger's key-down event (8 types), so a total of 72 (=9x8) features are extracted. Thus, a total of 151 features can be subdivided.

3.5 Data Preprocess

Key-down and key-up events that occurred while participants were typing text were stored in the database in the order in which they occurred. The stored data is grouped by sentence to generate general KH, KP, KI characteristics and KH, KP, KI characteristics that reflect finger characteristics. The characteristics reflecting the characteristics of the finger first go through a process of allocating the finger to determine the finger that generated each event according to Table 2. After the finger assignment process is finished, data preprocessing is performed by extracting the key hold waiting time, key pressing waiting time, and key interval waiting time for each finger as shown in Figure 1 below.

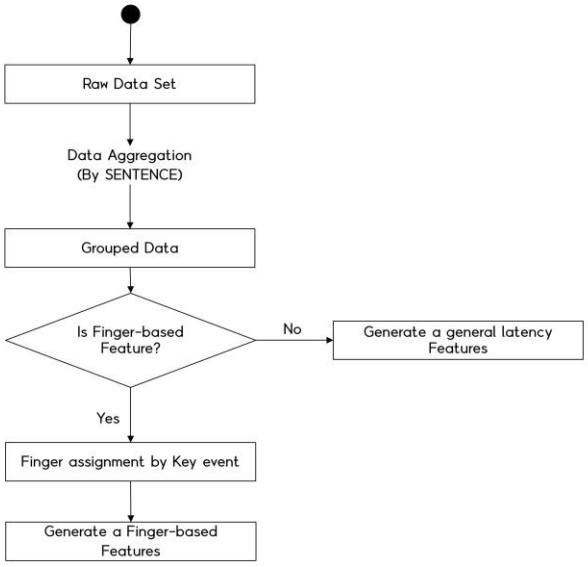

Figure 1. Keyboard Event Feature Flowchart

4 Continuous Authentication Model

4.1 Continuous Authentication Algorithm

In this paper, Support Vector Data Description (SVDD) and DT (Decision Tree), and Convolutional Neural Network (CNN) were adopted to perform continuous authentication. Each authentication model determines whether a user is authenticated by learning an extracted feature set. The concepts of Support Vector Data Description (SVDD) and DT (Decision Tree), and Convolutional Neural Network (CNN) are shown in Figure 2, 3 and 4.

- SVDD: SVDD, a type of nonlinear SVM, is a method using data mining and is most often used to classify single-class data. Single-class classification is a method for identifying invalid vectors in a given group of vectors, and SVDD was first proposed by David [27].

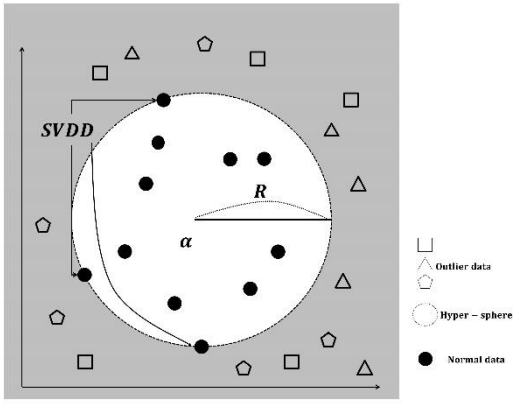

Figure 2. Basic Concept of SVDD

This method maps the data into a higher-dimensional vector space through a non-linear transformation. A data class consisting of data with the same characteristics is created by creating a boundary of a hyper-sphere with a minimum radius that includes most of mapped data. It is important, therefore, to find the origin, and radius, that generate an optimized hyper-sphere containing most of the mapped data. The validity of the data is determined by classifying the mapped data inside and outside of this hyper-sphere.

· DT: It is a predictive model that connects observations and target values using a decision tree and is used as a predictive modeling method in statistics, data mining, and machine learning [28].

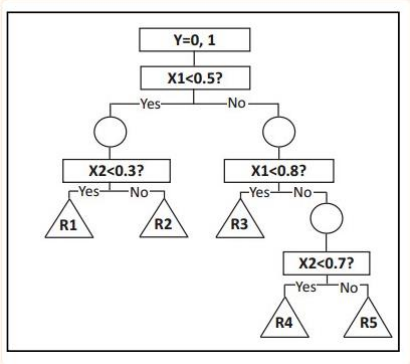

Figure 3. Basic concepts of Decision tree

In this model, the scores of the root node, child node, and leaf node are calculated using user features and verified using validation data.

· CNN: It is a normalized version of the multilayer perceptron. A multilayer perceptron is usually a fully connected network, meaning a neural network structure in which each neuron in a layer connects to all neurons in the next layer [29]. This study, referring to [30], uses a kernel of 1x1 pixel in three convolutional layers. ReLU was used as the activation function, and a dropout probability was set as 0.05 to prevent overfitting. In addition, each convolutional layer is connected to the Fully Connected layer after passing through the Max-Pooling layer, which determine the user authentication probability. The model trained on the

vectorized features calculates the user agreement rate using the input validation data.

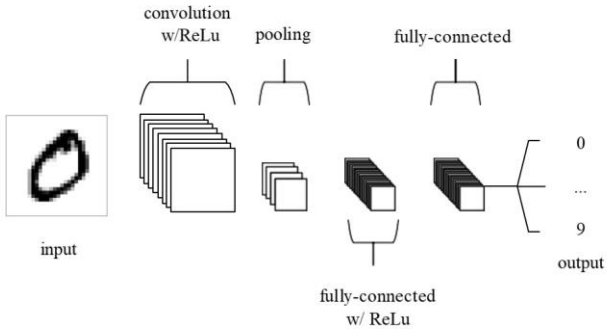

Figure 4. Simple CNN architecture [31]

4.2 Continuous Authentication Model Design

The continuous authentication model for authenticating users consists of a total of three steps. Sine the model generated for continuous authentication is verified using user keystroke behavior, while the step one data collection is performed only once, the step two learning and step three verification processes are repeated.

· Step 1: Data vector creation by using the user's keyboard input behavior.

A vector X containing n elements is generated based on the features described in 3.3 and 3.4.

$$X = \{x_{1,} x_2, x_3, \cdots, x_n\} \quad (1)$$

A vector group D including a vector X of a normal user behavior pattern when a user's keyboard input is repeated K times is generated.

$$D = \{X_{1,} X_2, X_3, \cdots, X_K\} \quad (2)$$

The generated vector group is separated into K data sets.

· Step 2: Machine Learning and Deep Learning

We train and model SVDD, DT, and CNN using user's keystroke data.

SVDD uses the set D defined in Step 1 as training data. The learned region is represented using a sphere H with center a and radius R defined by R_d. For each data, a penalty is imposed when the distance between the training data X_i and the center a exceeds R. The error function is defined as in the equation below, and by optimizing it, a sphere with a minimum radius is found and learned. ξ_i means a penalty point for the i-th learning data X_i to deviate from the sphere H, and C means a constant.

$$\min F(R, a) = R^2 + C\sum_i \xi_i$$

$$\text{s.t.} \left||X_i - \alpha|\right| \leq R^2 + \xi_i, \xi_i \geq 0, \forall i$$

C is a constant that controls the size of the sphere, i.e. the radius, and the relative importance of the error. In terms of performance, the higher the C value, the higher the detection rate, but the higher

the error detection rate accordingly. Conversely, when the value of C is small, the detection rate is low, but the error detection rate is also reduced. Therefore, the above equation can be converted into a dual problem by introducing a Lagrangean multiplier for the optimization operation. The expression of a function L formed using Lagrange multipliers is:

$$L(R, a, \alpha_i, \gamma_i, \xi_i) = R^2 + \ C\sum_i \xi_i \ - \ \sum_i \alpha_i R^2 + \xi_i - (\left|\left|X_i\right|\right|^2 - 2a \cdot X_i + ||a||^2) - \sum_i \gamma_i \xi_i$$
$$where, \alpha_i \geq 0 \ and \ \gamma_i \geq 0 \quad (3)$$

α_i, and γ_i represent Lagrangean multipliers, and under the condition of $\alpha_i \geq 0 \ and \ \gamma_i \geq 0$, α_i, and γ_i are maximized and R, a, and ξ_i are optimized with values that are minimized, respectively. Set the partial differential equation to 0 for the variables R, a, ξ_i and learn to find the minimum sphere using the condition $0 \leq \alpha_i \leq C$. The function L to maximize when finding the minimum sphere can be expressed as:

$$Max \ \mathrm{L} = \ \sum_i \alpha_i (x_i \cdot x_i) - \sum_{i,j} \alpha_i \alpha_j (x_i \cdot x_j) \quad (4)$$
$$where, 0 \leq \alpha_i \leq C \ and \ \sum_i \alpha_i = 1, \ \ i = 1, \dots, n$$

We can find the Lagrangean multiplier α_i that maximizes the L function and find the radius R and center *a* of the sphere. In addition, since the sphere defined on the input space can represent only a very simple area, it is possible to obtain better performance by enabling nonlinear classification through conversion to a high-dimensional feature space defined through kernel *k*. Applying the kernel to the dot product $(x_i \cdot x_j)$ of the data and rewriting it gives the following equation.

$$Max \ \mathrm{L} = \ \sum_i \alpha_i K(x_i \cdot x_i) - \sum_{i,j} \alpha_i \alpha_j K(x_i \cdot x_j) \quad (5)$$
$$where, 0 \leq \alpha_i \leq C \ and \ \sum_i \alpha_i = 1, \ \ i = 1, \dots, n$$

Continuous authentication of the user is performed using the hyper-sphere obtained in the above method.

Second, to create a decision tree, the entropy function is defined as follows.

$$Entropy(D) = -\sum_{k=1}^{C_k} p_k \log_2 p_k \quad (6)$$

D is a set of user behavior information, C_k is the total number of classes to which user behaviors belong, and p_k is the ratio of behaviors belonging to class k in D. Learning was performed using a greedy algorithm that selects attributes with high information gain., we corrected for overfitting by setting the minimum impurity and maximum depth.

Finally, CNN learning is performed by passing through three layers of Convolution + ReLU Function + Max Pooling three times. The size of the kernel used for convolution is 1×1, and the ReLU Function is used to solve the vanishing gradient phenomenon. In addition, to prevent overfitting, the dropout rate between each layer was set to 0.05, and the weight parameter was reduced using the max pooling method to extract features.

· Step 3: Verification of Performance of Continuous Authentication

Verification is performed through the user's keystroke behavior and continuous authentication performance is measured.

The verification of the SVDD model uses the test data extracted through the cross-validation method on the hyper-sphere generated through learning. If the test data is mapped to the boundary or

inside of the hypersphere, user authentication is successful,

$$||z-a||^2 = (z,z) - 2\sum_i a_i(z,X_i) + \sum_{i,j} a_i a_j (X_i,X_j) \leq R^2 \quad (7)$$

Verified that user authentication fails if mapped outside Hypersphere.

$$||z-a||^2 = (z,z) - 2\sum_i a_i(z,X_i) + \sum_{i,j} a_i a_j (X_i,X_j) > R^2 \quad (8)$$

In addition, the decision tree verified the model using the test data extracted through the cross-validation method and measured the accuracy of user authentication.

Finally, the CNN model used test data to measure model accuracy, and the performance of continuous authentication was verified by inputting data not used for training.

5 Function Implementation & Evaluation

5.1 Evaluation Method

10 subjects run the data collection software on their PC. Enter a sentence that appears randomly on the screen into the text window. The software hooks and stores keyboard events that occur when text is entered.

5.2 Evaluation Environment

The subject uses a Windows-based PC and a QWERTY keyboard, and the key re-entry time and repetition rate, which are keyboard settings provided by the operating system, are individually set.

5.3 Evaluation Process

· Software installation: Subjects install data collection software on their PC.

· Subject action: Each subject runs the software and enters a sentence that appears randomly.

· Raw data collection: The collected keystroke data is sampled based on each sentence.

· Feature extraction: User-specific keyboard input patterns and finger-based features are extracted using sampled data as shown in Figure 5. In addition, simple key interval, key press, and key hold features, excluding finger-based features, are extracted to compare with the existing research results.

· Data classification: The extracted features are classified into finger-based features and simple features. 10 data set are generated on each, among which, nine data sets are used as training data and one set is used as validation data.

· Model training: Learning is performed by loading the training dataset into three continuous authentication models (SVDD, DT, CNN).

· Model verification: Validation is performed using the model developed through learning, and FAR, FRR, and ERR are calculated. FAR refers to the probability that the user who should be rejected is authenticated, and FRR refers to the probability that the user who should be authenticated is rejected. EER refers to the threshold between FAR and FRR. In addition, the True Positive Rate

(TPR) versus the False Reject Rate (FRR) is displayed using the Receiver Operating Characteristic (ROC) curve to measure the performance of the model.[32]

$$\text{ACC} = (TP + TN)/(TP + TN + FP + FN)$$
$$\text{TPR} = TP/(TP + FN)$$
$$\text{TNR} = TN/(TN + FP)$$
$$\text{FPR} = FP/(FP + TN)$$
$$\text{FNR} = FN/(FN + TP)$$
$$\text{FAR} = (Number\ of\ incorrectly\ authorized\ users)/(Total\ number\ of\ users)$$
$$\text{FRR} = (Number\ of\ rejected\ users)/(Total\ number\ of\ users)$$
$$\text{EER} = (FAR + FRR)/2$$

where,

TP: True Positive

TN: True Negative

FP: False Positive

FN: False Negative

FAR: False Acceptance Rate

FRR: False Rejection Rate

EER: Equal Error Rate

The model validation, as with learning, is performed using a model created with 10 training data.

5.4 Evaluation

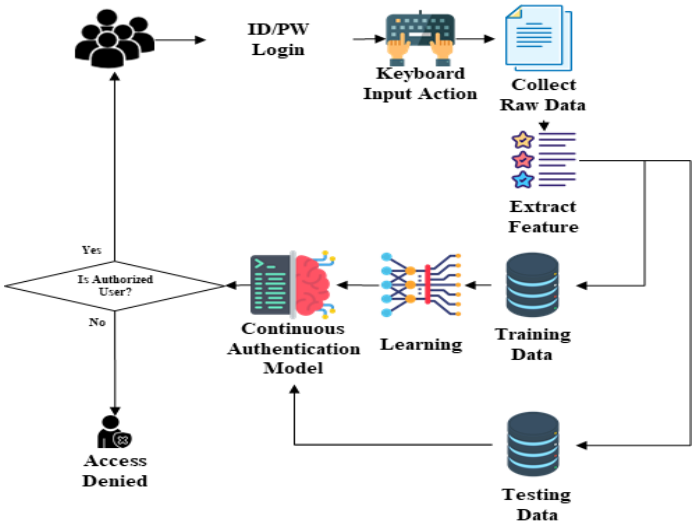

Figure 5. Continuous Authentication System using user finger-based Features

· Keyboard input: Each user logs in to the continuous authentication system using their ID and PW

and inputs 20 randomly displayed sentences, generating raw data.

· Raw Data Collection: pyHook.HookManager() object and onKeyboardEvent class are created to enable keyboard event monitoring. Keyboard events are hooked using HookKeyboard() and data such as event.MessageName, event.Time, and event.KeyID are saved.

· Feature extraction: KH, KI, and KP information is extracted from the data grouped by sentence using the collected raw data. After going through the process of assigning fingers as shown in Table 2 using the keyID of each data, KH, KI, and KP reflecting finger characteristics were newly extracted and used in the experiment.

The generated features are stored in a database and later used as training data and verification data.

DECLARE hand : Dictionary[integer:array]

BEGIN
FOR finger1 in hand
FOR finger2 in hand
IF finger1.key == finger2.key THEN
SET holdTotal = 0
GET KeyID included in finger1.value from database
SUM KeyUpTime-KeyDownTime from holdTotal
Compute holdTotal divided by finger length
Save finger hold latency to Database
ELSE
SET intervalTotal, pressTotal = 0
GET Key1ID included in finger1.value from database
GET Key2ID included in finger2.value from database
SUM Key1UpTime-Key2DownTime from intervalTotal
SUM Key1DownTime-Key2DownTime from pressTotal
Compute intervalTotal divided by Key2ID.length
Compute pressTotal divided by Key2ID.length
Save finger interval latency to Database
Save finger press latency to Database
END IF
END

· Model learning: Three types of machine learning algorithms are used for training. SVDD was trained using python's sklearn and numpy libraries. Data is loaded through np.array, the kernel matrix is calculated, and the support vector is identified by solving the Lagrange dual problem problem. Finally, SVDD is learned by calculating the circle's center and radius.

DECLARE learningData : ARRAY of DOUBLE

BEGIN
Load trainData from Database
Compute Kernel Matrix
Solve the Lagrange Dual Problem
Find Support Vectors
Compute the center of Initial Feature Space
Compute Radius R

END

DT performed learning using python's sklearn library. The classifier is created using tree.DecisionTreeClassifire(), which is the solution of the sklearn library, and the continuous authentication model is trained by fitting() the training data.

DECLARE learningData : ARRAY of DOUBLE

BEGIN
Load trainData from Database
model = tree.DecisionTreeClassifier().fit(trainData)
END

The third model, CNN, is implemented using the keras library. A model was created using the Sequential function of keras.models, and layers were inserted and added using the Dense, Conv2D, MaxPooling2D, Dropout, and Flatten functions of keras.layers. CNN model was created, as with SVDD and DT, by loading training data using a database.

BEGIN
Load trainData from Database
model = Sequential
Add 3-layer (Conv2D, ReLU, MaxPooling2D, dropout)
Add Flatten-layer
Add Dense-layer(softmax)
Model Compile
Model Fit(trainData)
END

· Model validation: The three models generated in the training phase were validated using data not used for training. SVDD and DT were verified using predict() provided by sklearn, and the accuracy of the model was measured using score(). The loss and accuracy of the CNN model were obtained using evaluate() provided by keras, and the evaluation results were confirmed through the continuous authentication scores output from each model.

5.5 Evaluation Result

The model trained in section 5.4 was evaluated using validation data. For evaluation, a model learned using finger-based features and a model learned using general features were compared for the user's keyboard input characteristics. Table 3 shows the accuracy of the three models when trained using normal keyboard functions and finger-based functions, respectively.

Table 3. Model Accuracy

Model (General)	SVDD	DT	CNN		Model (Finger-based)	SVDD	DT	CNN
Accuracy (%)	94.0	91.9	95.9		Accuracy (%)	95.3	93.5	96.1
Recall (%)	94.3	91.5	95.6		Recall (%)	96.2	92.6	97.4
Precision (%)	93.7	92.3	96.3		Precision (%)	94.5	94.3	94.8
F1-score (%)	93.9	91.8	95.9		F1-score (%)	95.3	93.4	96.0

Table 3 show that the model accuracy is SVDD:94.0%, DT:91.9%, and CNN:95.9% when trained using the general keyboard feature, and SVDD:95.3%, DT:93.5, and CNN:96.1% when trained using

the finger-based keyboard feature, indicating that the accuracy of the finger-based learning model is higher. Table 4 show the evaluation results of the model trained with the general feature and the model trained with the finger-based feature, respectively.

Table 4. Model Evaluation

Model (General)	SVDD	DT	CNN		Model (Finger-based)	SVDD	DT	CNN
FAR (%)	4.1	2.2	2.8		FAR (%)	2.7	1.1	1.2
FRR (%)	8.1	15.2	4.2		FRR (%)	6.2	9.6	3.3
EER (%)	6.1	8.7	3.5		EER (%)	4.5	5.4	2.3

The General Model in of Table 4 shows the results of measuring the FAR, FAR, and EER of the model trained using the general features, and the EER of each model is SVDD:6.1%, DT:8.7%, and CNN:3.5%. The ROC curve is shown in Figure 6.

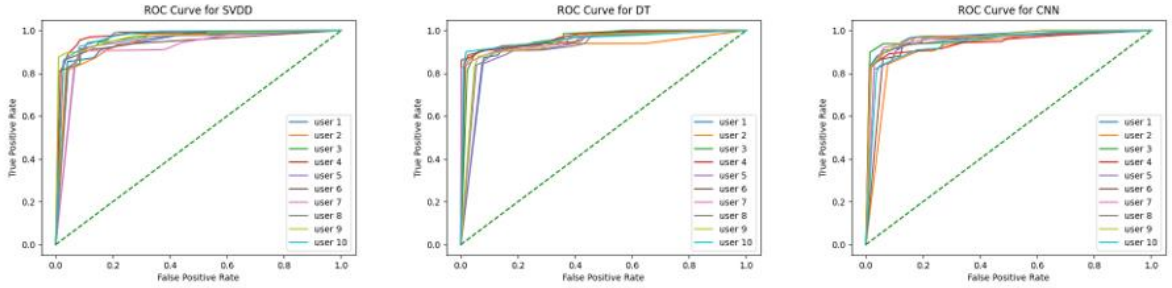

Figure 6. ROC Curve of Model Trained Using General Keyboard Features

In addition, The Finger-based Model in Table 4 shows the results of measuring FAR, FRR, and EER of models trained using finger-based features and shows that the EER of each model is SVDD: 4.5%, DT: 5.4%, and CNN: 2.3%. The ROC curves are shown in Figure 7, respectively.

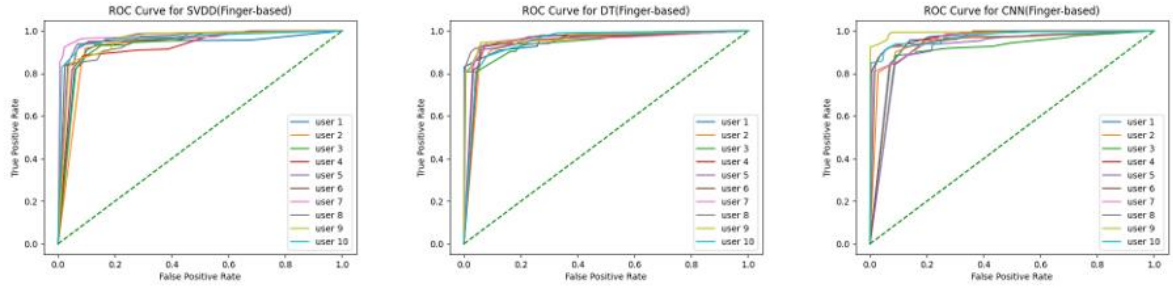

Figure 7. ROC Curve of Model Trained Using Finger-based Features

6 Conclusion

The continuous authentication was performed by creating a feature using the events occurring when users input sentence using QWERTY type keyboard. Continuous authentication is important in that it can be applied to various fields because it can continuously control user access by monitoring authorized users. This study extracted finger-based features from keyboard input data of 10 subjects

and developed a continuous authentication system that can identify users. The continuous authentication model in this study used SVDD, DT, and CNN, and the model trained using finger-based features and those trained using general features were compared. In the model trained using general features, the results of continuous authentication were as follows: FAR:4.1%, FRR:8.1%, EER:6.1% in SVDD; FAR:2.2%, FRR:15.2%, EER:8.7% in DT; FAR:2.8%, FRR:4.2%, EER:3.5% in CNN. In the model trained using finger-based features, the results of continuous authentication were as follows: FAR:2.7%, FRR:6.2%, EER:4.5% in SVDD; FAR:1.1%, FRR:9.6%, EER:5.4% in DT; FAR:1.2%, FRR:3.3%, EER:2.3% in CNN, indicating that the performance of the finger-based learning model is more excellent and that CNN displayed lower error rate than other two methods. The major contributions of the research results of this paper are as follows.

- The three features of key hold, key press, and key interval were subdivided into 151 types.
- Improved the accuracy of continuous authentication using the keyboard by using subdivided features.
- Higher accuracy can be secured in various fields such as continuous authentication as well as device validation and privacy protection based access control.

However, this paper has a limitation that it is difficult to authenticate users who do not use ten fingers. Therefore, it is determined that research on a method for collecting information on fingers using the keyboard for each user and performing continuous authentication is necessary.

References

[1] Frank, M., Biedert, R., Ma, E., Martinovic, I., Song, D. (2012). Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. IEEE Transactions on Information Forensics and Security, 8(1), 136–148.

[2] Yu, D.-H., Kim, Y.-U., Ha, Y.-J., Ryu, Y.-S. (2021). Consideration of New Convergence Security Threats and Countermeasures in the Zero-Contact Era. Journal of the Korea Convergence Society, 12(1), 1–9.

[3] Stafford, V. A. (2020). Zero trust architecture. NIST special publication, 800, 207.

[4] Alzubaidi, A., Kalita, J. (2016). Authentication of smartphone users using behavioral biometrics. IEEE Communications Surveys & Tutorials, 18(3), 1998-2026.

[5] Abuhamad, M., Abuhmed, T., Mohaisen, D., Nyang, D. (2020). AUToSen: Deep-learning-based implicit continuous authentication using smartphone sensors. IEEE Internet of Things Journal, 7(6), 5008-5020.

[6] Yap, R. H. C., Sim, T., Kwang, G. X., Ramnath, R. (2008). Physical Access Protection using continuous authentication. 2008 IEEE Conference on Technologies for Homeland Security.

[7] Shepherd, S. J. (1995). Continuous authentication by analysis of keyboard typing characteristics. In Proceedings of the European Convention on Security and Detection, 1995 (Vol. 408, pp. 16-18).

[8] Obaidat, M. S., Sadoun, B. (1996). Keystroke dynamics based authentication. Biometrics: Personal Identification in Networked Society, 213-229.

[9] Giot, R., Dorizzi, B., Rosenberger, C. (2011). Analysis of template update strategies for keystroke dynamics. In 2011 IEEE Workshop on Computational Intelligence in Biometrics and Identity Management (CIBIM) (pp. 21-28). IEEE.

[10] Lee, J. W., Choi, S. S., Moon, B. R. (2007). An evolutionary keystroke authentication based on ellipsoidal hypothesis space. In Proceedings of the 9th annual conference on Genetic and evolutionary computation (pp. 2090-2097).

[11] Flior, E., & Kowalski, K. (2010). Continuous biometric user authentication in online examinations. In 2010 seventh International Conference on information technology: new generations (pp. 488-492). IEEE.

[12] Kaneko, Y., Kinpara, Y., Shiomi, Y. (2011). A hamming distance-like filtering in keystroke dynamics. In 2011 Ninth Annual International Conference on Privacy, Security and Trust (pp. 93-95). IEEE.
[13] Hossain, M. S., Balagani, K. S., Phoha, V. V. (2012). New impostor score based rejection methods for continuous keystroke verification with weak templates. In 2012 IEEE fifth international conference on biometrics: theory, applications and systems (BTAS) (pp. 251-258). IEEE.
[14] Hossain, M. S., Haberfeld, C., Yuan, K., Chen, J., Rahman, K. A., Hussain, I. (2020). Continuous Authentication Using Creative Writing. In 2020 International Symposium on Networks, Computers and Communications (ISNCC) (pp. 1-6). IEEE.
[15] Stylios, I., Skalkos, A., Kokolakis, S., Karyda, M. (2022). Bioprivacy: Development of a keystroke dynamics continuous authentication system. In European Symposium on Research in Computer Security (pp. 158-170).
[16] Lu, X., Zhang, S., Hui, P., Lio, P. (2020). Continuous authentication by free-text keystroke based on CNN and RNN. Computers & Security, 96, 314–318.
[17] Kiyani, A. T., Lasebae, A., Ali, K. (2020). Continuous user authentication based on deep neural networks. In 2020 International Conference on UK-China Emerging Technologies (UCET) (pp. 1-4). IEEE.
[18] Kiyani, A. T., Lasebae, A., Ali, K., Rehman, M. U., Haq, B. (2020). Continuous user authentication featuring keystroke dynamics based on robust recurrent confidence model and ensemble learning approach. IEEE Access, 8, 156177-156189.
[19] de-Marcos, L., Cilleruelo, C., Junquera-Sánchez, J., Martínez-Herráiz, J. J. (2020). A Framework for BYOD Continuous Authentication: Case Study with Soft-Keyboard Metrics for Healthcare Environment. In Applied Informatics: Third International Conference, ICAI 2020, Ota, Nigeria, October 29–31, 2020, Proceedings 3 (pp. 347-358). Springer International Publishing.
[20] Martín, A. G., de Diego, I. M., Fernández-Isabel, A., Beltrán, M., Fernández, R. R. (2022). Combining user behavioural information at the feature level to enhance continuous authentication systems. Knowledge-Based Systems, 244, 108544.
[21] Ayotte, B., Banavar, M., Hou, D., Schuckers, S. (2020). Fast free-text authentication via instance-based keystroke dynamics. IEEE Transactions on Biometrics, Behavior, and Identity Science, 2(4), 377-387.
[22] Chen, Z., Cai, H., Jiang, L., Zou, W., Zhu, W., Fei, X. (2021). Keystroke dynamics based user authentication and its application in online examination. In 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD) (pp. 649-654). IEEE.
[23] Nonaka, H., Kurihara, M. (2004). Sensing pressure for authentication system using keystroke dynamics. International Journal of Computational Intelligence, 1(1), 19-22.
[24] Kotani, K., & Horii, K. (2005). Evaluation on a keystroke authentication system by keying force incorporated with temporal characteristics of keystroke dynamics. Behaviour & Information Technology, 24(4), 289-302.
[25] Loy, C. C., Lim, C. P., Lai, W. K. (2005). Pressure-based typing biometrics user authentication using the fuzzy ARTMAP neural network. In Proceedings of the Twelfth International Conference on Neural Information Processing (ICONIP 2005) (pp. 647-652).
[26] Grabham, N. J., White, N. M. (2008). Use of a novel keypad biometric for enhanced user identity verification. In 2008 IEEE Instrumentation and Measurement Technology Conference (pp. 12-16). IEEE.
[27] Tax, D. M., Duin, R. P. (2004). Support vector data description. Machine learning, 54, 45-66.
[28] Song, Y. Y., Ying, L. U. (2015). Decision tree methods: applications for classification and prediction. Shanghai archives of psychiatry, 27(2), 130-135.
[29] Albawi, S., Mohammed, T. A., Al-Zawi, S. (2017). Understanding of a convolutional neural network. In 2017 international conference on engineering and technology (ICET) (pp. 1-6). IEEE
[30] Almalki, S., Assery, N., Roy, K. (2021). An empirical evaluation of online continuous authentication and anomaly detection using mouse clickstream data analysis. Applied Sciences, 11(13), 6083-6107.
[31] O'Shea, K., Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.
[32] Hoo, Z. H., Candlish, J., Teare, D. (2017). What is an ROC curve?. Emergency Medicine Journal, 34(6), 357-359.