Smartphone's Multifactor Authentication Using Secret Pattern and User's Unique Behavioral Habit

Ayahiko Niimi
Faculty of Systems Information Science
Future University Hakodate
Japan

Abstract

In recent years, the widespread adoption of smartphones has significantly heightened the importance of robust security measures. Multifactor authentication has emerged as a critical method to strengthen authentication processes. This paper examines the application of multifactor authentication on smartphones, focusing on a method that integrates three key factors: 1) possession of a smartphone (possession-based information), 2) knowledge of a password or pattern (knowledge-based information), and 3) biometric authentication, such as fingerprint, face, or voice recognition. Given the possibility that a smartphone may already be in unauthorized possession during an access attempt, combining these factors is essential to enhance security. This study proposes an improved approach to the screen unlocking process through enhanced pattern authentication. Typical pattern authentication requires users to trace a sequence of nine points in a specific order to unlock their device. However, this method is inherently vulnerable if an unauthorized party observes and memorizes the pattern. To address this issue, we introduce a machine learning-based solution that enhances the security of pattern authentication by leveraging the unique behavioral traits of the user. This model analyzes not only the sequence of traced points but also the broader characteristics of the user's interaction with the device as a dataset. We developed a data collection application to support this research and validated the proposed method's effectiveness.

1. Introduction

In recent years, the growing prevalence of smartphones has made security measures increasingly critical. Multifactor authentication (MFA) provides a robust method for enhancing security by requiring two or more distinct types of authentication factors from the following categories:

i. Knowledge Information: Something You Know (e.g., passwords or PINs).

ii. Possession Information: Something You Have (e.g., smartphones or security tokens).

iii. Biometric Information: Something You Are (e.g., fingerprints, facial features, or voice patterns).

This paper focuses on the implementation of MFA for smartphones, combining three factors 1) Possession: The smartphone itself, 2) Knowledge: A password or pattern, and 3) Biometric Authentication: Fingerprint, facial recognition, or voice authentication. Since unauthorized access can occur even if a smartphone is physically obtained, employing multiple authentication factors is essential.

Our proposal focuses on improving screen unlocking methods by enhancing traditional pattern authentication. Standard pattern authentication requires users to pre-register a sequence of nine points on the screen, which they trace to unlock it. However, this method is susceptible to breaches if a third party memorizes the pattern. To address this issue, we propose developing a machine-learning model to strengthen pattern authentication by analyzing an individual's unique behavioral traits. The proposed model will examine not only the pattern trajectory but also the entirety of the user's interaction with the device as a dataset. This paper discusses MFA, but Hata's graduate thesis [1] covered the construction of a machine-learning model to identify individual patterns.

2. Related Works

Research on subject identification using freeform patterns includes the work of Junhong Kim [2]. In this study, 40 subjects were asked to freely draw pictures, English letters, numbers, and Korean characters, each repeated 20 times. By analyzing the drawing trajectories, the subjects were successfully identified. The study utilized the Temporal Convolutional Network (TCN), a convolutional network architecture designed for time-series data, which outperforms recurrent networks in many applications [3].

In contrast, our study employs a pattern recognition framework with predefined constraints, including the length of the lines and the specific points to be traversed. These stricter parameters differentiate our approach from Junhong Kim's freeform methodology. While Kim's study required

 | DOI: 10.20533/jitst.2046.3723.2024.0103

20 repetitions of input data per participant, our stricter conditions necessitated 100 repetitions.
The method proposed in this study maintains the same pattern authentication input method used in existing smartphones. This ensures improved authentication performance without causing any inconvenience to users.

3. Proposed Method

This research aims to develop a machine learning model capable of identifying an individual's unique behavioral patterns, to enhance the performance of existing pattern authentication methods. We hypothesize that an individual's unique habits are reflected not only in the trajectory of predefined points but also in the complete operation trajectory, including factors such as operation speed, motion style, and the way curves are. This study seeks to determine whether these tendencies can be learned from actual input data and leveraged to improve authentication accuracy. The proposed methodology consists of the following steps:

i. A dedicated application is used to input patterns and collect data.

ii. A machine learning model is built using the collected data.

iii. The pattern input data is given to the constructed machine learning model and the personal demand rate is evaluated.

Each step is explained in detail below.

Step 1: Data Collection Using a Dedicated Application

The first step involves using a specialized application to input patterns and collect corresponding data. This data includes complete operation trajectories, timing details, and acceleration sensor values. All data is saved in CSV format for further processing.

To capture detailed trajectory information on an Android device, a dedicated API was required, necessitating the development of a specialized data collection application. To meet this need, we created a Kotlin-based application using Android Studio (Figure 1).

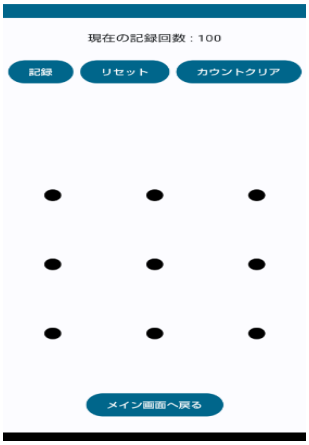

Figure 1. Screen of the Developed Data Collection Application

	A	B	C	D	E	F	G	H	I
1	rawX	rawY	pressure	size	sensorX	sensorY	sensorZ	eventTime	downTime
2	147	1033	+0.09804	0.13333	0.29688	-0.1245	-0.13408	2114613	2114613
3	153	1034	+0.10588	0.13333	0.00958	-0.21069	0.49799	2114678	2114613
4	192.95335	1038.40625	+0.10588	0.13333	0.00958	-0.21069	0.49799	2114700	2114613
5	225.36903	1040	+0.10588	0.13333	0.08619	0.32561	-0.21069	2114717	2114613
6	261.40488	1040.60022	+0.10588	0.13333	0.08619	0.32561	-0.21069	2114734	2114613
7	303.42014	1041.70093	+0.10588	0.13333	0.08619	0.32561	-0.21069	2114750	2114613
8	348.55325	1043.29614	+0.10588	0.13333	0.08619	0.32561	-0.21069	2114767	2114613
9	394.57239	1047.90076	+0.10588	0.13333	0.86191	-0.30646	0.62249	2114783	2114613
10	438.51828	1053.35181	+0.10588	0.13333	0.86191	-0.30646	0.62249	2114800	2114613
11	478.52594	1055	+0.10588	0.13333	0.86191	-0.30646	0.62249	2114817	2114613
12	527.80511	1052.70728	+0.10588	0.13333	0.86191	-0.30646	0.62249	2114833	2114613
13	580.41675	1043.60339	+0.10588	0.13333	-0.19154	-0.62249	1.01514	2114850	2114613
14	635.67523	1032.79321	+0.10588	0.13333	-0.19154	-0.62249	1.01514	2114866	2114613
15	680.20709	1025.11328	+0.10588	0.13333	-0.19154	-0.62249	1.01514	2114883	2114613
16	721.96661	1020.73029	+0.10588	0.13333	-0.19154	-0.62249	1.01514	2114900	2114613
17	764.6731	1018	+0.10588	0.13333	0.6608	0.19154	-0.09577	2114916	2114613
18	803.94696	1017	+0.10588	0.13333	0.6608	0.19154	-0.09577	2114933	2114613
19	834.89294	1018	+0.10588	0.13333	0.6608	0.19154	-0.09577	2114950	2114613
20	858.5274	1020.65271	+0.10588	0.13333	0.6608	0.19154	-0.09577	2114966	2114613

Figure 2. Image of Collected Data (CSV).

During data collection, 60 data points were recorded per second, starting from the moment a tap began until it ended. The collected data was saved in CSV format to facilitate processing. A total of nine attributes were recorded: "tap position (X, Y)," "pressure," "size," "accelerometer (X, Y, Z) ," "tap start time", and "recording time." If certain data

 | DOI: 10.20533/jitst.2046.3723.2024.0103

points were unavailable due to device limitations, the corresponding fields were filled with zeros. (Figure 2). Data collection was performed using an AQUOS sense4; running Android OS version 11.

Step 2: Building a Machine Learning Model Using the Collected Data

In the second step, a machine learning model was developed using Python 3, based on the data collected in step 1. To simplify the process and ensure accessibility, we used sci-kit-learn, an open-source machine-learning library for Python, to construct the model.

The data collected from Step 1 was not suitable for machine learning due to variations in data format. Therefore, preprocessing was performed, including dimensionality adjustments and data normalization, to prepare the dataset for modeling.

Step 3: Evaluating the Constructed Machine Learning Model

In step 3, the machine learning model constructed in step 2 was evaluated by testing it with both correct data (the user) and incorrect data (the attacker). The user acceptance rate and false rejection rate are then calculated and evaluated.

4. Experiments

To determine the parameters most effective for capturing individual behavioral patterns, an evaluation experiment was conducted with three subjects. Each participant entered data into the experimental application using the five predefined patterns shown in Figure 3. For each participant, their input was treated as correct data (input by the user) and incorrect data (input by an attacker). This experiment allowed us to verify whether the machine learning model could accurately identify individuals and serve as an effective identity authentication system. The evaluation experiment included three university students, each of whom entered data into the experimental application based on the five patterns depicted in Figure 3.

Participants were instructed to trace a blue line starting from a star, repeating each pattern 100 times. Input conditions were standardized: participants used only their dominant hand while sitting in a chair (see Figure 4.)

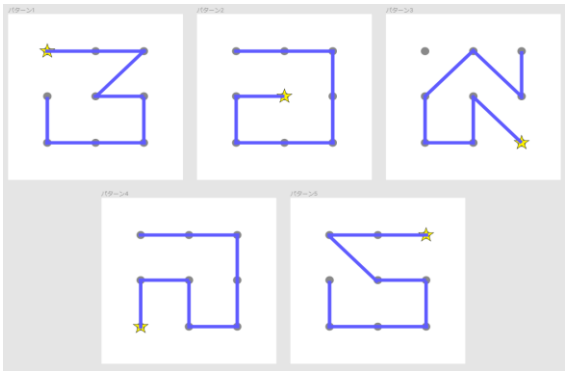

Figure 3. Patterns Used in the Evaluation Experiment.

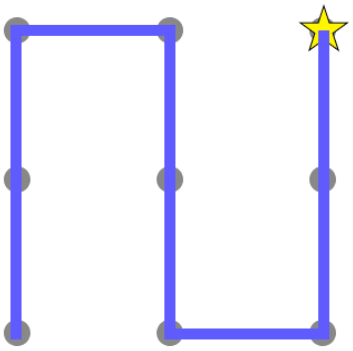

Figure 4. Patterns with stars and blue lines.

The application used in the evaluation experiment included three buttons: "Record," "Reset," and "Clear Count," enabling participants to track their input count during the experiment. The actual screen is displayed in Figure 1.

Since the data collected consisted solely of numerical strings, manual verification of accuracy was not feasible. To address this, we developed a trajectory viewer using Processing. Figure 5 illustrates input being visualized in the trajectory viewer. We verified that the trajectory viewer successfully collected data without issues.

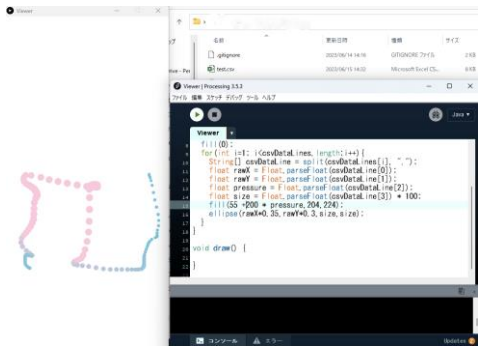

Figure 5. Trajectory Viewer

 | DOI: 10.20533/jitst.2046.3723.2024.0103

Before starting, the procedure was explained to each subject, and their consent to participate was obtained. The five input patterns were introduced, and participants were instructed to enter the corresponding data. Upon successful entry, they pressed the "record" button. In the cases of input error, they pressed the "reset" button to retry. This process continued until 100 successful inputs were recorded for each pattern. Participants repeated this procedure for all five patterns.

During the experiment, the facilitator presented the patterns. After each pattern was completed, they updated the recording file name and reset the count. This marked the conclusion of one cycle of the experiment.

The following three patterns were prepared for the preprocessed dataset.

i. Dataset 1: Outlier removal + naive dimensional expansion + 0 filling.

ii. Dataset 2: Outlier removal + linear interpolation + sorting.

iii. Dataset 3: Outlier removal + linear interpolation + normalization

In all datasets, outliers were removed. Outliers were defined as inputs lasting only 1 or 2 consecutive frames, typically caused by incorrect tapping operations instead of the intended swiping for pattern input. Such erroneous inputs were excluded from the dataset due to their unusable nature.

Furthermore, we focused on seven parameters: rawX, rawY, pressure, size, sensorX, sensorY, and sensorZ. The event time and downtime were omitted to reduce dimensionality, with input timing inferred from the quantity of data (i.e., the number of dimensions).

Dataset 1: Dataset 1 was subjected to naive dimensional expansion and 0-padding performed in the evaluation experiments.

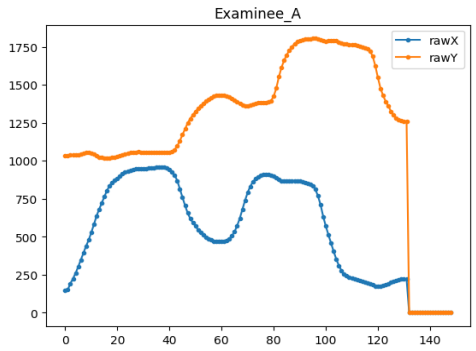

Figure 6. Dataset 1: rawX and rawY for pattern 1, first input by subject A.

The actual visualization of Dataset 1 is presented in Figures 6, 7, and 8. For simplicity, only the numerical values of rawX and rawY from the first round of input for pattern 1 (Figure 3) are visualized.

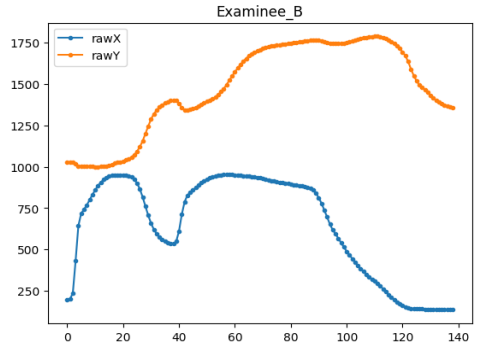

Figure 7. Dataset 1: rawX and rawY for the first input of pattern 1 by subject B.

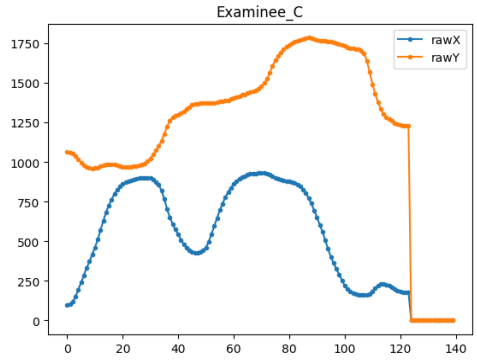

Figure 8. Dataset 1: rawX and rawY for pattern 1, first input by subject C.

The characteristics of Dataset 1 are as follows:

i. Due to the naive dimensional expansion, the data order is not uniform.

ii. The number of dimensions for a single input varies depending on the input duration.

iii. For inputs with shorter duration, the latter half of the parameters is padded with zeros, particularly noticeable in the inputs from subjects A and C (Figure 6, 8).

The inconsistency in data order is shown in Figure 9. Specifically, the data is arranged under labels such as x1, y1, pressure1, size1, sX1, sY 1, sZ1, x2, y2, pressure2, size2, sX2, sY 2, sZ2.... In addition, because the number of dimensions varied, zero-padding was applied was applied to align all inputs to the maximum dimensionality observed among subjects A, B, and C. This standardized dataset was then used this data in the machine learning model.

 | DOI: 10.20533/jitst.2046.3723.2024.0103

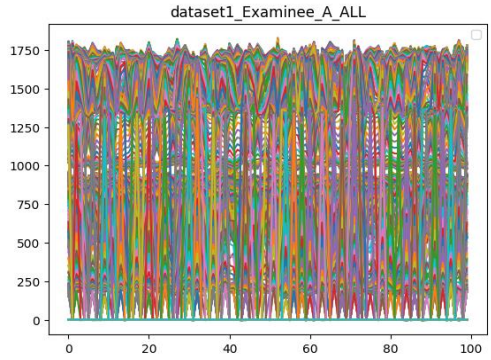

Figure 9. Data is not organized.

Dataset 2: In Dataset 2, in addition to removing outliers, the dimensionality was adjusted by applying linear interpolation to each parameter. This resulted in a dataset with 700 dimensions, comprising 100 dimensions per parameter. The actual number of dimensions adjusted is illustrated in Figures 10, 11, and 12.

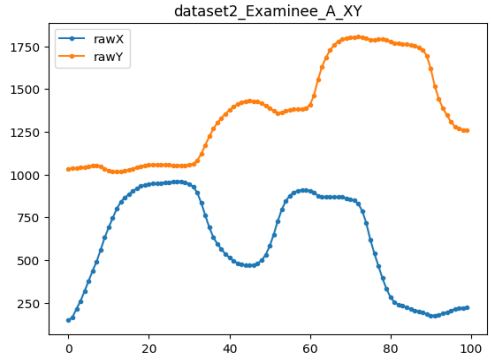

Figure 10. Dataset 2: rawX and rawY for the first input of pattern 1 by subject A.

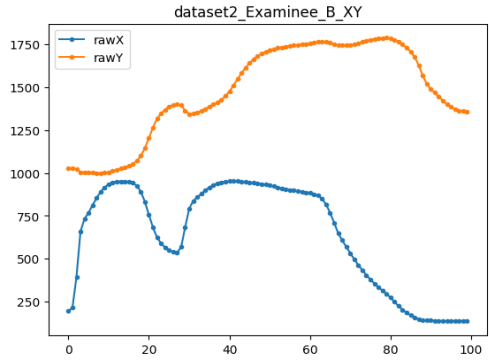

Figure 11. Dataset 2: rawX and rawY for the first input of pattern 1 by subject B.

The characteristics of Dataset 2 are as follows:

i. The parameters were reorganized by data type as rawX, rawY, .., resulting in a structured and coherent data order.

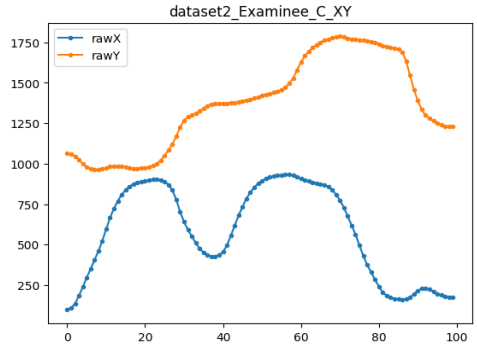

Figure 12. Dataset 2: rawX and rawY for the first input of pattern 1 by subject C.

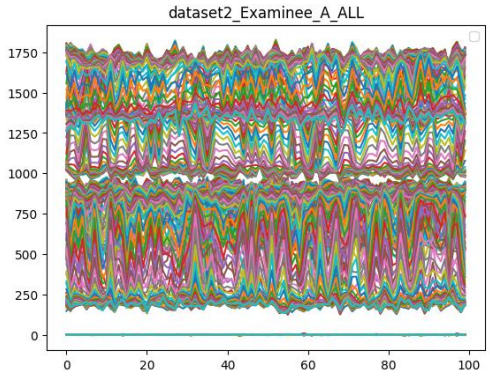

Figure 13. A diagram showing the data coherently.

ii, Linear interpolation was applied to unify the number of dimensions to 700. You can see in Figure 13 that the data order is consistent. Specifically, the data is arranged under the labels x1, x2, x3, ..., x100, y1, y2,...

iii. Dataset 3: Dataset 3 is the same as Dataset 2, except that the scale of the data is adjusted by normalizing rawX and rawY.

5. Experimental Results And Evaluation

The machine learning algorithms utilized in this study include the One-Class Support Vector Machine (SVM), Random Forest, and Isolation Forest. One-class SVM was selected for its simplicity and ease of application, while Random Forest and Isolation Forest were chosen based on their proven effectiveness in related research.

Technologies Used for Evaluation

The One-Class SVM is a variant of the SVM model. While traditional SVM is a supervised learning algorithm primarily used for classification

 | DOI: 10.20533/jitst.2046.3723.2024.0103

tasks, the One-Class SVM is designed for unsupervised learning, specifically for anomaly detection [4]. It is widely applied in outlier classification and was selected in this study for its simplicity and ease of use in unsupervised learning.

Random Forest is an algorithm that integrates two techniques — decision trees and ensemble learning (bagging) — and is primarily employed for classification and regression tasks [5]. In this study, Random Forest was utilized specifically for classification purposes.

Decision tree learning is a method for representing knowledge about target attributes in data as a set of rules in a tree structure [6]. In the resulting decision trees, branching nodes represent condition attributes, branches denote outcomes, and terminal leaves correspond to target attributes. A key advantage of decision trees is their interpretability and readability, as the tree structure explains the decision-making process.

Isolation Forest, similar to Random Forest, is constructed using decision trees. It is particularly suitable for handling time-series data and identifying outliers [7]. In this study, we adopted Isolation Forest for its ability to process time-series data using unsupervised learning while effectively classifying outliers.

To evaluate the performance of our machine learning models, we employed k-fold cross-validation. This method divides the dataset into k subsets, using one subset as test data while training on the remaining subsets. This process is repeated k times to ensure a comprehensive evaluation of the model's performance [8]. In this study, k-fold cross-validation was utilized to rigorously assess the performance of the constructed models.

Experimental Results

The One-Class SVM was trained using the defender's input, it was expected to output −1 (indicating an anomaly) or 1 (indicating normal input). However, the model failed to function correctly across all datasets. Adjustments to the kernel, nu parameter, or gamma parameter did not improve its performance. Even after applying k-Fold Cross Validation on the generated dataset, the model consistently output was−1 (abnormal value), suggesting that learning did not occur properly.

Random Forest was trained using normal data to construct the model also failed to perform as expected on any datasets. Modifications to kernel settings and random parameters did not affect the outcome. As with the One-Class SVM, k-fold cross-validation was applied, yet the model consistently produced an output of 0 (normal value), indicating that training was not successfully executed.

Given that both the One-Class SVM and Random Forest models failed to train effectively, this is likely due to the high dimensionality and the complexity of handling time-series data. Consequently, we turned to Isolation Forest, a machine learning algorithm capable of unsupervised learning and well suited for processing time-series data. Isolation Forest has been referenced in related research, further supporting its use in this study.

To evaluate whether the input originated from the user, the first 70 of 100 inputs were designated as training data for the machine learning model, with the remaining 30 inputs used as test data for evaluation. However, due to the removal of outliers and erroneous data during preprocessing, the total number of usable inputs was reduced to fewer than 99, with test data consisting of 24 or 30 entries in some cases.

For this experiment, we utilized the default parameters provided by the sklearn library without making adjustments.

Table 1 presents the performance of each dataset when applying Isolation Forest to Pattern 1. In the table, A represents the acceptance rate for the actual user, while A patterns 2, B, and C denote the rejection rates for other users. The false acceptance rate remained consistent across all datasets, but differences were observed for other input patterns. Dataset 2 showed the best performance when evaluating the true user's alternate patterns, while Dataset 1 was most effective for Subject B's input, and Dataset 2 excelled for Subject C's input. For Dataset 3, the false acceptance and rejection rates for the true user's alternate patterns were reasonable. However, the false rejection rates for inputs from subjects B and C were below 13%, indicating lower effectiveness compared to other datasets.

Table 1. Performance of each dataset in Pattern 1

	A	A pattern 2	B	C
Dataset 1	93.3%	43.8%	92.5%	6.1%
Dataset 2	93.3%	100%	53.1%	25.7%
Dataset 3	93.3%	67.3%	4.25%	12.3%

Based on these results, Dataset 2 emerged as the most effective, prompting further evaluation of each pattern's performance using this dataset. The evaluation metrics are summarized in Tables 2 and 3. In these tables, A represents the acceptance rate for the target person, B indicates the rejection rate for others, C shows the rejection rate for other individuals, and BC composite denotes the combined rejection rate for non-target individuals.

Using Dataset 2, inputs from the original individuals were correctly identified with a probability exceeding 88.0%. However, inputs from subject B resulted in a false rejection rate of 50% for patterns 1, 2, and 4, which decreased to below 17.0% for patterns 3 and 5. Similarly, inputs from subject C showed a false rejection rate of 62% for patterns 4 and 5, but this dropped to 1.0% for pattern 3.

 | DOI: 10.20533/jitst.2046.3723.2024.0103

Considering the overall rejection rates for subjects B and C, pattern 3 demonstrated the poorest performance.

Table III also details the false rejection rates when subject A entered a different pattern. In most pattern combinations, the false rejection rate exceeded 90%; however, for patterns 1 and 5, the false rejection rates were below 90%.

6. Discussions

Selection of Machine Learning Algorithms

This study employed three machine learning algorithms: One-Class SVM, Random Forest, and Isolation Forest. One-class SVM was chosen for its ease of implementation, while Random Forest and Isolation Forest were selected due to their proven effectiveness in research. As a result, there was no special reason for not employing deep learning or other machine learning algorithms in this study. We plan to explore and discuss additional machine-learning models in future work.

Model-dependent issues

In this study, AQUOS sense4 was used as the data acquisition device for all experiments, operating on Android OS version 11. Consequently, the results may be influenced by model-dependent factors. The amount and quality of data collected are affected by variations in screen resolution and size. Addressing these model-dependent issues remains a challenge for future research.

Table 2. Performance of Each Pattern (Dataset 2)

	A	B	C	BC composite
Pattern 1	93.3%	53.1%	25.7%	39.2%
Pattern 2	100%	99.0%	35.0%	67.6%
Pattern 3	100%	9.4%	1.0%	5.5%
Pattern 4	96.1%	64.9%	77.0%	72.5%
Pattern 5	88.0%	17.0%	62.8%	41.8%

Table 3. Rejection Rate of Other People When Subject A Inputs a Different Pattern (Dataset 2)

	Pattern 1	Pattern 2	Pattern 3	Pattern 4	Pattern 5
Pattern 1	-	100%	100%	1.000000	0.768000
Pattern 2	100%	-	100%	1.000000	1.000000
Pattern 3	100%	96.9%	-	0.989000	1.000000
Pattern 4	98.0%	90.8%	98.8%	-	1.000000
Pattern 5	57.9%	91.8%	100%	0.906000	-

Issue of needing a large amount of input data for learning

Currently, a significant challenge is the need for a large volume of input data for training. In the pre-evaluation experiment, participants were required to input the correct answer pattern 100 times to generate the correct data for building the machine-learning model. Such a requirement is not suitable for real-world systems. One potential solution is the introduction of fine-tuning. Fine-tuning is a transfer learning technique that trains the weights of a pre-trained model with new data, reducing the need for extensive input.

In this study, we did not address this issue, as our primary goal was to develop a machine-learning model capable of verifying a person's identity. However, we believe that incorporating fine-tuning will be essential for real-world applications of the system.

Parameter Tuning

Parameter tuning was not conducted on the Isolation Forest, the primary method used for evaluation in this study. However, parameter tuning is expected to further enhance performance. Optimizing authentication performance through parameter tuning remains a challenge for future work.

7. Conclusions

This paper focuses on improving multifactor authentication for smartphones by leveraging a machine learning model that identifies individual user habits to improve the performance of pattern authentication. We implemented the proposed approach and validated its effectiveness through user experiments.

Future challenges include optimizing the selection of machine learning algorithms, addressing model-specific limitations, and incorporating fine-tuning techniques for further performance enhancement.

8. References

[1] Hata, D. (2023). Enhanced authentication performance in smartphone pattern authentication, Graduation Thesis,

 | DOI: 10.20533/jitst.2046.3723.2024.0103

School of Systems Information Science, Future University Hakodate. (In Japanese.).

[2] Kim J., and Kang, P. (2022). Draw-a-deep pattern: Drawing pattern-based smartphone user authentication based on temporal convolutional neural network," Applied Sciences, vol. 12, no. 15, p. 7590. https://www.mdpi.com /2076-3417/12/15/7590 (Access Date: 11 September 2024).

[3] Bai, S., Kolter, J. Z., and Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, CoRR, vol. abs/1803.01271. http://arxiv.org/abs/1803. 01271 (Access Date: 9 September, 2024).

[4] SKLearn. (n.d,). OneClassSVMO—scikit-learn 1.5.2 docume- ntation. https://scikit-learn.org/stable/modules/ge nerated/sklearn.svm.OneClassSVM.html (Access Date: 13 September, 2024).

[5] SKLearn. (n.d,). RandomForestClassifier—scikit-learn 1.5.2 documentation. https://scikit-learn.org/stable/module s/generated/sklearn.ensemble.RandomForestClassifier.htm l (Access Date: 13 September, 2024).

[6] SKLearn. (n.d,). DecisionTreeClassifier—scikit-learn 1.5.2 documentation. https://scikit-learn.org/stable/module s/generated/sklearn.tree.DecisionTreeClassifier.html (Acce -ss Date: 13 September, 2024).

[7] SKLearn. (n.d,). IsolationForest—scikit-learn 1.5.2 do -cumentation. https://scikit-learn.org/stable/modules/gener ated/sklearn.ensemble.IsolationForest.html (Access Date: 13 September, 2024).

[8] SKLearn. (n.d,). 3.1. Cross-validation: evaluating est -imator performance = scikit-learn 1.5.2 documentation. ht tps://scikit-learn. org/stable/modules/cross validation.html (Access Date: 13 September, 2024).

 | DOI: 10.20533/jitst.2046.3723.2024.0103