applied sciences

Article

Real-Time Mouse Data Protection Method Using GANs for Image-Based User Authentication Based on GetCursorPos() and SetCursorPos() Functions

Jinwook Kim [1], Kyungroul Lee [2] and Hanjo Jeong [3,*]

[1] Interdisciplinary Program of Information & Protection, Mokpo National University, Muan 58554, Republic of Korea; wlsdnr0816@mokpo.ac.kr
[2] Department of Information Security Engineering, Mokpo National University, Muan 58554, Republic of Korea; carpedm@mnu.ac.kr
[3] Department of Software Convergence Engineering, Mokpo National University, Muan 58554, Republic of Korea
* Correspondence: hanjojeong@mnu.ac.kr; Tel.: +82-61-450-2773

Abstract: In online services, password-based authentication, a prevalent method for user verification, is inherently vulnerable to keyboard input data attacks. To mitigate these vulnerabilities, image-based authentication methods have been introduced. However, these approaches also face significant security challenges due to the potential exposure of mouse input data. To address these threats, a protective technique that leverages the SetCursorPos() function to generate artificial mouse input data has been developed, thereby concealing genuine user inputs. Nevertheless, adversaries employing advanced machine learning techniques can distinguish between authentic and synthetic mouse data, leaving the security of mouse input data insufficiently robust. This study proposes an enhanced countermeasure utilizing Generative Adversarial Networks (GANs) to produce synthetic mouse data that closely emulate real user input. This approach effectively reduces the efficacy of machine learning-based adversarial attacks. Furthermore, to counteract real-time threats, the proposed method dynamically generates synthetic data based on historical user mouse sequences and integrates it with real-time inputs. Experimental evaluations demonstrate that the proposed method reduces the classification accuracy of mouse input data by adversaries to approximately 62%, thereby validating its efficacy in strengthening the security of mouse data.

Keywords: image-based authentication; mouse data; SetCursorPos() function; generative adversarial network(GANs); machine learning

check for updates

Academic Editors: Luis Javier García Villalba and Ana Lucila Sandoval Orozco

Received: 14 December 2024
Revised: 3 January 2025
Accepted: 16 January 2025
Published: 20 January 2025

Citation: Kim, J.; Lee, K.; Jeong, H. Real-Time Mouse Data Protection Method Using GANs for Image-Based User Authentication Based on GetCursorPos() and SetCursorPos() Functions. *Appl. Sci.* **2025**, *15*, 977. https://doi.org/10.3390/app15020977

1. Introduction

With the advent of the digital age, user authentication technologies have become indispensable for accessing online services [1]. While offline identity verification often relies on tangible credentials such as Social Security Numbers (SSN), online environments necessitate remote and contactless authentication methods due to the absence of direct verification mechanisms. Among these, password-based authentication remains the most widely employed technique due to its ease of implementation [2,3]. In this framework, users create their passwords during registration, and authentication is performed by matching the entered password with the stored credentials. Typically, passwords are entered via keyboard input, rendering keyboard data a critical asset that must be securely protected [4].

Despite its prevalence, password-based authentication is vulnerable to keyboard data attacks aimed at intercepting credentials. Attackers have successfully intercepted keyboard data using C/D-bit weakness and the RESEND command, leveraging the fact that PS/2 and USB interface keyboards were designed without robust security considerations [5]. As a result, these devices remain susceptible to hardware-level attacks. Additionally, attackers deploy a range of strategies across application, operating system, and hardware layers, complicating comprehensive defensive measures. Consequently, password-based authentication exhibits fundamental limitations in maintaining security, prompting the exploration of alternative approaches such as image-based authentication [6,7].

Image-based authentication technologies enable users to input passwords by interacting with graphical images displayed on the screen, circumventing the need for keyboard input and thus mitigating risks associated with keyboard data attacks. However, these systems are not immune to exploitation. Specifically, if screen and mouse data are inadequately protected, attackers can extract authentication-related information. A notable vulnerability lies in the operating system's mouse position management, where attackers can track cursor movements by repeatedly invoking the GetCursorPos() function [8]. This highlights a significant security gap in image-based authentication systems. To address this vulnerability, defensive techniques utilizing the SetCursorPos() function [9] have been developed to generate synthetic mouse data, thereby obfuscating real input and confusing attackers. However, recent advancements in machine learning-based attack methods have enabled attackers to distinguish between real and synthetic mouse data by identifying patterns within the generated inputs, achieving classification accuracies of up to 99% [10,11].

This paper proposes a novel approach to enhance the protection of mouse data in image-based authentication systems by leveraging Generative Adversarial Networks (GANs) [12]. The proposed method generates synthetic mouse data that closely resemble authentic user input, thereby diminishing the effectiveness of machine learning-based attacks. Furthermore, the system dynamically generates fake data in real time by training GANs on historical user mouse movement data and blending the synthetic data with live inputs. Experimental results demonstrate that the proposed technique reduces the classification accuracy of adversarial attacks from 99% to approximately 62%, thereby validating its effectiveness in securing mouse data. The contributions of this study are as follows:

- The proposed technique addresses the shortcomings of existing mouse data protection methods, thus improving the security of image-based authentication systems by significantly reducing the success rates of machine learning-based attacks.
- A dataset of real mouse movement data was constructed to facilitate the generation of realistic synthetic data using GANs. This dataset can be continuously updated with additional user data, enabling the production of increasingly sophisticated synthetic inputs to enhance defensive performance.
- By leveraging historical user mouse movement sequences, the proposed system pre-generates fake data, which is then mixed with live input to provide real-time protection against active attacks.
- The experimental results demonstrate that the proposed technique effectively reduces attack accuracy, dropping from 99% to an average of 62%. This is close to the statistically insignificant baseline of 50% in binary classification tasks distinguishing real from fake mouse data. This indicates the method's success in obscuring critical patterns necessary for extracting sequential mouse coordinates, thereby ensuring the protection of sensitive authentication data.

The remainder of this paper is organized as follows: Section 2 reviews related works on mouse data attacks and existing protection techniques. Section 3 details the methodology and simulation environment of the proposed system. Section 4 analyzes the experimental results, and Section 5 concludes the study with a discussion of its implications and future research directions.

2. Related Research

2.1. Generative Adversarial Networks (GANs)

To counter machine learning-based mouse data attacks, this study proposes a mouse data protection technique utilizing Generative Adversarial Networks (GANs). GANs, first introduced in 2014, comprise two neural networks—a generator and a discriminator—that engage in a competitive learning process. The generator creates synthetic data that mimic real data, while the discriminator's task is to distinguish between the real and generated data. This iterative competition improves the performance of both networks, with the goal of generating highly realistic fake data. GANs are commonly employed in applications such as image generation and data augmentation, particularly when there is a lack of sufficient training data for machine learning models [12,13].

Generative Adversarial Networks (GANs) are increasingly being applied in the field of information security, particularly in generating synthetic attack data and enhancing security systems. One such study investigates how GANs can be utilized in cybersecurity to create synthetic attack data, and this is crucial for training intrusion detection systems. Since acquiring realistic cyberattack data can be challenging due to privacy concerns, GANs provide a solution by generating diverse and realistic attack data that simulate a wide range of cyber threats. This synthetic data can be employed to improve the training of security models without compromising the privacy of real-world data [14–16]. Another application of GANs in information security is enhancing encryption methods for digital image protection. By using GANs to generate encrypted versions of images, researchers aim to make it more difficult for unauthorized users to access sensitive information. This approach also incorporates a customized super-resolution network for reconstructing these images, ensuring secure data handling even when faced with potential adversarial attacks [17,18]. These studies underscore the growing interest in leveraging GANs to improve data security, particularly in areas like attack simulation and encryption, where the generation of synthetic data plays a critical role in advancing cybersecurity defense mechanisms.

2.2. Mouse Data Attacks and Protection Techniques Based on Windows API

The Windows API (Application Programming Interface) [19] provides a collection of functions that enable applications to interact with hardware and software components within the Microsoft Windows operating system. Among these, two critical mouse-related functions, GetCursorPos() and SetCursorPos(), are utilized to manage mouse coordinates. These functions, defined in the winuser.h header file, enable the retrieval of the current mouse cursor position (GetCursorPos()) and the movement of the cursor to specified coordinates (SetCursorPos()). Traditional mouse data attacks involve attackers repeatedly invoking GetCursorPos() to track mouse movements. In response to this, defensive techniques have been developed, where the mouse cursor is periodically repositioned to random coordinates, thereby disrupting the attacker's ability to track real mouse positions. Specifically, for a set of real mouse coordinates (A_1, A_2, …, A_n), random synthetic coordinates (B_1, B_2, …, B_n) are generated using SetCursorPos(), resulting in the collection of both real and fake coordinates (e.g., A_1, B_1, B_2, A_2, …, B_n, …, A_n). Since attackers lack knowledge of the fake coordinates, they are unable to distinguish between real and fake data, preventing the extraction of true mouse data.

However, this randomization technique can be circumvented if attackers also collect the timing information associated with each coordinate collection. By analyzing the temporal distribution between real, non-periodic coordinates and the periodic fake ones, attackers can identify patterns and use these distinguishing features to classify the coordinates effectively [10,11].

2.3. Machine Learning-Based Mouse Data Attack Techniques

To overcome defenses based on SetCursorPos() that generate fake mouse data, machine learning-based attack techniques have emerged. In these attacks, attackers collect both real and fake mouse data by periodically calling GetCursorPos(). Then, they apply various machine learning algorithms, such as K-Nearest Neighbors (KNN), Logistic Regression, and Support Vector Machines (SVM), to classify the data. Key features, such as the X and Y coordinates of the mouse and the elapsed time between movements, are used to construct the dataset. Results from applying these machine learning models show that classification accuracy can reach up to 98% [10]. Furthermore, by incorporating additional features, such as the distance between consecutive mouse coordinates, classification accuracy can be improved to 99% [11]. These findings suggest that simply using SetCursorPos() to generate random fake data is insufficient against machine learning-based attacks, necessitating the development of more sophisticated techniques for generating fake data to protect mouse data effectively.

3. Proposed Method for Real-Time Mouse Data Protection Using Generative Adversarial Networks (GANs)

In this section, we propose a method to protect real-time mouse data from machine learning-based attacks. To prevent attackers from accurately classifying mouse data, synthetic data closely resembling real data must be generated. The following subsections describe the data generation methodology, dataset construction, and experimental results from the attacker's perspective.

3.1. Overall Flow of Real-Time Mouse Data Protection Method Using GANs

This subsection outlines the methodology for protecting mouse data using GANs in an image-based authentication system. Figure 1 shows the flowchart illustrating how GANs protect real-time mouse data. The blue and red colors in the figure represent the generation and storage flows of real mouse data and fake mouse data within the system, respectively. The defense tool collects mouse sequence data from a user when they interact with an image-based authentication webpage, applying GANs to generate synthetic mouse sequence data for each authentication page. If the user revisits the authentication page, the defense tool integrates the synthetic data by using the SetCursorPos() function, making the cursor movement reflect the generated fake data.

This process naturally mixes real and synthetic mouse sequence data. If an attacker collects data using the GetCursorPos() function, they will capture both real and synthetic data that closely resemble real mouse data. Unlike random methods of generating fake data, the proposed method uses previous mouse sequence data, trained using the Conditional Tabular Generative Adversarial Network (CTGAN) [20,21] to generate synthetic data. There are certain constraints to consider when generating synthetic fake data using CTGAN. First, generating synthetic data with CTGAN requires training time. In our experiment, it takes approximately 1 to 2 min to generate synthetic data for a 5 to 10-s image authentication process, performed on a machine equipped with an NVIDIA GeForce RTX 4090 GPU using CUDA, which is produced in Taipei, Taiwan. More importantly, the entire mouse movement sequence data from the authentication process is necessary, as adversarial learning networks require complete data during the initial training. Consequently, real-time generation is not feasible.

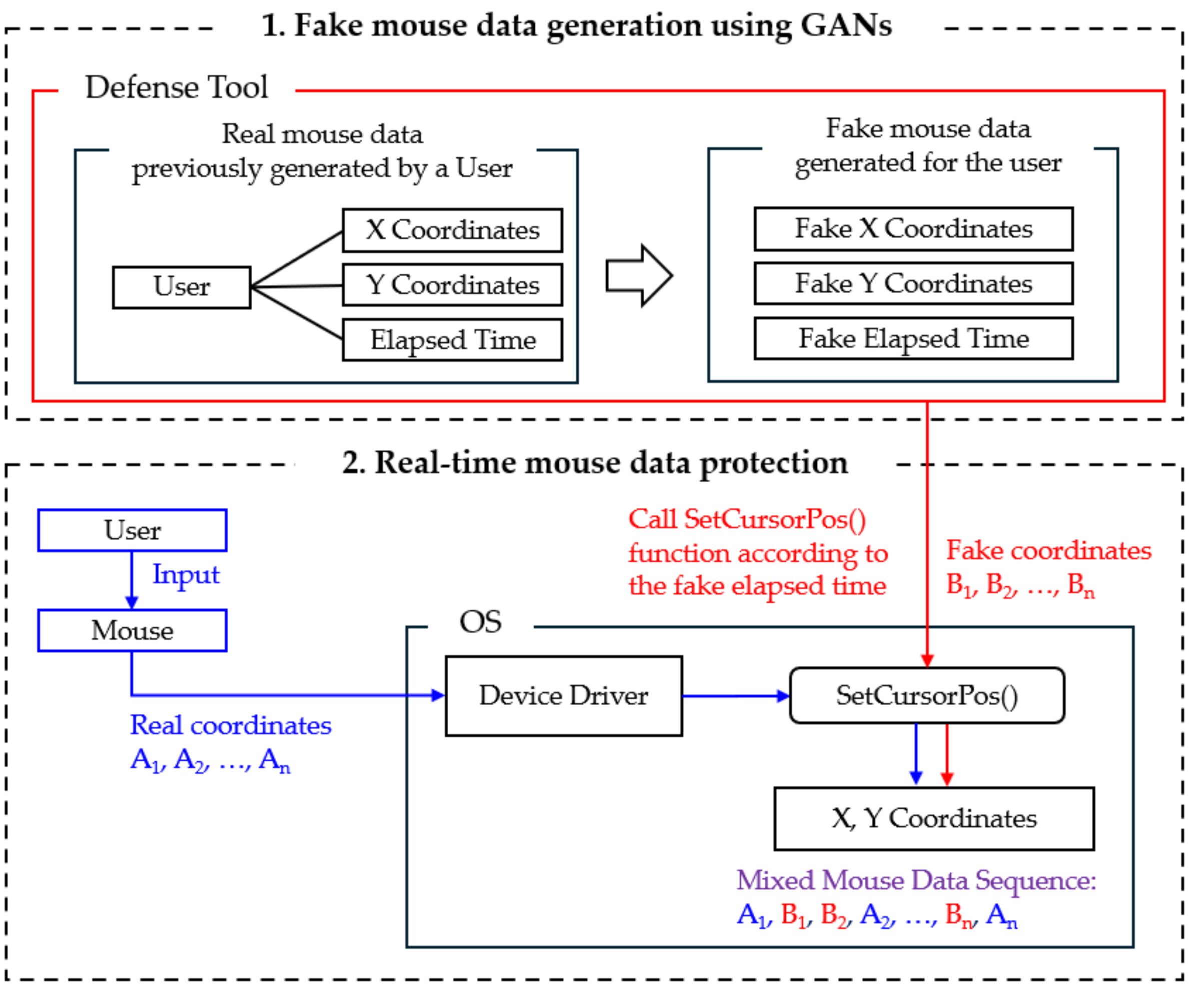
1. Fake mouse data generation using GANs
Defense Tool
Real mouse data previously generated by a User
User
X Coordinates
Y Coordinates
Elapsed Time
Fake mouse data generated for the user
Fake X Coordinates
Fake Y Coordinates
Fake Elapsed Time
2. Real-time mouse data protection
User
Input
Mouse
Real coordinates A1, A2, …, An
Call SetCursorPos() function according to the fake elapsed time
Fake coordinates B1, B2, …, Bn
OS
Device Driver
SetCursorPos()
X, Y Coordinates
Mixed Mouse Data Sequence: A1, B1, B2, A2, …, Bn, An

Figure 1. Overall flow of the proposed method for real-time mouse data protection using GANs.

To address this limitation, this paper proposes pre-generating synthetic data by training on historical mouse sequence data for each user and authentication page, which is then blended with current mouse sequence data. This approach effectively counters real-time data theft attacks.

3.2. Dataset Configuration

In this paper, to demonstrate the effectiveness of the proposed mouse data protection method in image-based authentication, we simulated realistic attacks by having the same user attempt authentication twice on four types of image-based authentication pages. During authentication, mouse movement data were collected using the GetCursorPos() function. The dataset was created by removing any data points where the mouse coordinates did not change from the previous ones. Table 1 shows the number of sequences with actual movement. The dataset is labeled with a numbering system indicating the sequence and authentication page type. For example, Dataset A-1 and A-2 represent the mouse sequences from the first and second attempts on the same page.

Each dataset consists of elapsed time, X-coordinate, and Y-coordinate features, all sorted by the sequence of mouse movements. Elapsed time refers to the time taken to move from the previous coordinate to the current one, and the X and Y coordinates refer to the absolute mouse positions on the screen. To ensure equal importance for each feature, all values were normalized to a range of 0–1 using the min–max normalization. Table 2 shows a sample of the normalized Dataset A-1.

Table 1. Dataset configuration and the number of collected data for real mouse movements.

Dataset	Data Collection Details	Number of Collected Movement Data
Dataset A-1	Collected from the first experiment on page A	16,623
Dataset A-2	Collected from the second experiment on page A	16,628
Dataset B-1	Collected from the first experiment on page B	11,364
Dataset B-2	Collected from the second experiment on page B	11,365
Dataset C-1	Collected from the first experiment on page C	7947
Dataset C-2	Collected from the second experiment on page C	7952
Dataset D-1	Collected from the first experiment on page D	10,074
Dataset D-2	Collected from the second experiment on page D	10,147

Table 2. Sample data of dataset A-1.

Seq#	Elapsed Time	X-Coordinate	Y-Coordinate
1	0.117283	0.431419	0.580363
2	0.195521	0.430637	0.581614
3	0.194678	0.429465	0.585366
4	0.184249	0.509965	0.205128
5	0.1165	0.506839	0.213258
...	...	...	...
16,621	0.145252	0.953497	0.545966
16,622	0.468025	0.032435	0.446529
16,623	0.508667	0.111762	0.347717

'...' represents the omitted intermediate data from seq#6 to seq#16,620.

3.3. Generation of Fake Mouse Data Using GANs

In generating fake mouse data, to obscure the detection of continuous mouse movements, the included features cover not only the X and Y coordinates of the actual mouse but also the elapsed time. Previous studies have shown that adding features such as elapsed time and distance significantly improves the detection performance of fake data from the attacker's perspective [10,11]. This might be because the combination of elapsed time and distance features encapsulates inherent characteristics, such as movement speed, and these are important for distinguishing real mouse movements from randomly generated ones.

Therefore, in this study, the feature set includes not only the X and Y coordinates of the mouse sequence data but also the elapsed time. To generate realistic mouse data based on this tabular, continuous-feature dataset, we employ the CTGAN model [20,21]. The CTGAN model extends the CGAN model by proposing a method specialized in generating tabular data that includes both categorical and continuous variables. In particular, it effectively addresses the sparsity of both types of variables using techniques such as conditional sampling and mini-batch discrimination [20,22,23]. These methods allow the model to handle the sparsity of small values in continuous variables, such as mouse coordinates and elapsed time, and generate them accurately. When generating fake data using the CTGAN model, the distance feature is excluded because, even if included, attackers can use the X and Y coordinates of the mouse to calculate the actual distance and use this information for detection. Thus, including the distance feature becomes redundant. Figure 2 compares the distribution of fake mouse data generated by the CTGAN model using X-coordinate, Y-coordinate, and elapsed time features against the distribution of actual data from Dataset A-1. The comparison is based on the distribution of real and fake data.

The upper distribution in Figure 2 shows the X and Y coordinate distributions of randomly generated fake mouse data and real mouse data with elapsed time as the axis. The lower distribution shows the fake mouse data generated using CTGAN. In the distribution, blue represents real mouse data, while red represents fake mouse data. The X-axis represents the mouse's X and Y coordinates, and the Y-axis represents the elapsed time. As shown in the distribution, the fake mouse data generated by CTGAN closely resemble the real mouse data distribution, unlike the randomly generated fake data.

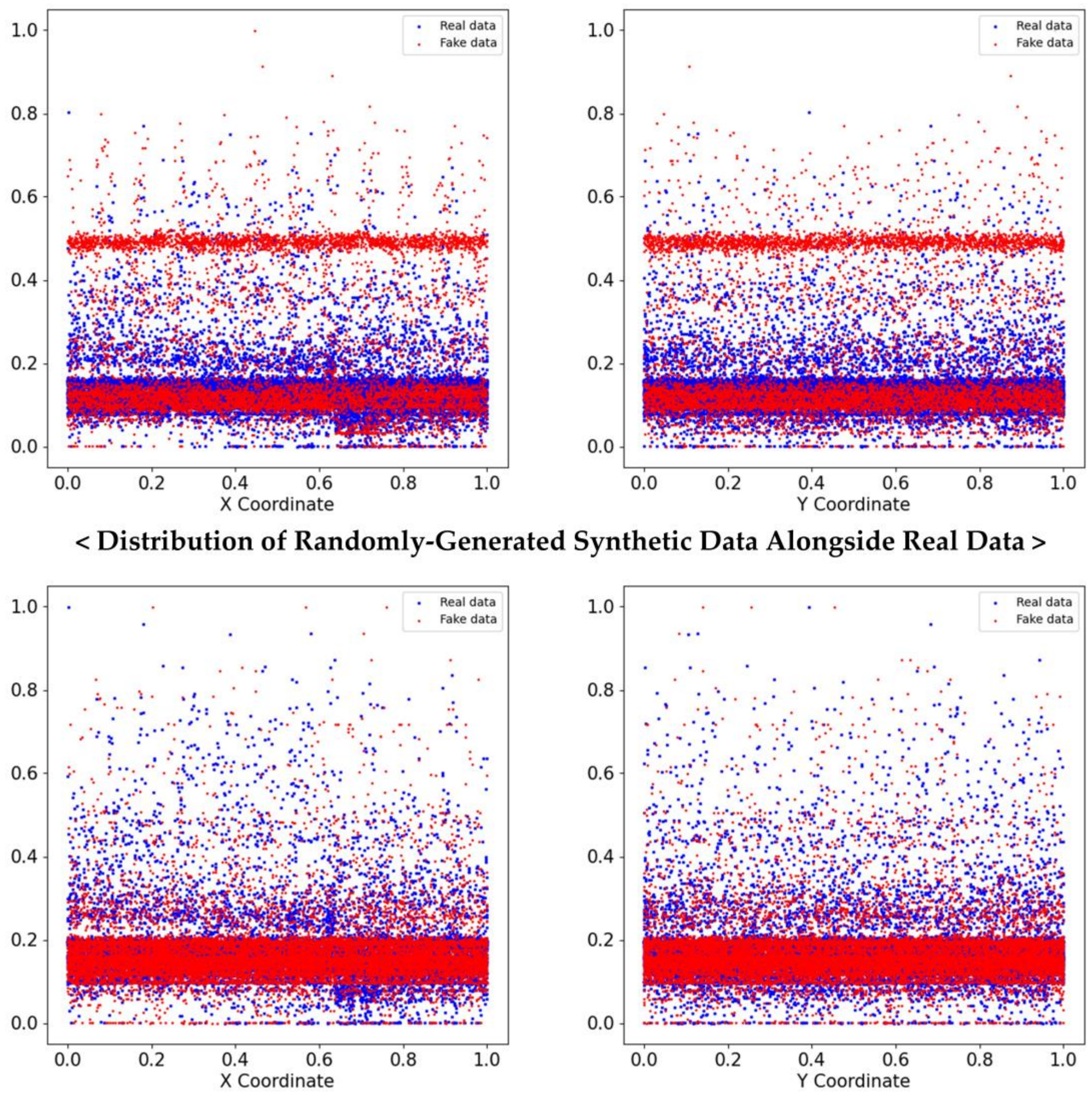

Figure 2. Comparison of random and GAN-based fake mouse data generation results.

3.4. Simulation of Fake Data Classification from the Attacker's Perspective

In this section, simulations were conducted assuming that attackers use various machine learning models to classify real and fake data. The attackers might also generate fake data using random methods or the GAN-based technique proposed in this paper and then train a model to classify the fake mouse data. To simulate this, machine learning models such as K-Nearest Neighbor, Logistic Regression, Random Forest, Gradient Boosting, Support Vector Machine, and Multi-Layer Perceptron were used. The dataset used for the simulation was collected by attempting authentication twice on the same image-based authentication page. Data from the first and second attempts were paired together to simulate the classification from an attacker's perspective. Additionally, real and fake data were merged based on the collection time and re-ordered using cumulative elapsed time to simulate how an attacker might combine data. Table 3 shows a sample of the mixed mouse sequence data generated for Image Authentication Page A. The data include fake mouse data generated by CTGAN using the actual mouse sequence data from the first trial (Dataset A-1) and the actual data from the second trial (Dataset A-2).

Table 3. Sample data of the mixed dataset of dataset A-2 and synthetic data created based on dataset A-1.

Seq#	Elapsed Time	X	Y	X Distance	Y Distance	Class
1	0.143747	0.431419	0.580363	0	0	1
2	0.085557	0.794842	0.791745	0.36385	0.212846	0
3	0.104029	0.782337	0.160725	0.01252	0.63539	0
4	0.050052	0.430637	0.581614	0.352113	0.423804	1
5	0.154898	0.451348	0.595372	0.020736	0.013854	0
. . .	. . .	. . .	. . .	. . .	. . .	. . .
33,249	0.18158	0.415006	0.377111	0.071596	0.627204	1
33,250	0.184869	0.415006	0.377736	0	0.00063	1
33,251	0.188698	0.415006	0.378361	0	0.00063	1

'. . .' represents the omitted intermediate data from seq#6 to seq#33,248.

In Table 3, the elapsed time is calculated based on the cumulative elapsed time of the two-sequence data, which are integrated into a single sequence and then recalculated. X Distance and Y Distance refer to the distances moved from the previous sequence's X and Y coordinates. Lastly, the Class column indicates whether the data are real or fake, with 1 representing real data and 0 representing fake data. In this dataset, the data from Dataset A-2 are labeled as 1, while the fake data generated from Dataset A-1 are labeled as 0.

To determine optimal performance from the defender's perspective, the number of epochs during the CTGAN model training was varied to generate fake data. Various machine learning-based classification models were then employed to evaluate and identify the optimal epoch that resulted in the lowest classification performance. Figure 3 presents the results of this preliminary experiment aimed at determining the optimal number of epochs.

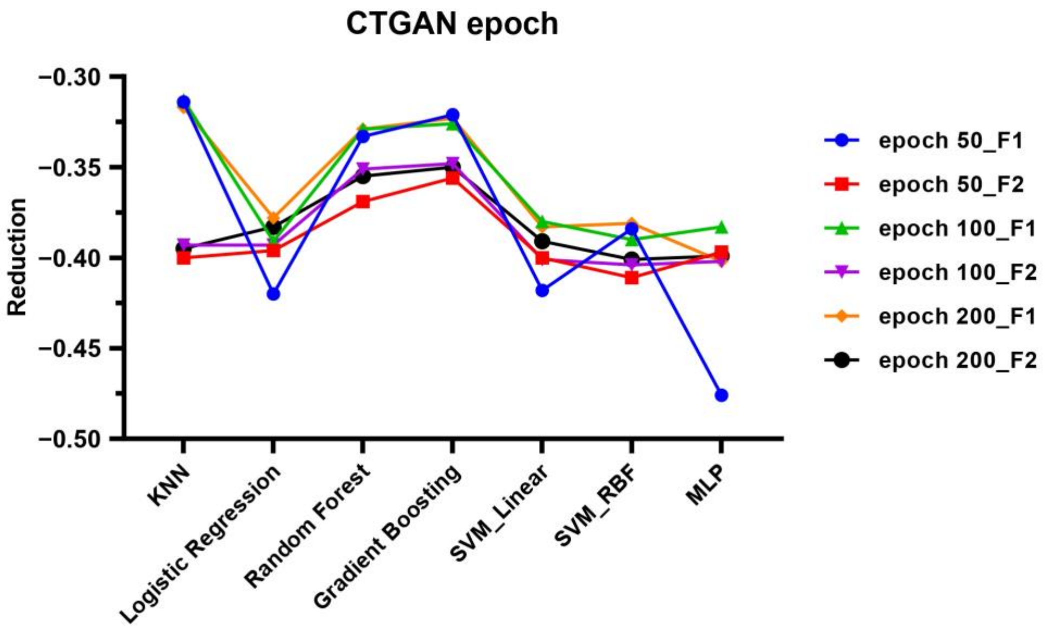

Figure 3. Performance evaluation results of generated fake mouse data based on different epochs.

The X-axis in Figure 3 represents different machine learning algorithms, while the Y-axis displays the average performance metrics (accuracy, precision, recall, etc.) of the classification models. The number of epochs for training the CTGAN model is indicated in the legends. F1 and F2 in the legends represent two feature combinations: (Elapsed Time, X Coordinate, Y Coordinate) and (Elapsed Time, X Coordinate, Y Coordinate, X Distance, and Y Distance), respectively. The lowest classification performance occurred at 50 epochs for all feature sets, and subsequent experiments were conducted with 50 epochs. Both the generator and discriminator learning rates were set to 2×10^{-4} for stability.

To maximize classification performance from the attacker's perspective, the optimal hyperparameters for each machine learning-based classification model and dataset were determined using k-fold cross-validation. First, the mixed datasets were split into training and test datasets at an 8:2 ratio. Then, k-fold cross-validation was applied to the training dataset to evaluate performance based on various parameter combinations for each classification model. The most effective parameter combinations were selected, and the final performance was evaluated using the test dataset. Detailed performance evaluation results are described in Section 4.

4. Experimental Results

In this section, we present the experimental results of the proposed method for protecting real-time mouse data. The dataset used in these experiments is the one constructed in Section 3.2. To evaluate the effectiveness of the proposed approach, we compared it with the traditional random method of generating fake data, using various feature combinations. The results of these comparisons are shown in Figures 4 and 5. Figure 4 compares the classification performance from the attacker's perspective using the feature combination of the Elapsed Time, X Coordinate, and Y Coordinate, while Figure 5 presents the performance comparison results after adding the X Distance and Y Distance features.

Table 4 represents the average values of the performance metrics for all machine learning classification algorithms shown in Figures 4 and 5. As shown in Table 4, for all datasets, the classification performance using various feature combinations significantly decreased when applying the proposed data protection method compared to the method that uses randomly generated fake data. Depending on the performance metrics, a reduction of approximately 30% to 50% was observed. Even in terms of absolute accuracy, the performance was found to be 61.9%, which is only slightly higher than the baseline accuracy of 50% for binary classification, where random selection is possible even without any meaningful classification. This result demonstrates that the proposed data protection method is highly effective.

Additionally, to verify whether the classification accuracy of fake data generated using GANs is statistically significantly lower than the classification accuracy of randomly generated fake data, a one-tailed paired *t*-test was performed based on the accuracies of all machine learning methods presented in Table 4. The confidence intervals and *p*-values for each feature set were summarized in Table 5. The results of the paired *t*-test show that for all feature sets, there was a decrease ranging from a minimum of 33.7% to a maximum of 42% within the 95% confidence interval, with *p*-values that are much smaller than 0.01. Therefore, it can be concluded that, even at the 99% confidence level, the decrease in classification accuracy is statistically significant.

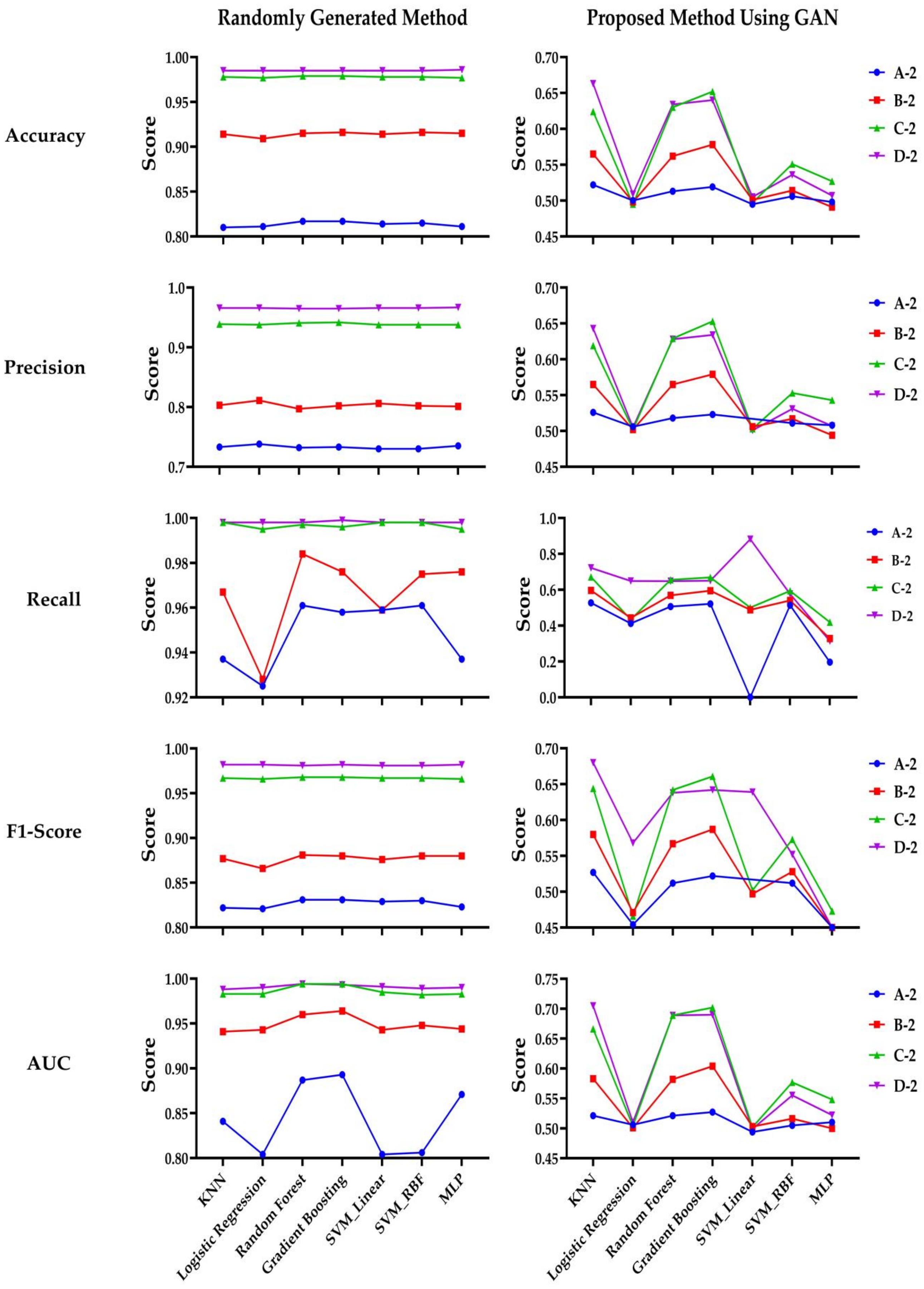

Figure 4. Comparison of classification performance based on elapsed time, X-coordinate, and Y-coordinate features.

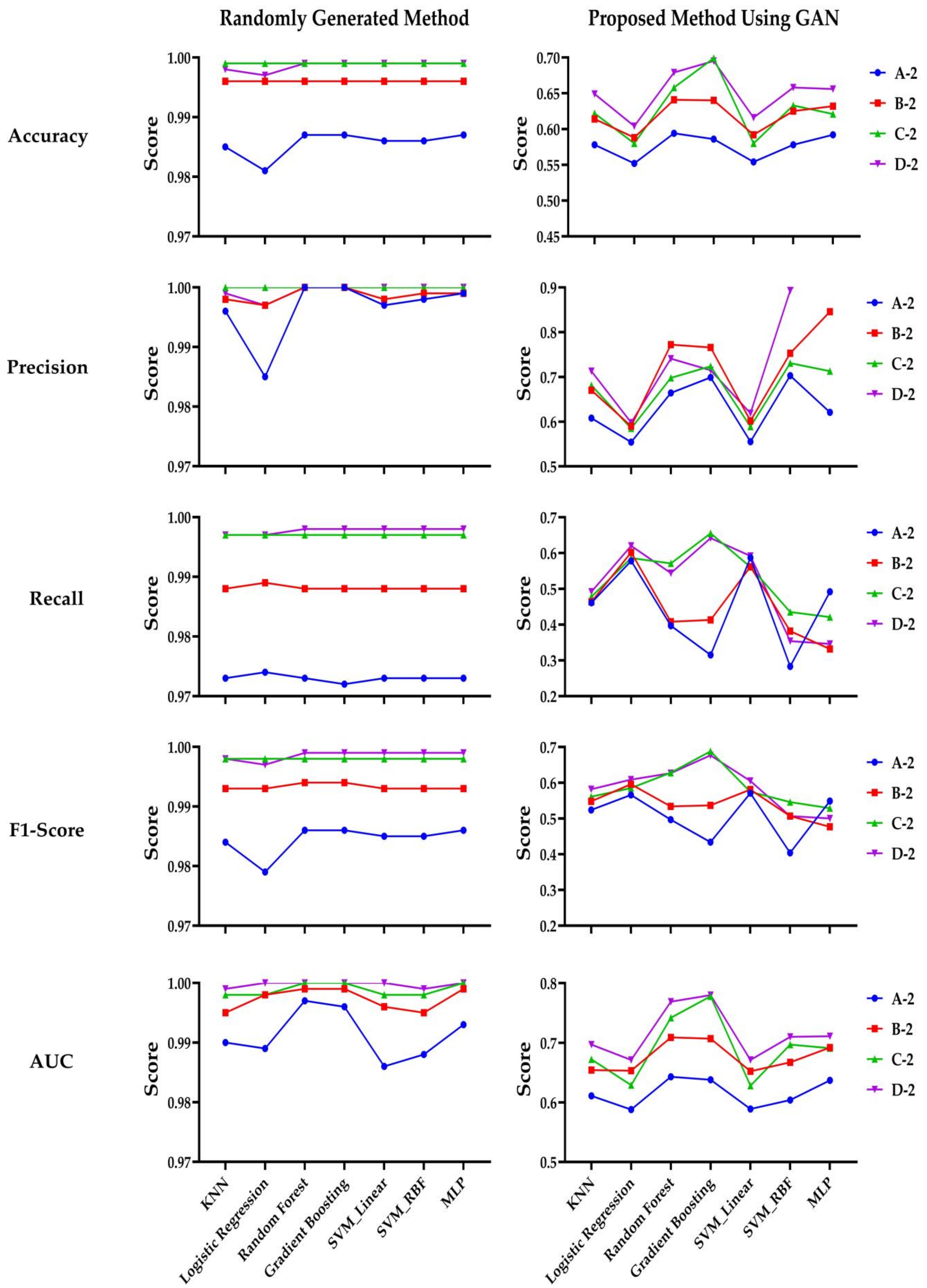

Figure 5. Comparison of classification performance based on elapsed time, X-coordinate, Y-coordinate, X-distance, and Y-distance features.

Table 4. Comparison of classification performance between the randomly generated method and the proposed method using GANs.

Feature Set	Metrics	Randomly Generated Method	Proposed Method Using GANs	Difference (+/−)
Elapsed Time, X, Y Coordinates	Accuracy	0.923	0.544	−0.379
	Precision	0.860	0.545	−0.315
	Recall	0.978	0.522	−0.456
	F1-score	0.915	0.534	−0.381
	AUC	0.942	0.562	−0.380
Elapsed Time, X, Y Coordinates X, Y Distances	Accuracy	0.995	0.619	−0.376
	Precision	0.999	0.690	−0.309
	Recall	0.989	0.485	−0.504
	F1-score	0.994	0.569	−0.425
	AUC	0.997	0.675	−0.322

Table 5. Results of the one-tailed paired *t*-test for the classification accuracies between the randomly generated method and the proposed method using GANs.

Feature Set	*t*-Statistic	Confidence Interval (95%)	*p*-Value
Elapsed Time, X, Y Coordinates	22.200	0.337–0.420	2.731×10^{-7}
Elapsed Time, X, Y Coordinates X, Y Distances	37.059	0.351–0.401	1.288×10^{-8}

5. Conclusions

In this paper, we propose a method to protect mouse movement sequence data from real-time interception attacks during image-based authentication. The proposed approach generates fake data by leveraging the user's previous mouse movement sequences and mixes this data in real time with the user's current actual mouse movement data. The effectiveness of this method is validated through various simulation experiments designed to mimic real-world scenarios from the attacker's perspective.

Specifically, the proposed mouse data protection method utilizes GANs (Generative Adversarial Networks) to generate fake mouse sequence data that closely resemble real data, making it indistinguishable from actual mouse movement data during interception attacks that rely on the GetCursorPos() function. The generated fake data also incorporate elapsed time as an additional feature. Consequently, this process alters not only the mouse position but also the movement sequence itself, thereby preventing attackers from accurately identifying the user's true mouse movement sequence.

To demonstrate the effectiveness of the proposed method, we simulated various image-based authentication processes. Using the collected mouse sequence data, fake data were generated and mixed with real data, and classification simulations were conducted using various machine learning models from the attacker's perspective. Comparative evaluations with existing methods that randomly generate fake data for protection showed that these existing methods resulted in attack success rates of up to 99%. In contrast, the proposed method reduced classification accuracy by 30% to 50%. From an absolute classification accuracy perspective, the proposed method achieved an average accuracy of 61.9% and a maximum of 65.5% across all models, which is close to the baseline random selection accuracy of 50% in binary classification. This indicates that attackers were unable to accurately classify individual mouse movement sequence data, rendering attacks focused on intercepting such data during image-based authentication ineffective.

The results of this study demonstrate that the proposed mouse data protection method effectively mitigates security threats posed by attackers attempting to intercept mouse data during image-based authentication. This approach is expected to significantly enhance the security of most image-based authentication technologies. In addition, this approach can be applied to voice-based authentication by generating fake voice sequence data based on real voice sequence data, such as waveforms and pitch. In future work, we will model the voice sequence characteristics as feature values and perform experimental analysis by collecting data in environments similar to real-world conditions to apply the proposed method to voice-based authentication technologies.

Author Contributions: Conceptualization, J.K., K.L. and H.J.; methodology, J.K., K.L. and H.J.; software, J.K. and H.J.; validation, J.K. and H.J.; data curation, J.K. and K.L.; writing—original draft preparation, J.K. and H.J.; writing—review and editing, K.L. and H.J.; supervision, H.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: https://github.com/jinwk1/Mouse-Data-Protection-.git (accessed on 14 January 2025).

Acknowledgments: This Research was supported (in part) by Glocal University Project of Mokpo National University in 2024.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Shah, S.W.; Kanhere, S.S. Recent Trends in User Authentication—A Survey. *IEEE Access* **2019**, *8*, 112505–112519. [CrossRef]
2. Walia, K.S.; Shenoy, S.; Cheng, Y. An Empirical Analysis on the Usability and Security of Passwords. In Proceedings of the 2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI), Las Vegas, NV, USA, 11–13 August 2020; pp. 1–8. [CrossRef]
3. Conklin, A.; Dietrich, G.; Walz, D. Password-based authentication: A system perspective. In Proceedings of the 37th Annual Hawaii International Conference on System Sciences, Big Island, HI, USA, 5–8 January 2004. [CrossRef]
4. Jeong, J.; Hwang, M. User Authentication System using Base Password and Member Registration Information. *J. Korea Inst. Inf. Commun. Eng.* **2016**, *20*, 2289–2296. [CrossRef]
5. Sidhu, S.; Mohd, B.J.; Hayajneh, T. Hardware Security in IoT Devices with Emphasis on Hardware Trojans. *J. Sens. Actuator Netw.* **2019**, *8*, 42. [CrossRef]
6. Dhamija, R.; Perrig, A. Déjà vu. A user study Using Images for Authentication. In Proceedings of the 9th Usenix Security Symposium, Denver, CO, USA, 14–17 August 2000; pp. 44–58.
7. Newman, R.E.; Harsh, P.; Jayaraman, P. Security analysis of and proposal for image-based authentication. In Proceedings of the 39th Annual 2005 International Carnahan Conference on Security Technology, Las Palmas, Spain, 11–14 October 2005; pp. 141–144. [CrossRef]
8. MSDN. Available online: https://learn.microsoft.com/ko-kr/windows/win32/api/winuser/nf-winuser-getcursorpos (accessed on 14 December 2024).
9. MSDN. Available online: https://learn.microsoft.com/ko-kr/windows/win32/api/winuser/nf-winuser-setcursorpos (accessed on 14 December 2024).
10. Lee, K.; Esposito, C.; Lee, S.-Y. Vulnerability Analysis Challenges of the Mouse Data Based on Machine Learning for Image-Based User Authentication. *IEEE Access* **2019**, *7*, 177241–177253. [CrossRef]
11. Lee, K.; Lee, S.-Y. Improved Practical Vulnerability Analysis of Mouse Data According to Offensive Security based on Machine Learning in Image-Based User Authentication. *Entropy* **2020**, *22*, 355. [CrossRef] [PubMed]
12. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. In Proceedings of the Advances in Neural Information Processing Systems 27 (NIPS 2014), Montreal, QC, Canada, 8–13 December 2014; pp. 2672–2680. [CrossRef]
13. Park, S.-W.; Ko, J.-S.; Huh, J.-H.; Kim, J.-C. Review on Generative Adversarial Networks: Focusing on Computer Vision and Its Applications. *Electronics* **2021**, *10*, 1216. [CrossRef]

14. Shahriar, M.H.; Haque, N.I.; Rahman, M.A.; Alonso, M. G-ids: Generative adversarial networks assisted intrusion detection system. In Proceedings of the 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), Madrid, Spain, 13–17 July 2020; pp. 376–385. [CrossRef]
15. Zhao, X.; Fok, K.W.; Thing, V.L. Enhancing Network Intrusion Detection Performance using Generative Adversarial Networks. *Comput. Secur.* **2024**, *145*, 104005. [CrossRef]
16. Agrawal, G.; Kaur, A.; Myneni, S. A review of generative models in generating synthetic attack data for cybersecurity. *Electronics* **2024**, *13*, 322. [CrossRef]
17. Singh, M.; Baranwal, N.; Singh, K.N.; Singh, A.K. Using GAN-Based Encryption to Secure Digital Images With Reconstruction Through Customized Super Resolution Network. *IEEE Trans. Consum. Electron.* **2024**, *70*, 3977–3984. [CrossRef]
18. Singh, P.; Dutta, S.; Pranav, P. Optimizing GANs for Cryptography: The Role and Impact of Activation Functions in Neural Layers Assessing the Cryptographic Strength. *Appl. Sci.* **2024**, *14*, 2379. [CrossRef]
19. MSDN. Available online: https://learn.microsoft.com/ko-kr/windows/win32/apiindex/windows-api-list (accessed on 14 December 2024).
20. Xu, L.; Skoularidou, M.; Cuesta-Infante, A.; Veeramachaneni, K. Modeling tabular data using conditional GAN. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; pp. 7335–7345. Available online: https://proceedings.neurips.cc/paper/2019/hash/254ed7d2de3b23ab10936522dd547b78-Abstract.html (accessed on 14 December 2024).
21. CTGAN GitHub. Available online: https://github.com/sdv-dev/CTGAN/blob/main/README.md (accessed on 14 December 2024).
22. Wang, Z.; She, Q.; Ward, T.E. Generative adversarial networks in computer vision: A survey and taxonomy. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–38. [CrossRef]
23. Jabbar, A.; Li, X.; Omar, B. A survey on generative adversarial networks: Variants, applications, and training. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–49. [CrossRef]