User Authentication on Remote Connections with Siamese Networks Using Keyboard Usage Behavior and Corresponding Noise Performances

Mehmet Fide[a] and Emin Anarim[b]
Department of Electrical & Electronics Engineering, Boğaziçi University, İstanbul, Turkey
{*mehmet.fide, anarim*}*@boun.edu.tr*

Keywords: Information Security, Networks Security, Keyboard Keystroke Dynamics, User Identification, Anomaly Detection, Siamese Networks, GRU, LSTM, BI-GRU, BI-LSTM, Network Latency, Jitters.

Abstract: The investigation into user authentication via analysis of keyboard usage behaviors has garnered considerable attention in the literature, resulting in numerous published works on the subject. Continually, novel metric-based or artificial intelligence-based authenticators are introduced, with their respective advantages being documented. However, none of these studies have evaluated the performance of these authenticators when the system is accessed from a remote computer or through a cloud-based environment. This study aims to address this gap by observing users' keyboard usage behaviors through a remote terminal over a network. Utilizing the Carnegie Mellon University (CMU) dataset, features were transmitted over a local network using various protocols to assess the impact of delay variations on the performance of three newly proposed classifiers in addition to the 17 existing classifiers in the literature. Furthermore, to bolster the practical findings, a mathematical model incorporating network delays as input was proposed, and the performances of the studied classifiers were compared at different signal-to-noise ratios.

1 INTRODUCTION

With the increasing prevalence of digital assets, protecting services from unauthorized access has become crucial. Authentication processes today aim to address modern security requirements through various methods such as password entry, asymmetric certificates, hardware dongles, or two-factor authentication systems. However, since these tools can be duplicated or compromised by unauthorized individuals, achieving the desired level of security remains challenging. Biometric identification enables multi-layered authentication systems by tracking users during service access or active session.

Biometric identification technologies can be categorized into two groups. The first category seeks to identify users based on physiological characteristics such as facial features, fingerprints, and retinas. The second category focuses on behavioral characteristics, such as signatures, keyboard usage, and mouse movements. Keystroke biometrics, a field that aims to identify users based on their keyboard behaviors, has garnered academic interest since the late 1970s (Peacock et al., 2004). Upon examining studies related to keyboard behavior, research efforts can be observed in two subcategories. The first is referred to as the "Free Text" group, which is employed in scenarios where the content and length of the written text are unspecified. The second is the "Fixed Text" or also known as "Short Text" category, which focuses on shorter or fixed-length text entries. These classification methods are predominantly used in scenarios such as password verification or the validation of PowerShell or Bash command usage. The primary aim of the study is to validate the performance of Fixed Text classifiers when operated through a network service, using parameters that allow for comparison with other classifiers, thereby addressing a gap in the existing literature. Therefore we utilized the CMU dataset as input to evaluate the performance of the proposed classifiers.

The CMU dataset was created by recording the keyboard usage behaviors of users on a local computer. With the rise of remote work, especially during the 2020 pandemic, the management of dedicated and virtual servers through remote access has become more common. Consequently, it has become increasingly important to examine user behaviors observed from remote terminals such as SSH (Se-

[a] https://orcid.org/0009-0004-7046-6188
[b] https://orcid.org/0000-0002-3305-7674

Fide, M. and Anarim, E.
User Authentication on Remote Connections with Siamese Networks Using Keyboard Usage Behavior and Corresponding Noise Performances.
DOI: 10.5220/0013252500003899
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 11th International Conference on Information Systems Security and Privacy (ICISSP 2025) - Volume 2*, pages 585-594
ISBN: 978-989-758-735-1; ISSN: 2184-4356
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

cure Shell), Telnet (Teletype Network), and RDP (Remote Desktop Protocol) in addition to local keyboard behaviors. In modern systems, key exchanges between clients and servers are conducted using algorithms such as Diffie–Hellman (Lange and Winterhof, 2003), Curve25519 (Bernstein, 2006) or similar methods (Ylonen, 2019) ensuring that keys are not transmitted in plain text to mitigate the risk of network sniffing. However, short text commands that need to be executed by the user afterward such as Linux Bash commands like ”ls”, ”rmdir”, ”cp”, ”find”, ”sudo” or passwords required within an active session or critical Windows PowerShell commands like ”Get-ChildItem”, ”Remove-Item”, and ”Copy-Item” are still considered as short-text classifications.

In the second part of the study, we emulated the keystroke timings from the CMU dataset as if the same users were physically present at the computer. We then transmitted these keystrokes to a server on a local network using various protocols, including UDP (User Datagram Protocol), TCP (Transmission Control Protocol) with and without enabling Nagle Algorithm, TCP+AES128 (Advanced Encryption Standard), and TCP+AES256. We examined the changes in classifier performance when using the newly observed timing information. Additionally, we modeled network latency and jitter as noise at varying intensities to demonstrate the impact on classifier performance.

This study for the first time in the literature, demonstrated how the performance of classifiers would change if CMU's local users conducted the same experiment over a network. Additionally, it introduced a parameter that had not been previously examined in the literature, such as the noise immunity of the classifiers.

2 RELATED WORK

K. S. Killourhy (Killourhy and Maxion, 2009) attempted to detect users during password entry and compared the performance of various metric-based classifiers within a specific framework. The CMU Keystroke Dynamics Benchmark Dataset used in this study has become a foundational resource for subsequent research in this area. In a 2016 study, Morales et al. (Morales et al., 2016b) summarized the ongoing competition among researchers in this field, noting that the best-known result, with an Equal Error Rate (EER) of 5.32%, was achieved by the U.S. Army Research Laboratory (Morales et al., 2016a). In 2019, M. Yıldırım (Yıldırım and Anarım, 2019) showed that users could be distinguished based on mouse behavior instead of keyboard behavior. Similarly, E. Davarcı (Davarci and Anarim, 2022) demonstrated in his work in 2022 that using dual-class classifiers, instead of single-class classifiers, could further reduce error rates.

In the 2020 study titled TypeNet-Scaling up Keystroke Biometrics (Acien et al., 2020), a free-text classification was performed using the Siamese Network architecture and the LSTM (Long Short Term Memory) module, achieving a 4.8% equal error rate for sequences of 50 keystrokes. In the study, five features were used for each keystroke. It has been observed that in free-text studies, as the sequence length increases, the system's performance improves, whereas it decreases significantly with shorter sequences which are the cases for the short-text classifications. In a 2022 study (Acien et al., 2022), a similar topology was trained under different loss functions, yielding error rates of 10.7% with the contrastive loss and 8.6% with the triplet-loss function for sequences of 30 keystrokes. Subsequently, in 2023, a study (Medvedev et al., 2023) focused on enhancing the efficiency of the Siamese structure in short-text classification by transforming keystroke timing information into 2D images using the Recurrence Plots method. This approach allowed for the creation of longer sequences, reducing the system's error rate to 7.8%. Later, in another study conducted in 2024 (Fide and Anarım, 2024), the Siamese network structure was optimized with a GRU (Gated Recurrent Units) module. Instead of averaging of each short sequential evaluation, feeding the network with a long sequence includes the all averaged sequence from the beginning, successfully reduced the error rate to 7.0% using contrastive loss function. In this current study, we enhanced the existing Siamese network GRU model with a more efficient outlier filtering method and successfully reduced the equal error rate to 6.5% using 31 features available in the CMU's dataset. Additionally, we tested the same architecture under Bi-GRU (Bidirectional Gated Recurrent Unit), LSTM, and Bi-LSTM (Bidirectional Long Short Term Memory) models and reported their performances. Furthermore, we listed the remote performance metrics of these four proposed new networks and examined their noise immunity capabilities.

3 METHOD

3.1 Dataset

Most studies on user keyboard behaviors for fixed-length short texts utilize the CMU Keystroke Dynamics Benchmark Dataset (Killourhy and Maxion, 2009). In this dataset, 51 different users were asked to type the string ".tie5Roanl" 50 times over 8 days at various times. Transition times between keystrokes and the duration each key was pressed were recorded to create the features. Precise timestamps were recorded using a high-precision external timer connected to a laptop running Windows XP. The system's accuracy was monitored with a high-precision signal generator and measured as $\pm 200\mu$s. Each keystroke has three fundamental features: hold time of a key ($H.key1$), duration between down events of two consecutive keys ($DD.key1.key2$), and duration between up to down events of two consecutive keys ($UD.key1.key2$), resulting in a total of 31 features for the 11 keystrokes in each row.

For transmissions over the network, the keystroke hold time ($H.key1$) and the time between releasing one key and pressing the next ($UD.key1.key2$) were not used. Only the delays between pressing consecutive keys ($DD.key1.key2$) were utilized. Consequently, the feature inputs for the classifiers were reduced to 10 features.

Figure 1 illustrates the completion times for each tasks performed by Subject-03 as an example and the variations in these times over the observed period. It is observed that over time, users gained proficiency with the experiment and, completed the tasks in shorter durations on average.

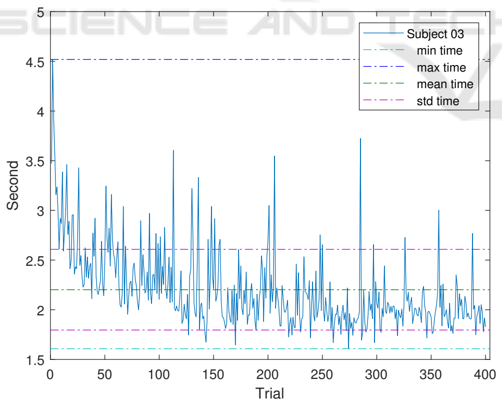


Figure 1: Subject 03 - Total Timings over trials.

3.2 Proposed Siamese Network Architecture

As explained in the 3.1, there are 10 keystroke timestamps transferred over the network in the dataset. Therefore our classification procedures were performed based on this set of 10 features. Before any processing, the values in the dataset were normalized by dividing each value by 255. Subsequently using the Recurrent Plot (Medvedev et al., 2023) method, the timing values were converted into $10x10$ images, and these images were then flattened to form a $1x100$ input vector. Instead of individually comparing this vector with the 128 rows belonging to genuine users in the training set and averaging the results, the vector was reformatted to $128x100$ and input into the system in a single step. This approach eliminated the need for averaging, and an improvement in system performance was observed (Fide and Anarım, 2024).

Each Bi-GRU/Bi-LSTM structure fundamentally consists of 32 cells, but due to its bidirectional nature, this number increases to 64 cells. Each cell has a *tanh* activation function and a recurrent dropout rate of 0.2. To further prevent over-fitting, an additional dropout layer with a rate of 0.2 has been included in the system. The final dense layer has been added to stabilize the validation loss function. The Siamese network constructed in this configuration contains a total of 46,952 (Bi-GRU) and 61,288 (Bi-LSTM) trainable parameters, which are shared between the left and right branches of the network.

3.3 Training Phase

The whole CMU dataset consists of 51 users, each with 400 samples, and each sample contains 31 timestamp features related to keystroke durations but reduced to 10 after transmitting them thought the local network. When training the Siamese biGRU/biLSTM network architecture for each user, 200 samples out of the 400 were used for training, and the remaining 200 for testing. In all studies conducted within the scope of this article, the training and test sets were kept strictly separate.

10% of the training data were set aside for validation and for early stopping during the training phase. Additionally, in the normal training set, values that deviated from the median by more than 5.5 times the mean absolute deviation were considered outliers and were excluded. To be able to train the proposed network structure, samples representing anomalies, as well as positive labels specific to the user were required. This requirement was met using the training sets of other users within the CMU dataset. Negative labels were taken from first five samples of 50 other users to ensure a balance between positive and negative labels during training. Therefore, a total of 450 records in the training set are available for each user, comprising 200 positive and 250 negative samples.

As the loss function, the contrastive loss function (Hadsell et al., 2006) was used. Given input vector pairs x_i and x_j containing the user's keystroke timestamps, with network output $f(.)$ as shown in Figure

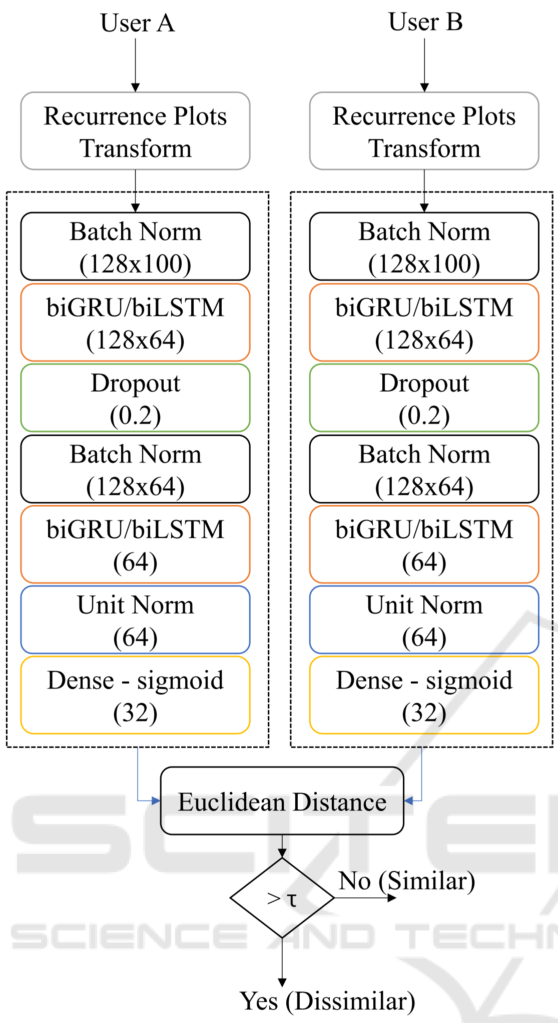


Figure 2: Proposed Siamese Network with Bi-GRU/Bi-LSTM models.

2, and L_{ij} defined as 0 for pairs x_i and x_j from the same class and 1 for pairs from different classes, the contrastive loss function $\mathcal{L}_{\text{CF}}$ is defined as shown in (1):

$$\begin{aligned} \mathcal{L}_{\text{CF}} = & (1 - L_{ij}) \frac{\left\| f(x_i) - f(x_j) \right\|_2^2}{2} \\ & + L_{ij} \frac{\max^2 \left\{ 0, \varepsilon - \left\| f(x_i) - f(x_j) \right\|_2 \right\}}{2} \end{aligned} \tag{1}$$

During training, the margin value ε was set to 0.1. Optimal results were achieved using the Adam optimizer with a learning rate of 0.0001, ε=1e-8, a maximum of 200 epochs, and a batch size of 32. The processes were performed on an AMD Ryzen Threadripper 3990X 64-Core Processor, with 256GB RAM, Linux Kernel v6.8, using Python 3.12.3 and TensorFlow v2.17.0. For reproducibility, the seed of the Pseudo Random Generators were set to a fixed value of 7. All Python codes developed for the study are provided on GitHub (Fide, 2024).

3.4 Real-Time Keystroke Emulator

The purpose of the real-time keystroke emulator is to accurately replicate the keystroke events of users from the CMU dataset in a network environment. This allows us to obtain the keystroke timing information of CMU dataset users as if they were connecting to a remote terminal rather than typing on a local computer. The structure of the emulator and server is shown in Figure 3.

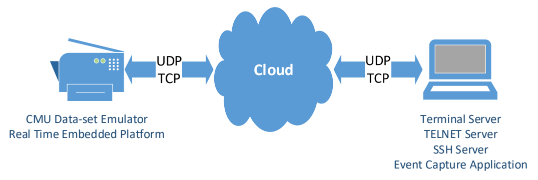


Figure 3: Proposed real-time keystroke emulator.

The Real-Time Keystroke Emulator was designed because non-real-time operating systems such as Windows (Microsoft, 2024) or Linux cannot generate events with a sufficient accuracy to match the timing information in CMU's dataset while they can easily capture the events with accurate timestamps with the help of existing hardware supports such as IEEE 1588 Time Stamping (IEEE, 2008) provided in the Ethernet transceivers. The emulator is equipped with a MIPS (Microprocessor without Interlocked Pipelined Stages) based micro controller runs at 200MHz, a 10MHz temperature-controlled oscillator, a Real-Time Clock, and an Ethernet port with a bandwidth of 100Mbps. Operating without any operating system, the emulator runs a TCP stack directly on bare metal. It has been observed that the emulator can generate keystroke events with 10μs precision based on CMU's dataset.

The emulator listens on specific TCP and UDP ports as a server. Upon receiving timing information in the format shown in Figure 4, it automatically replicates the specified events over the network at the indicated times.

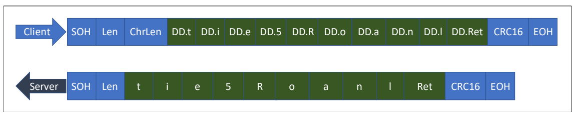
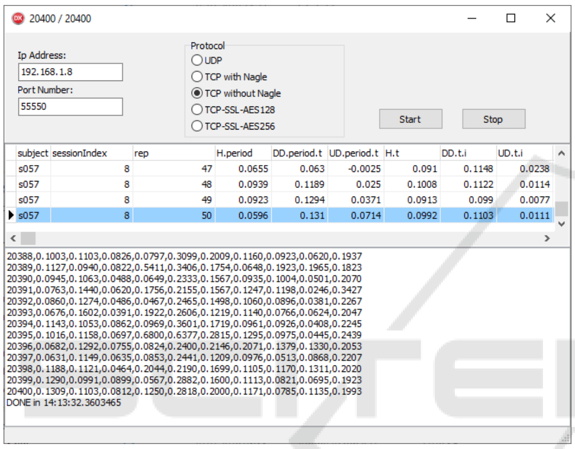


Figure 4: The Emulator communication protocol, the client asks and the server replies.

3.5 The Windows Application

CMU's dataset contains 20400 rows of records from 51 users, encompassing a total of 204000 keystrokes. As illustrated in Figure 5, the developed Windows application (Embarcadero, 2024) emulated the entire dataset under various scenarios including UDP, TCP+Nagle, TCP-Nagle, TCP+AES128, and TCP+AES256. The network packet information was captured using Wireshark (Orebaugh et al., 2007), and the timing information was analyzed. Each scenario took approximately 14 hours to complete. Embedded C and Delphi codes can be accessed on GitHub (Fide, 2024) publicly.



Ip Address: 192.168.1.8
Port Number: 55550

Protocol
UDP
TCP with Nagle
[✓ TCP without Nagle]
TCP-SSL-AES128
TCP-SSL-AES256

	subject	sessionIndex	rep	H.period	DD.period.t	UD.period.t	H.t	DD.t.i	UD.t.i
	s057	8	47	0.0655	0.063	-0.0025	0.091	0.1148	0.0238
	s057	8	48	0.0939	0.1189	0.025	0.1008	0.1122	0.0114
	s057	8	49	0.0923	0.1294	0.0371	0.0913	0.099	0.0077
▶	s057	8	50	0.0596	0.131	0.0714	0.0992	0.1103	0.0111

```
20388,0.1003,0.1103,0.0826,0.0797,0.3099,0.2009,0.1160,0.0923,0.0620,0.1937
20389,0.1127,0.0940,0.0822,0.5411,0.3406,0.1754,0.0648,0.1923,0.1965,0.1823
20390,0.0945,0.1063,0.0488,0.0649,0.2333,0.1567,0.0935,0.1004,0.0501,0.2070
20391,0.0763,0.1440,0.0620,0.1756,0.2155,0.1567,0.1247,0.1198,0.0246,0.3427
20392,0.0860,0.1274,0.0486,0.0467,0.2465,0.1498,0.1060,0.0896,0.0381,0.2267
20393,0.0676,0.1602,0.0391,0.1922,0.2606,0.1219,0.1140,0.0766,0.0624,0.2047
20394,0.1143,0.1053,0.0862,0.0969,0.3601,0.1719,0.0961,0.0926,0.0408,0.2245
20395,0.1016,0.1158,0.0697,0.6800,0.6377,0.2815,0.1295,0.0975,0.0445,0.2439
20396,0.0682,0.1292,0.0755,0.0824,0.2400,0.2146,0.2071,0.1379,0.1330,0.2053
20397,0.0631,0.1149,0.0635,0.0853,0.2441,0.1209,0.0976,0.0513,0.0868,0.2207
20398,0.1188,0.1121,0.0464,0.2044,0.2190,0.1699,0.1105,0.1170,0.1311,0.2020
20399,0.1290,0.0991,0.0899,0.0567,0.2882,0.1600,0.1113,0.0821,0.0695,0.1923
20400,0.1309,0.1103,0.0812,0.1250,0.2818,0.2000,0.1171,0.0785,0.1135,0.1993
DONE in 14:13:32.3603465
```

Figure 5: The Windows application that transmits CMU data to the emulator.

3.6 Performance Criteria

To compare the classifier performances in the biometric studies, usually the Equal Error Rate (EER) and the Zero Miss Rate (ZMR) are used. To calculate the EER, the sensor threshold level is selected such that false rejection and false acceptance rates are equal, as shown in Figure 6. This method was employed by Kang in 2007 (Kang et al., 2007a). Because the EER is an error rate, a lower values indicate better sensor performances. In this study, a total of $51 * 6 * 20 = 6120$ ROC (Receiver Operating Characteristic) curves were generated for each of the six different training sets as CMU local, UDP, TCP+Nagle, TCP-Nagle, TCP+AES128 and TCP+AES256 and for 20 different classifiers. And the EER values were calculated as the mean of 51 users. While calculating these values for each individual classifiers, the train and test method used by Killourhy (Killourhy and Maxion, 2009), which later became a reference for researchers in this field, was strictly followed. The first 200 out of 400 data points for each user were utilized to train the classifiers, while the remaining 200 were used to test genuine users. For the imposter test, a total of 250 data points were used, comprising the first 5 data points from each of the 50 other users.

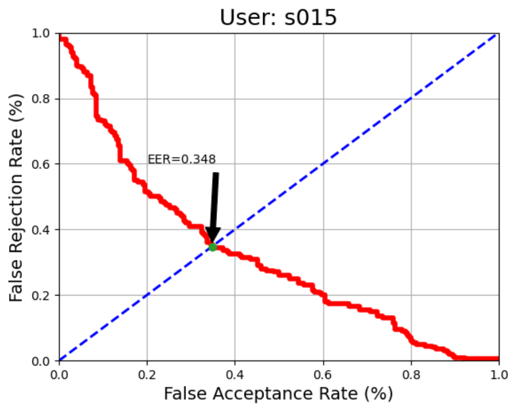


Figure 6: A ROC curve and EER value belong to user $s015$.

3.7 The Classifier Performances

In the literature, classifiers utilizing the CMU dataset report their performances using all 31 features in the dataset (Killourhy and Maxion, 2009). Since not all of these physical features are available during network transmissions, Table 1 lists the EER of existing classifiers by replicating the conditions where only 10 features from the original CMU dataset are used. This allows for a comparison between the classifier performances obtained using the original data and those obtained after introducing network and protocol delays. Additionally, Table 1 presents the EER values for scenarios where keystrokes were transmitted using UDP. UDP is a connection-less protocol commonly used in applications where low network latency is crucial, but packet delivery and content consistency are not guaranteed. It is observed that when network latency is significantly lower than the user's keystroke timings, classifiers trained using UDP transmission dataset and local CMU dataset perform nearly identically in metric-based classifiers and similarly in AI-based classifiers.

Table 2 lists the changes in EER values when using TCP. The impact of the Nagle algorithm (Peterson and Davie, 1996), which degrades classifier performance by approximately 7%, is clearly shown. The Nagle algorithm improves the efficiency of TCP protocol headers by delaying the transmission of small data packets, leading to additional delays in the transmission of small packets. Unlike UDP, TCP is a connection-oriented protocol that ensures the correct order and delivery of data. However, this comes at the cost of increased network latency, which signifi-

Table 1: These results were obtained by using only the DD.key1.key2 features from the original CMU dataset and transmitting the data over the network using UDP.

UDP Comparisons using only DD.key1.key2 features vs Original CMU			
No	**Classifier**	**CMU - EER**	**UDP - EER**
1	Nearest Neighbor with PSR (Zhong et al., 2012)	0.115	0.115
2	Siamese GRU (Fide and Anarım, 2024)	0.119	0.122
3	Nearest Neighbor with Mahal. (Cho et al., 2000)	0.121	0.121
4	Siamese Bi-GRU*	0.125	0.126
5	Siamese Bi-LSTM*	0.125	0.127
6	Siamese LSTM*	0.131	0.132
7	One-Class SVM (Yu and Cho, 2003)	0.138	0.137
8	z-score_optim_nu(1.292) (Haider et al., 2000)	0.145	0.145
9	Scaled Manhattan (Araujo et al., 2005)	0.151	0.151
10	Bayesian (Rennie et al., 2003)	0.153	0.153
11	Mahalanobis (Duda et al., 2001)	0.153	0.153
12	Filtered Manhattan (Joyce and Gupta, 1990)	0.154	0.154
13	k-means (Kang et al., 2007b)	0.156	0.156
14	Normalized Mahalanobis (Bleha et al., 1990)	0.168	0.169
15	Manhattan (Duda et al., 2001)	0.172	0.172
16	Euclidean (Duda et al., 2001)	0.187	0.187
17	z-score (Haider et al., 2000)	0.189	0.190
18	Multilayer Perceptron NN (Cho et al., 2000)	0.201	0.218
19	Standard Neural Network (Haider et al., 2000)	0.201	0.202
20	Normalized Euclidean (Bleha et al., 1990)	0.219	0.219

Table 2: The reflections of the EER values in scenarios where the Nagle algorithm is enabled and disabled during TCP usage are listed.

TCP+Nagle (Nagle enabled) vs TCP-Nagle (Nagle disabled)			
No	**Classifier**	**TCP+Nagle EER**	**TCP-Nagle EER**
1	Nearest Neighbor with PSR (Zhong et al., 2012)	0.189	0.116
2	Nearest Neighbor with Mahal. (Cho et al., 2000)	0.189	0.124
3	Siamese Bi-GRU*	0.201	0.125
4	Siamese Bi-LSTM*	0.221	0.130
5	Siamese GRU (Fide and Anarım, 2024)	0.224	0.123
6	Siamese LSTM*	0.232	0.131
7	One-Class SVM (Yu and Cho, 2003)	0.267	0.138
8	z-score_optim_nu(1.195) (Haider et al., 2000)	0.243	0.145
9	Scaled Manhattan (Araujo et al., 2005)	0.234	0.152
10	Bayesian (Rennie et al., 2003)	0.189	0.153
11	Mahalanobis (Duda et al., 2001)	0.189	0.153
12	Filtered Manhattan (Joyce and Gupta, 1990)	0.231	0.154
13	k-means (Kang et al., 2007b)	0.225	0.157
14	Normalized Mahalanobis (Bleha et al., 1990)	0.247	0.169
15	Manhattan (Duda et al., 2001)	0.239	0.172
16	Euclidean (Duda et al., 2001)	0.325	0.187
17	z-score (Haider et al., 2000)	0.259	0.191
18	Multilayer Perceptron NN (Cho et al., 2000)	0.286	0.218
19	Standard Neural Network (Haider et al., 2000)	0.279	0.211
20	Normalized Euclidean (Bleha et al., 1990)	0.332	0.219

Table 3: The reflections of EER values for scenarios where AES128 and AES256 encryption algorithms are used during TCP transmissions are listed.

TCP+AES128 vs TCP+AES256			
No	**Classifier**	**TCP+AES128 EER**	**TCP+AES256 EER**
1	Nearest Neighbor with PSR (Zhong et al., 2012)	0.116	0.116
2	Nearest Neighbor with Mahal. (Cho et al., 2000)	0.122	0.123
3	Siamese Bi-GRU*	0.125	0.124
4	Siamese GRU (Fide and Anarım, 2024)	0.126	0.124
5	Siamese Bi-LSTM*	0.127	0.126
6	Siamese LSTM*	0.132	0.132
7	One-Class SVM (Yu and Cho, 2003)	0.138	0.139
8	*z*-score_optim_nu(1.293) (Haider et al., 2000)	0.146	0.146
9	Scaled Manhattan (Araujo et al., 2005)	0.152	0.152
10	Bayesian (Rennie et al., 2003)	0.152	0.153
11	Mahalanobis (Duda et al., 2001)	0.152	0.153
12	Filtered Manhattan (Joyce and Gupta, 1990)	0.154	0.154
13	*k*-means (Kang et al., 2007b)	0.156	0.157
14	Normalized Mahalanobis (Bleha et al., 1990)	0.169	0.169
15	Manhattan (Duda et al., 2001)	0.172	0.171
16	Euclidean (Duda et al., 2001)	0.186	0.186
17	*z*-score (Haider et al., 2000)	0.190	0.190
18	Multilayer Perceptron NN (Cho et al., 2000)	0.210	0.216
19	Standard Neural Network (Haider et al., 2000)	0.223	0.209
20	Normalized Euclidean (Bleha et al., 1990)	0.218	0.219

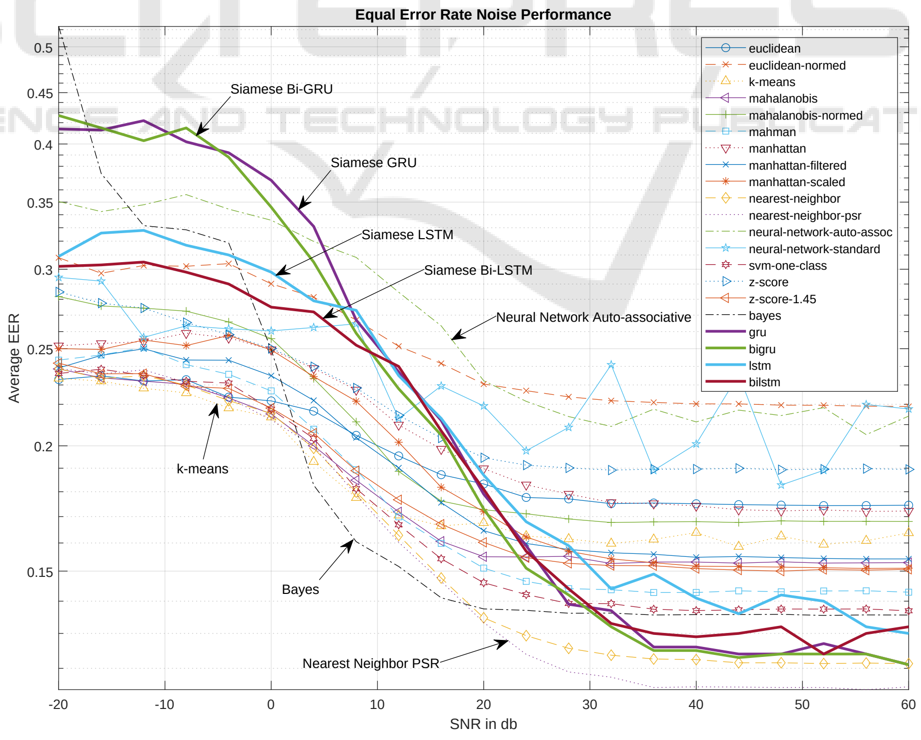


Figure 7: The effects of network delays and jitters on the performance of classifiers are illustrated.

cantly affects the physical characteristics of a user's keystroke dynamics.

Table 3 presents the EER values obtained when a 128-bit and 256-bit AES encryption layer is added to the data transmitted over the TCP protocol. Today, most applications requiring secure connections (HTTPS, SSL, SSH, etc.) communicate using this structure. It is observed that communications conducted in this manner are not affected by the Nagle algorithm, which is enabled by default in TCP, and the EER values are comparable to the EER values obtained locally from the CMU dataset.

4 NOISE IMMUNITY OF CLASSIFIERS

In the practical study, a completely isolated network structure was used, minimizing the effects of traffic congestion on the results. To test this effect at different levels, network delays and jitters affecting keystroke attributes were modeled as compounded semi-normal distributed noise. Given x as the feature vector, n as the size of each feature vector, η_i as the delay noise added to each keystroke timings, $E[\cdot]$ as the expectation operator, $\sigma_\eta^2 = E[xx^T]/10^{\frac{SNR_{dB}}{10}}$, $\beta = 1 - 2/\pi$, and SNR as the desired signal-to-noise ratio in dB, the resulting feature vector y was obtained as shown in Equation (3) and T total time to a classifier input is given in Equation (4).

$$\eta_i \sim \left| \aleph(0, \frac{\sigma_\eta^2}{\beta}) \right| \tag{2}$$

$$y_i = x_i + \eta_i \quad i = 1..n \tag{3}$$

$$T = \sum_{i=1}^{n} x_i + \eta_i \tag{4}$$

The EER breakdowns of classifiers at signal-to-noise ratios ranging from -20 dB to +60 dB, resulting from the modeling, are shown in Figure 7. Accordingly, in environments with low network delay and jitters, the Nearest Neighbor PSR method (Zhong et al., 2012) shows the best performance. In contrast, in environments with high delay and jitters, the classical Euclidean method (Duda et al., 2001), followed closely by the One-Class SVM method (Yu and Cho, 2003), provides the best results. Generally, at signal-to-noise ratios of +20 dB or lower, classifier EER values deteriorate, with the maximum performance degradation occurring around +10 dB.

While GRU and Bi-GRU structures exhibit similar EER values in low-noise environments, Bi-GRU outperforms GRU within the 0 to 25 dB SNR range. This suggests that Bi-GRU has greater noise immunity than GRU. Interestingly, LSTM and Bi-LSTM models do not perform as much as GRU and Bi-GRU in noise-free environments. However, they demonstrate significantly better performance than GRU-based models, especially below +10 dB SNR.

5 CONCLUSIONS

In this study, we proposed three new Siamese network authenticators and investigated the keyboard usage behaviors of users connecting to a remote computer over a network, taking into account network delays and jitters. To facilitate this examination, a keystroke emulator was designed and developed. The performance of our newly proposed classifiers as well as others frequently evaluated using the CMU dataset in local connections in the literature, was observed for the first time in a network environment. Their performance changes were systematically recorded in tables, under the influence of UDP, TCP, encrypted, and unencrypted channels. Additionally, network delays and jitters were modeled as compounded semi-normal distributed noise to identify the classifiers with the highest and lowest noise immunity. The study revealed that classifiers demonstrating the best performance on the CMU dataset in local tests do not maintain the same performance levels in network environments, particularly under conditions involving Nagle's algorithm or network congestion. Conversely, some classifiers that perform poorly on local datasets exhibited improved performance in network environments. We also identified that Bi-LSTM and Bi-GRU models have better noise immunity and perform better in a noisy environments than LSTM and GRU models.

Future research can further investigate the noise immunities and average EER changes of current artificial intelligence algorithms, in addition to metric-based classifiers. This study assumed that network characteristics were stationary during the training and testing phases; however, realistic scenarios necessitate examining performance under non-stationary jitters and delays.

Understanding why certain classifiers perform better in noisy environments could lead to the development of methods to enhance noise immunity. Additionally, examining classifier performance in the presence of various security layers like Virtual Private Networks (VPN) (Ostroukh et al., 2024) or physical network environments, such as WiFi, 3G, 4G, and 5G, can provide a comprehensive understanding of their effectiveness under different conditions.

REFERENCES

Acien, A., Morales, A., Monaco, J. V., Vera-Rodriguez, R., and Fierrez, J. (2022). Typenet: Deep learning keystroke biometrics. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 4(1):57–70.

Acien, A., Morales, A., Vera-Rodriguez, R., Fierrez, J., and Monaco, J. V. (2020). Typenet: Scaling up keystroke biometrics. In *2020 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–7.

Araujo, L., Sucupira, L., Lizarraga, M., Ling, L., and Yabu-Uti, J. (2005). User authentication through typing biometrics features. *IEEE Transactions on Signal Processing*, 53(2):851–855.

Bernstein, D. J. (2006). Curve25519: New diffie-hellman speed records. In Yung, M., Dodis, Y., Kiayias, A., and Malkin, T., editors, *Public Key Cryptography - PKC 2006*, pages 207–228, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bleha, S., Slivinsky, C., and Hussien, B. (1990). Computer-access security systems using keystroke dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12):1217–1222.

Cho, S., Han, C., Hee, D., and Kim, H.-I. (2000). Web-based keystroke dynamics identity verification using neural network. *Journal of Organizational Computing and Electronic Commerce*, 10:295–307.

Davarci, E. and Anarim, E. (2022). User identification on smartphones with motion sensors and touching behaviors. In *2022 30th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification*. Wiley, New York, 2 edition.

Embarcadero (2024). Delphi community edition. https://www.embarcadero.com.

Fide, M. (2024). Boun cmu emulator. https://github.com/mfide.

Fide, M. and Anarım, E. (2024). User authentication with gru based siamese networks using keyboard usage behaviour. In *2024 32nd Signal Processing and Communications Applications Conference (SIU)*, pages 1–4.

Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742.

Haider, S., Abbas, A., and Zaidi, A. (2000). A multi-technique approach for user identification through keystroke dynamics. In *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics. 'cybernetics evolving to systems, humans, organizations, and their complex interactions' (cat. no.0*, volume 2, pages 1336–1341 vol.2.

IEEE (2008). Ieee standard for a precision clock synchronization protocol for networked measurement and control systems. In *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pages 1–269.

Joyce, R. and Gupta, G. (1990). Identity authentication based on keystroke latencies. *Commun. ACM*, 33(2):168–176.

Kang, P., Hwang, S.-s., and Cho, S. (2007a). Continual retraining of keystroke dynamics based authenticator. In *Proceedings of the 2007 International Conference on Advances in Biometrics*, ICB'07, page 1203–1211, Berlin, Heidelberg. Springer-Verlag.

Kang, P., Hwang, S.-s., and Cho, S. (2007b). Continual retraining of keystroke dynamics based authenticator. In *Proceedings of the 2007 International Conference on Advances in Biometrics*, ICB'07, page 1203–1211, Berlin, Heidelberg. Springer-Verlag.

Killourhy, K. S. and Maxion, R. A. (2009). Comparing anomaly-detection algorithms for keystroke dynamics. In *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, pages 125–134.

Lange, T. and Winterhof, A. (2003). Interpolation of the elliptic curve diffie-hellman mapping. In Fossorier, M., Høholdt, T., and Poli, A., editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pages 51–60, Berlin, Heidelberg. Springer Berlin Heidelberg.

Medvedev, V., Budžys, A., and Kurasova, O. (2023). Enhancing keystroke biometric authentication using deep learning techniques. In *2023 18th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6.

Microsoft (2024). High-resolution timers. https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/high-resolution-timers.

Morales, A., Fierrez, J., Gomez-Barrero, M., Ortega-Garcia, J., Daza, R., Monaco, J. V., Montalvão, J., Canuto, J., and George, A. (2016a). Kboc: Keystroke biometrics ongoing competition. In *2016 IEEE 8th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–6.

Morales, A., Fierrez, J., Tolosana, R., Ortega-Garcia, J., Galbally, J., Gomez-Barrero, M., Anjos, A., and Marcel, S. (2016b). Keystroke biometrics ongoing competition. *IEEE Access*, 4:7736–7746.

Orebaugh, A., Ramirez, G., Beale, J., and Wright, J. (2007). *Wireshark & Ethereal Network Protocol Analyzer Toolkit*. Syngress Publishing.

Ostroukh, A. V., Pronin, C. B., Podberezkin, A. A., Podberezkina, J. V., and Volkov, A. M. (2024). Enhancing corporate network security and performance: A comprehensive evaluation of wireguard as a next-generation vpn solution. In *2024 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO)*, pages 1–5.

Peacock, A., Ke, X., and Wilkerson, M. (2004). Typing patterns: a key to user identification. *IEEE Security & Privacy*, 2(5):40–47.

Peterson, L. L. and Davie, B. S. (1996). *Computer net-*

works: a systems approach. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Rennie, J. D. M., Shih, L., Teevan, J., and Karger, D. R. (2003). Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, page 616–623. AAAI Press.

Ylonen, T. (2019). Ssh key management challenges and requirements. In *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5.

Yu, E. and Cho, S. (2003). Ga-svm wrapper approach for feature subset selection in keystroke dynamics identity verification. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 3, pages 2253–2257 vol.3.

Yıldırım, M. and Anarım, E. (2019). Session-based user authentication via mouse dynamics. In *2019 27th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4.

Zhong, Y., Deng, Y., and Jain, A. K. (2012). Keystroke dynamics for user authentication. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 117–123.