ON THE USER KEYBOARD USAGE BEHAVIOR ANALYSIS

by

Mehmet Fide

B.S, in Electronics Engineering, Erciyes University, 2003

Executive MBA, in Management, Boğaziçi University, 2016

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical Electronics Engineering

Boğaziçi University

2025

ACKNOWLEDGEMENTS

First and foremost, I want to extend my heartfelt thanks to my thesis supervisor, Prof. Emin Anarım, for bringing me into the PAM project. His direct and insightful guidance greatly streamlined my research journey.

I'm also incredibly grateful to Prof. Mutlu Koca for his constant support and inspiring lectures during my master's studies. His invaluable feedback significantly enhanced the quality of this thesis. A big thank you to Assoc. Prof. Ali Emre Pusane for his good humor and guidance, and to Mete Yıldırım for introducing me to the fascinating subject of keyboard biometrics, which forms the foundation of this work.

I owe a deep gratitude to my beloved sweetheart, Ceren Fide. Her unwavering patience, encouragement, and support have been my rock throughout this journey. Her motivational words were the beacon of hope during challenging times, helping me see the light at the end of the tunnel.

I would also like to thank my uncle, Oğuz Orhan, for sharing his innovative ideas and engaging in thought-provoking discussions.

Special thanks to Erhan Davarcı, Negin Kazami, and Mehmet Emin Kazanç for their assistance during the preparation of this thesis. I truly appreciate each of you for making the world a more understandable and enjoyable place.

This thesis was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under the grants for the Cloud-Based Privileged Access Management System, Project No 117R030. The ideas, views, and conclusions expressed here are my own and do not necessarily reflect TUBITAK's official stance.

ABSTRACT

ON THE USER KEYBOARD USAGE BEHAVIOR ANALYSIS

Although many services available today in emerging information technologies started to use fingerprint, electronic key, face recognition methods to identify authorized or privileged users, still the majority usage of such services requires keyboard typed password. In order to increase the security and prevent unwanted accesses, security authorities mandate users to pick a password that has enough complexity and change it periodically but by ignoring the share of the password intentionally or unintentionally with third persons that workarounds all complexity relied on for security. This thesis is the part of more general topic that is Privileged Access Management (PAM). Firstly, this study examines user keyboard behavior, extracting features from the data collected during brief text entry, and attempts to authenticate the user's identity using various classification methods. The performance metrics of these methods are then calculated and compared.

Secondly, in addition to classifiers found in the literature, a new GRU-based Siamese network architecture has been designed. The results of this architecture have been analyzed in the presence of leakage, and a method that completely prevents leakage has been proposed and the results shared.

Finally, since the PAM project aims to offer all its services and functions in a cloud environment, CMU's keystroke database will be simulated in a network environment through a real-time embedded system. The performance of the proposed classification methods will be examined and compared under the conditions of new data collected using protocols such as UDP, TCP, and SSL.

ÖZET

KULLANICININ KLAVYE KULLANIM ALIŞKANLIKLARI ANALİZİ

Gelişen bilişim teknolojileri alanında yer alan birçok hizmet, yetkilendirilmiş veya ayrıcalıklı kullanıcılarını ayırt edebilmek için, parmak izi okuyucu, elektronik anahtar, yüz tanıma gibi yöntemleri kullanmaya başlamış olsa da yaygın kullanım hâlâ klavye üzerinden parola girişinin yapılması yönündedir. Güvenlik otoriteleri, kullanıcıların parolalarını yeterince karmaşık seçmelerini ve belirli süreler zarfında değiştiriyor olmalarını yeterli görmekte, kullanılan parolanın üçüncü şahıslarla istemli ya da istemsiz paylaşılması durumunda oluşan zafiyeti göz ardı etmektedirler. Bu tez çalışması, daha kapsamlı bir çalışma olan Ayrıcalıklı Erişim Yönetimi (AEY) projesinin bir alt parçasıdır. Bu doğrultuda birinci olarak kullanıcının klavye kullanım davranışlarını inceleyerek, kısa bir metin girişi esnasındaki verilerden özniteliklerini çıkartarak farklı sınıflandırma yöntemleri ile kullanıcının kimliğini doğrulamaya çalışmış ve başarım ölçütleri hesaplanarak karşılaştırılmıştır.

İkinci olarak, literatürdeki sınıflandırıcılara ek olarak yeni bir GRU temelli Siyamikizi ağ mimarisini temel alan bir doğrulayıcı tasarlanmış, bunların sonuçları sızıntı varlığında incelenmiş, sızıntıyı tamamen önleyen bir yöntem önerilerek sonuçlar paylaşılmıştır.

Üçüncü ve son olarak ise AEY projesi, tüm servis ve hizmetlerini bulut ortamında sunmayı hedeflediğinden, CMU'nun tuştakımı veritabanı gerçek zamanlı gömülü bir sistem vasıtasıyla ağ ortamında simüle edilecek, UDP, TCP ve SSL kullanan protokollerin varlığı altında toplanan yeni verilerin kullanımı durumunda önerilen sınıflandırma yöntemlerinin değişen performansları incelenmiş ve kıyas edilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

a_i	Mean Absolute Deviation of i-th row of feature set
C_c	Center of a sphere
$f(.)$	Function of a given variable
$\mathcal{L}_{\mathrm{CF}}$	Contrastive Loss Function
L_{ij}	i-th and j-th class identifier
m	Sample number
M	Mean matrix
n	Dimensions
P	Feature selection row vector
r_c	Radius of a sphere
S	Anomaly score
U	Unitary Matrix
V	Unitary Matrix
x_i	i-th Row of feature set
X	Random Variable
y_i	Mean value of i-th row of feature set
z_i	i-th row of z-score
η_x	z-score threshold
μ_j	j-th mean vector of μ_X
μ_X	Row vector mean of X
σ	Standard deviation
σ^2	Variance
Σ_{ij}	i-th and j-th element of covariance matrix
$\Sigma_{n\times n}$	n-by-n covariance matrix
Φ	Principle square root

LIST OF ACRONYMS/ABBREVIATIONS

ACC	Accuracy
ACO	Ant Colony Optimization
AES128	Advanced Encryption Standard - 128 bit key
AES256	Advanced Encryption Standard - 256 bit key
AEY	Ayrıcalıklı Erişim Yönetimi
AUC	Area Under the Curve
BPNN	Back Propagation Neural Network
CER	Crossover Error Rate
CMU	Carnegie Mellon University
EER	Equal Error Rate
FAR	False Acceptance Rate
FIN	Finalize Flag
FN	False Negative
FNR	False Negative Rates
FP	False Positive
FPR	False Positive Rates
FRR	False Rejection Rate
GA	Genetic Algorithm
GRU	Gated Recurrent Units
HMM	Hidden Markov Model
IAM	Identity Access Management
IP	Internet Protocol
KB	Keystroke Bio-metrics
KPI	Key Performance Indicators
LCD	Liquid-crystal display
LED	Light-emitting diode
LFN	Large Fat Network-pronounced elephant
LSTM	Long Short-Term Memory

LWIP	Lightweight TCP/IP stack
MAD	Mean Absolute Deviation
NN	Neural-Network
OSI	Open Systems Interconnections
PAM	Privileged Access Management
POHMM	Partially Observable Hidden Markov Model
PS/2	Personal System/2
PSR	Principle Square Root
RFC	Request for Comment
ROC	Receiver Operating Characteristic
RSA	Rivest–Shamir–Adleman - asymmetric cryptography algorithm
RTC	Real Time Clock
RTT	Round Trip Time
SD	Standard Deviation
SSH	Secure Shell
SSL	Secure Sockets Layer
SVD	Singular Value Decomposition
SVM	Support Vector Machines
SYN	Synchronization Flag
SWS	Silly Window Syndrome
TCP	Transmission Control Protocol
TN	True Negative
TLS	Transport Layer Security
TP	True Positive
TUBITAK	Scientific and Technological Research Council of Turkey
TXCO	Temperature Compensated Crystal Oscillator
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Data-gram Protocol
URG	Urgent Pointer Flag
USB	Universal Serial Bus

ZMFAR	Zero-Miss False-Alarm Rate

1. INTRODUCTION

1.1. Privileged Access Management

The number of privileged or authorized transactions in various services and applications within the field of information technology has been rapidly increasing. Improper management of accounts with extensive privileges poses significant security risks. Most targeted attacks aim to acquire privileged transaction information and exploit the associated privileges to execute the attack. Following the implementation of security measures at the perimeter and endpoint levels in institutional networks, privileged access security has emerged as a critical issue in systems and applications. However, it is often overlooked or not adequately addressed. In the network security sector, the concept of Privileged Access Management (PAM) encompasses various tools and methods designed to secure critical transactions, particularly at the administrative level. This approach can be effectively modeled as privileged identity management.

Proper management of processes associated with privileged transactions and service types, along with the logging of system access for future reference, is crucial for both security management and compliance with standards such as DSS, PCI, and ISO 27001. Consequently, both Privileged Access Management (PAM) and Identity Access Management (IAM) systems are increasingly in demand, with their market shares expanding daily. A primary objective of the PAM system is to anticipate potential attacks on network and cloud systems, mitigate the associated damage through proactive measures, conduct financial evaluations, and analyze the potential losses resulting from such attacks. Additionally, PAM systems aim to prevent intrusions at an early stage.

In this context, the primary objective of the project is to develop a unique cloud-based Privileged Access Management (PAM) system characterized by high security and ease of adaptability. The project's originality lies in its proposed cloud-based PAM

system, featuring a cloud-compliant architecture, dynamic anomaly detection, user behavior analysis, prioritization, contextual and content data analysis, cloud-compliant coding algorithms, countermeasures, and business cost efficiency. It aims to address a significant gap in the academic literature. The PAM system proposed in this project incorporates numerous novel features and capabilities compared to existing software products in the sector and established methods in related academic literature. A crucial goal of the project is to transform the proposed cloud-compliant PAM system into a product or a unique industrial software solution with the potential for patentability and competitive strength in both national and international markets through university-industry collaboration.

Efficiently managing employee access to sensitive applications and data is one of the most critical security challenges faced by organizations today. In general, large institutions are striving to manage access priorities for millions of users within vast Information Technology environments [1]. Employees possess greater authority than required for most of their tasks due to lack of proper user management methods. Additionally, the inability to centrally manage organizational directives and policies exacerbates this issue. As a result, institutions implement a Privileged Access Management (PAM) system across the company to centrally manage authorized transactions [2]. This system ensures that employees use their standardized privileges, thereby reducing security gaps and ensuring compliance with national and international regulations.

A traditional Privileged Access Management (PAM) system comprises three fundamental components: a database where user data is stored, a functional tool that ensures the automation and execution of user management tasks, and a set of policies that manage the PAM system itself [3]. The database stores identity information, authorization definitions, and authorization data. The functional section handles transactions such as user management, access management, data management, peer management, and user authorization, in accordance with predefined policies, and automates these processes [4]. Policies define the behavior of the system based on certain parameters [5]. In the current systems, policy management is often performed manually and

heavily relies on Information Technology managers. These managers typically work with loosely structured policy definitions or pre-existing policy management frameworks. Furthermore, these transactions are conducted using static data, such as the department where an employee works. This scenario diminishes the benefits of centralized user management, introduces security gaps, and results in outdated policies. Consequently, to reflect organizational and technological changes, it becomes necessary to develop policies over time.

To create system policies more efficiently and overcome the limitations of traditional PAM systems, it has been proposed to automate policy management, referred to as "Dynamic Policy Management". In this system, policy management consists of a Policy Mining Engine and Policy Storage, separate from traditional policy management. Two key elements in this new structured system are Contextual Data and Key Performance Indicators (KPIs). KPIs are essential tools used by all company employees today. Contextual data summarizes employee account information, location, activity, time, and relevance. KPIs, combined with contextual data, provide a solution to manage the load on modern PAM systems and prevent an overwhelming volume of transactions. KPIs can be defined as thresholds for normal user behavior, automatically created by associating and processing static and contextual data within the system. By determining normal user behaviors, measures can be taken when activities approach or exceed these thresholds, signaling anomalies. The Dynamic Policy Management Process aims to enhance information security through consistent policy updates, simplify the correction and modeling of PAM engineers' policies, and create initial policies with minimal effort. This process consists of four main steps:

(i) Infrastructure Setup:
    (a) Security Policies: The policies related with security and authorization, take place in this scope.
    (b) Process Policies: These are obligatory policies affecting values of the organization independent from the user.
    (c) Implicit PAM Policies: These are positive or negative policy suggestions

obtained as a result of an evaluation according very long threshold value and created as <user, item, context, rating>.

(ii) Data Collection: It is desired to create a periodical and completely automatic data loading process. Obtained raw data is normalized and converted into required or desired format and is stored.

(iii) Data Correlation and Policy Mining: Identity information, contextual data and data mining over various multidimensional methods are applied. Normal and abnormal behaviors are obtained. "Data creating", "data analysis" and "contextual evaluation" steps are applied for below three types of policy models.

  (a) Security Policies: The policies related with security and authorization, take place in this scope.

  (b) Process Policies: These are obligatory policies affecting values of the organization independent from the user.

  (c) Implicit PAM Policies: These are positive or negative policy suggestions obtained as a result of an evaluation according very long threshold value and created as <user, item, context, rating>.

(iv) Data Validation and Recommendation: It is to examine obtained potential policies. It is required to be approved before submitting to the PAM engineers as suggestion. After that, it is understood by the help of visual tools, it can be interpreted and outputs are obtained.

In the newly implemented system, Policy Mining experts incorporate contextual data alongside static user data, process this information, and store it for future use. This process is undertaken to discover undocumented but utilized policies and to identify Key Performance Indicators (KPIs). By monitoring employee activities over a sufficient period, it becomes possible to update old policies or generate new ones. However, the high volume of transactions and data within the system increases the computational load, resulting in significant costs for the organization. Moreover, to prevent security vulnerabilities in the system, these processes must be performed promptly. Another issue is the absence of a feedback loop for the policies established within the system, which hinders the ability to measure the security and accuracy of

these policies.

The proposed system aims to reduce both costs and potential abuse by leveraging the transaction capabilities of the cloud system and enabling quicker anomaly detection. This approach ensures that losses are minimized in the event of any adverse scenarios within the organization. Additionally, by implementing a feedback system to audit established policies, the auditing of the Dynamic Policy Management process will be ensured. Consequently, policy management within the PAM system will become faster, more efficient, and more secure.

The Feedback Loop in the PAM system is available in several different ways. These include:

(i) As a result of processing contextual data, developing the suggested policies and available ones and updating them in a way to meet requirement (Dynamic Policy Management Process),
(ii) Adding the most appropriate countermeasures those found as a result of the risk analysis in terms of financial to the current policies or improving them when it is required (Selecting countermeasure),
(iii) Presenting investment options those will improve the system for the decision makers of the organization by using user data with periodical evaluations (Suggesting the finest security investment).

The computational efficiency of cloud-based systems, the accessibility to these systems at any time and from anywhere, the readiness for customization, the provision of a collaborative workspace for cloud users, adjustment to transaction power requirements, and lower costs are key factors that make cloud systems attractive [6]. However, alongside these advantages, certain disadvantages do exist. One crucial element in the PAM system to be implemented is access consistency (availability), which is essential for security systems. In other words, the consistency of user activities is of great importance. Therefore, security is a primary concern among the challenges encountered

in cloud systems. In the proposed system, specific security measures will address the security gaps of cloud systems in multiple aspects. The unique characteristics of the system, which make it more successful than traditional PAM systems, can be summarized under five main categories. These include modeling user behaviors and creating data sets, measuring and improving the financial performance of the system, identifying anomalies, classifying and modeling big data, and developing cloud-compliant coding algorithms.

1.2. Modeling User Behaviors and Creating Data Set

Users leverage cloud systems to access various services, including distributed computations, database operations, web services, file sharing, and version control systems. The approaches and methods users employ to access these services vary significantly. Users may utilize different standards or specifically designed tools across various operating systems, following individualized paths based on their priorities. Cloud system providers are interested in understanding the diverse methods users employ, their priorities, and the reasons behind their choices. Modeling user behaviors can facilitate the work of service providers and aid in limiting and detecting unauthorized uses as well as verifying user identities.

User behaviors can be defined as the aggregate of activities performed while using these services and the methods employed to access the cloud system. Given that users connect to the cloud system for specific services, their actions can be anticipated. These behaviors generally consist of sequential steps that are interrelated, though they may vary dynamically based on system responses and desired outcomes. Nonetheless, it is possible to extract user fingerprints through data mining of user behavior.

A key characteristic of privileged access management is the real-time detection of unauthorized persons using authorized users' credentials. Therefore, authorized users are consistently monitored, and characteristic information is extracted from cloud system interactions. In non-cloud-based systems, physical features such as keyboard usage

speed, mouse double-click speed, acceleration, and movement speed can be considered. However, access to cloud-based systems should primarily be assessed through Transmission Control Protocol (TCP). Due to unpredictable network delays, locally performed physical movements are sent to the cloud system as command sequences or TCP streams in single packets. Consequently, the same physical features may not be used.

1.3. Data Collection Environment

Data can be categorized into two types: synchronous and asynchronous. User movements are generated asynchronously, while data from the source layer is created and recorded periodically. Although contemporary operating systems record data from the source and service layers, there is no standard normalization operation to ensure comparability between different systems. Additionally, they do not facilitate the recording of session, application, and command layer data for user behaviors. In the proposed system, this data will be collected within the cloud environment and analyzed within the scope of data mining. The development of an agent to evaluate this data will be a key aspect of the project.

1.4. Layered User Behavior

The data set is created using various parameters, including the activity periods of programs used in cloud systems, the vocabulary used by the user (command set) and their distribution, the services utilized, as well as hard disk, RAM, CPU usage rates, and network bandwidth usage characteristics. The data set will be evaluated in a layered structure, as suggested by Hlavacs and Kotsis in 1999 [7], and will be categorized and blended into five different categories:

(i) Session Layer: Session layer follows how long the user used the application. The user can start and end more than one session in the same time.

(ii) Application Layer: In this layer, interaction of the user with selected applica-

tion shall be monitored. The order of the commands given by the user shall be evaluated separately for each session.

(iii) Command Layer: It is the layer consisting text type orders and not consisting graphical interface and be used in lower level transaction by the user. Here, the user can start or end any service or can change parameters of a service that is operational (for instance, Web Service and SQL Server Service).

(iv) Service Layer: This layer shall study usages of known internet services by the user. For instance, in TCP port numbered 80, by gathering user's web service usage, or by using video and voice streams through UDP ports, it shall be tried to create a data-set for user profile.

(v) Resource Layer: Usages of the user for physical layer such as processor, memory, bandwidth, shall be added to data-set.

The first three layers, session, application, and command—directly define the user's behavior. In contrast, the service and source layers reflect indirect results of the user's behaviors.

While layered structures in the literature do not provide a complete picture, they are available for predicting user identity, with algorithms and suggestions proposed separately. The hybrid system model suggested by Dhanalakshmi and Lakshmi [8] for the session layer, the GSP algorithm suggested by Zhu [9] for the application layer, and spam detection methods and models based on text in the command layer are notable. In Kaur's 2016 study [10], n-gram and opinion-based types of Random Forest, SVM, Naive Bayesian, and KNN algorithms were compared, concluding that Random Forest demonstrated the best performance. Within the scope of this project, estimated information compiled in each layer will be integrated using the Kalman Filter [11] to ensure the most accurate estimation. Compiling all layers within such an architecture in a cloud environment represents a unique study and will be attempted for the first time.

1.5. Keyboard Usage

One of the key objectives for anomaly detection models is to mitigate potential security threats to the system. Whenever passwords are used to gain authorized access to system resources, there is always a risk of an imposter accessing the system using compromised passwords. Intrusion attempts can originate from both external and internal sources. External attackers often test common or shared passwords to infiltrate the system. Insider threats may arise from users who gather shared or cracked passwords and use them maliciously, especially if they are disgruntled or have been terminated, seeking revenge against the organization. Insiders might also sell confidential information to competitors for personal gain. Enhancing security can be achieved by adding an additional layer, such as analyzing typing rhythms, as an alternative method to filter users and add an extra security feature for authorizing system access. Keystroke rhythms, or the analysis of typing patterns, allow detection models to differentiate between imposters and genuine users, serving as a biometric identification method. The principle is that when an imposter uses a compromised password, their typing rhythm will differ from that of the genuine user, leading to authentication rejection. Evaluating the performance of anomaly detection models in keystroke dynamics is essential to establish their reliability and suitability for access authorization requirements. Several international standards specify metrics for access control system acceptance. According to European quality standards [12], the total false alarm rate in the system should be less than 1%, and the miss rate should not exceed 0.001% to be accepted as an access control model. The evaluation criteria and subsequent findings will provide a framework to investigate and identify models that meet these benchmarks, enabling organizations to select appropriate models for their access authorization needs. Identifying an anomaly-detection strategy that outperforms others involves recognizing the characteristics that contribute to its superior performance. These insights will form the basis for further investigation to improve existing performance standards. However, the current investigation's constraints include the necessity for consistent evaluation methodologies across all detectors. Variations in evaluation procedures between models result in unreliable results.

G. Forsen, M. Nelson and R. Staron, Jr. in 1977 [13] were the first to propose and provide supporting evidence that typing rhythms are highly unique and can differentiate users. Literature on keystroke dynamics is valuable for comparing different models [14]. Gaines, R. Stockton and William Lisowski and S. James Press and Norman Shapiro in 1980 [15] compared keystroke rhythms for passages and passwords, finding that different evaluation data and procedures are required for password rhythms versus simple text rhythms. One-class anomaly detection procedures differ from multiple-class procedures. One-class anomaly detection creates a typing sample for a single genuine user. When new samples are provided, the detector compares the new behavior to the established pattern, generating an anomaly score. Multiple-class detection procedures involve groups of users, developing a decision model based on collective typing rhythms to distinguish between users.

Password typing data invariably includes timing information. Each anomaly detector uses different criteria to evaluate user genuineness, leading to challenges in comparing them on a common platform. Password timing detection is complex, involving intervals between consecutive key presses, hold times, and inter-key delays. Although various detectors employ these timing conventions, drawing definitive conclusions about their relative performance remains premature. Literature reports instances where one keystroke dynamics-based detector outperforms another in terms of miss rates and false alarm rates. However, several factors, such as training methodologies, data sets, and evaluation frequency, impact performance outcomes. Reliable evaluation requires consistent evaluation platforms. Reported low error rates alone cannot distinguish detector performance. Comprehensive evaluation procedures, standards, and routines are necessary for meaningful comparison. The current study incorporates these aspects into its evaluation.

The aim of this study is to validate and compare existing methods in the literature that distinguish users based on data generated during password entry. The findings will inform the development and enhancement of constituent PAM systems.

2. REVIEW OF DETECTION METHODS

One of the key objectives of the current study was focused on the following three aspects,

(i) Developing a common dataset for training and testing the algorithms. The dataset needs to be same for all the anomaly detection models and to be employed accordingly.
(ii) The Evaluation procedures to be followed are similar for all the anomaly detection models and the common features of the evaluation procedure should provide basis for reliable comparison of the performance.
(iii) The performance measurement of the anomaly detection models make up the final aspects of the evaluation and they can be compared on equal basis for all the detection models.

Some of the detectors identified for comparison studies include Manhattan, Euclidean, Mahalanobis, Nearest Neighbor, Neural network, Fuzzy Logic, Outlier Count, One Class SVM, k-means etc.

2.1. Dataset

In order to validate methods that detect genuine or imposter users we need an input data set belong to different users. Due to wide usage in the recent literature, we picked Keystroke dataset [16] provided by Carnegie Mellon University (CMU).

This data-set contains keystrokes belong to 51 different users. For whole eight days and in different time slices of days, they are requested to enter password text ".tie5Roanl". They repeated the action fifty times for each days and for each correctly typed password characters, it is added to data-set with event time stamps. Password Data collection is based on following purposes:

(i) In spite of letting each user to select his/her own password, a single password is allowed by all the people to use for the sake of comparison.

(ii) A publicly available password generator [17] is employed for this purpose, a password strength checker [18] is used, and the total characters of the password are set as 10.

(iii) Password is selected to contain letters, numbers, and special characters. Also included with changes in the capitalization and punctuations to make it represent the best password possible.

(iv) The password employed in the study is ".tie5Roanl"

(v) The length of the password is optimized, neither allowed to have more characters like 13 and more and also not allowed to have small number of characters. This is due to the fact that when there are more number of characters, the typing of password has become much easier as there are easily type-able phrases and names found in.

To collect the accurate time stamps, it has been used a laptop computer that runs a Windows XP operating system and a logger program equipped with a special hardware called high precision external timer. The setup is also tested with a high accuracy signal generator and it has been observed that accuracy of the logger was within $\pm 200\,\mu s$ [16].

The data-set doesn't include wrongly typed password timings. Also users who have relatively huge variation in their typing are filtered out. For each users, there are 400 records in the data-set and each row contains below fields:

- *subject*: 51 Users labeled as from $s002$ to $s057$.
- *sessionIndex*: Daily sessions, total experiment is 8 days long so it is labeled from 1 to 8.
- *rep*: Repetition index in a session from 1 to 50.
- *H.period*: Hold duration for the key ".".
- *DD.period.t*: Time duration between the key "." pressed and "t" pressed.

- $UD.period.t$: Time duration between the key "." released and "t" pressed.
- $H.t$: Hold duration for the key "t".
- $DD.t.i$: Time duration between events the key "t" pressed and "i" pressed.
- $UD.t.i$: Time duration between the key "t" released and "i" pressed.
- $H.i$: Hold duration for the key "i".
- $DD.i.e$: Time duration between events the key "i" pressed and "e" pressed.
- $UD.i.e$: Time duration between the key "i" released and "e" pressed.
- $H.e$: Hold duration for the key "e".
- $DD.e.five$: Time duration between events the key "e" pressed and "5" pressed.
- $UD.e.five$: Time duration between the key "e" released and "5" pressed.
- $H.five$: Hold duration for the key "5".
- $DD.five.Shift.r$: Time between the key "5" pressed and "Shift+r" pressed.
- $UD.five.Shift.r$: Time between the key "5" released and "Shift+r" pressed.
- $H.Shift.r$: Hold duration for the key "Shift+r".
- $DD.Shift.r.o$: Time between events the key "Shift+r" pressed and "o" pressed.
- $UD.Shift.r.o$: Time between the key "Shift+r" released and "o" pressed.
- $H.o$: Hold duration for the key "o".
- $DD.o.a$: Time duration between events the key "o" pressed and "a" pressed.
- $UD.o.a$: Time duration between the key "o" released and "a" pressed.
- $H.a$: Hold duration for the key "a".
- $DD.a.n$: Time duration between events the key "a" pressed and "n" pressed.
- $UD.a.n$: Time duration between the key "a" released and "n" pressed.
- $H.n$: Hold duration for the key "n".
- $DD.n.l$: Time duration between events the key "n" pressed and "l" pressed.
- $UD.n.l$: Time duration between the key "n" released and "l" pressed.
- $H.l$: Hold duration for the key "l".
- $DD.l.Return$: Time between events the key "l" pressed and "Enter" pressed.
- $UD.l.Return$: Time duration between the key "l" released and "Enter" pressed.
- $H.Return$: Hold duration for the key "Enter".

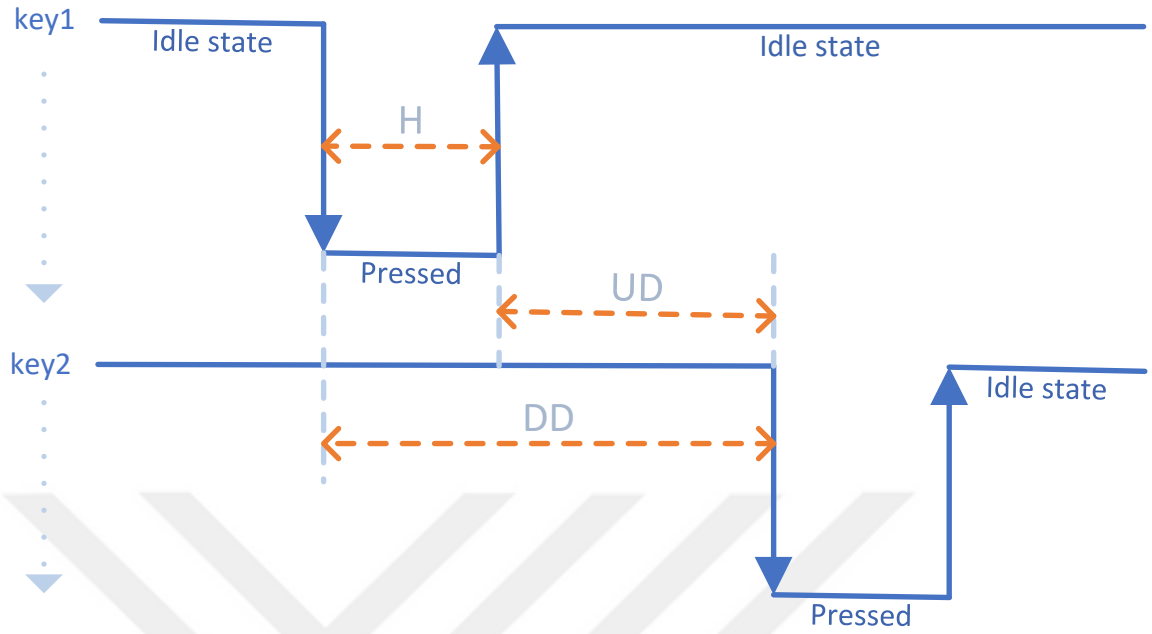

Figure 2.1. CMU dataset - event presentation.

Figure 2.1 represents timing data shown as H, DD and UD. Respectively, H, DD and UD timings can be written as

$$DD = H + UD, \tag{2.1}$$

$$UD = DD - H, \tag{2.2}$$

$$H > DD. \tag{2.3}$$

As we can clearly see in Equation (2.2), "UD" can be negative when condition in Equation (2.3) is met.

The password chosen consists of alphanumerical and non-alphanumerical characters. As we can see in Figure 2.2, total typing times of fully correct password dramatically reduced by time as long as the users get familiar with the typing.

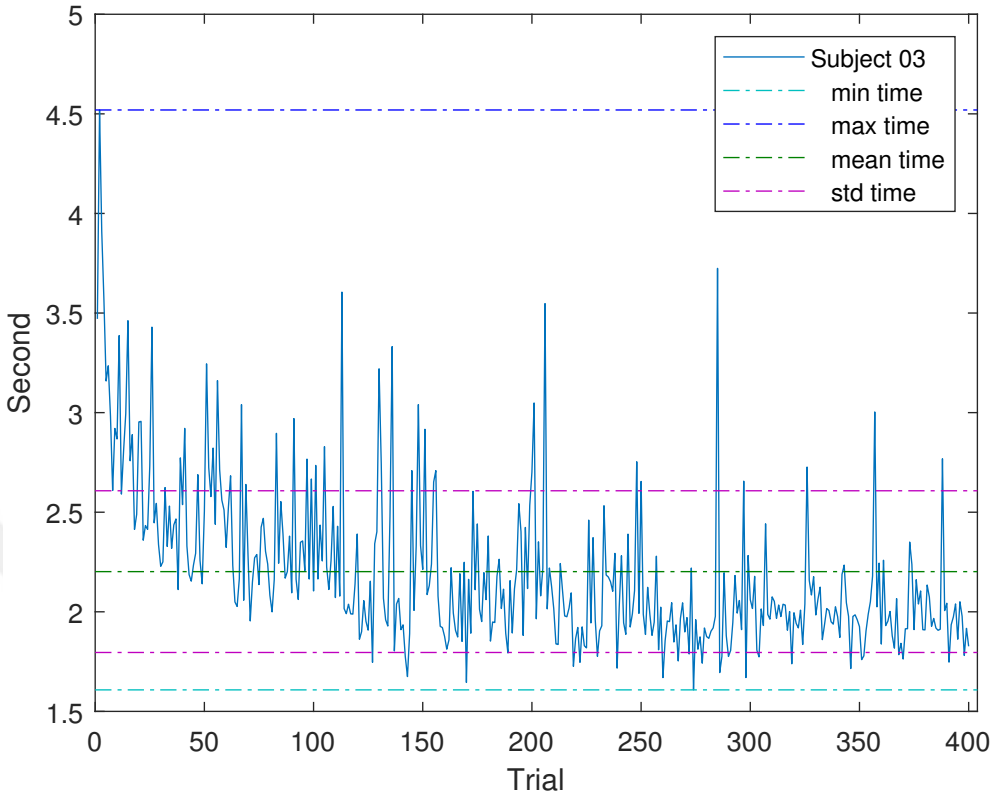

Figure 2.2. Subject 03 - total timings over trials.

2.1.1. Data Collection

Keystroke benchmarked datasets are extensively utilized by researchers for data collection. Data can be gathered from various sources, such as simple electronic keyboards or pressure-sensitive devices. Even mobile phones and electronic keyboards can be used for data collection purposes. The type of data collected is crucial for obtaining suitable input for modeling. Typically, character-based text data input is used for data collection requirements; however, numerical or hybrid numerical and text-based data inputs are also applicable. Keystroke benchmarked datasets are highly versatile.

A laptop with an external keyboard is used for data collection for evaluation purposes. A Windows application is installed on the laptop, which prompts the user to enter a password. The application allows the user to type the password on the screen. The user can proceed to the next phase of the process only when the password is entered correctly. If the password is entered incorrectly, the application prompts the

user to re-enter it.

Each user is required to type the password 50 times to enable the system to collect data. The system extracts and collects all keystroke dynamics features, such as key down and key up events, the key name, the time of the event, and its occurrence. Additionally, timestamps are used to record data entry into the system. The clock used for this purpose is accurate up to 200 µs.

Inherent limitations include the fact that the evaluation is somewhat removed from reality, as actual users type passwords on their own systems at their convenience, whereas, in this evaluation, they use a supplied keyboard. Nonetheless, these limitations are mitigated by ensuring accurate time-stamping and creating a uniform environment as much as possible.

2.1.2. Running Subjects

Each subject is required to type approximately 50 passwords per session, with data collection extending over 8 sessions, resulting in a total of 400 datasets per user. The sessions are not conducted consecutively; subjects are permitted to take a day off between sessions. This approach is designed to account for normal day-to-day variations, thereby enhancing the ecological validity of the study and ensuring that the data closely reflect real-world conditions.

The subjects include both males and females to ensure gender diversity, with a total of 30 males and 21 females selected for the study. Additionally, the study includes 43 right-handed and 8 left-handed individuals from various ages to enhance the sample's diversity and realism.

2.1.3. Timing Vectors

Features sets collected from the data ie, the timing vectors varied from detector to detector. Some typical observations include the following;

- Enter key is made part of the password.
- Key down and key down times are computed.
- Key up and key down times are computed.
- Hold times are computed for all keys.
- 31 timing features are collected for each string and organized into a timing vector.
- Some of the features are expressed to be representative of combination of other key timing data elements.

2.2. Realizing Detectors

For the purpose of the study, various anomaly detection algorithms from the literature, which utilize password typing timings, are implemented from scratch in MATLAB and Python environments.

Each detector has distinct training and test phases. During the training phase, a portion of the genuine users' password timings is used. For the test phase, the remaining portion of the genuine users' password timings, which were not used in the training phase, along with all other users' timing sets, are used as imposters.

2.3. Performance Comparison Criteria

We used Equal Error Rate (EER) suggested by Kang [19] in 2007 to compare performance of various anomaly detectors. To calculate EER, threshold for the detector is chosen such that false-negative rates (FNR) and false positive rates (FPR) are equal as shown in Figure 2.3. Lower EER values imply better detector performance since EER is a kind of error ratio. In some literature, it is also known as crossover error

rate (CER) [20]. Confusion Matrix is provided in the Table 2.1. False Positive Rates, False Negative Rates and Accuracy (ACC) at threshold value of EER point can be calculated as

$$FPR = \frac{FP}{FP+TN}, \tag{2.4}$$

$$FNR = \frac{FN}{FN+TP}, \tag{2.5}$$

$$ACC = \frac{TP+TN}{P+N}. \tag{2.6}$$

Another important performance criterion used in classification literature is the ZM-FAR, which is calculated by selecting a threshold value that sets the FAR to zero and minimizes the FRR.

Table 2.1. Confusion Matrix.

	True Positive	**True Negative**
Positive Prediction	True Positive (TP)	False Positive (FP)
Negative Prediction	False Negative (FN)	True Negative (TN)

Claimant validation is pivotal in the experiment. The objective is to determine whether the claimant is indeed the person they claim to be and whether they correspond to the individual associated with the given sample. The Classification Accuracy Rate (ACC) is the metric used to evaluate the overall performance of the model. The classification output scores form the basis for validating whether a sample is genuine or not. This verification method is also known as the authentication process.

CMU keystroke benchmark datasets are employed for user verification during the testing and validation of hybrid models in the current experimental setup. The process involves generating a test dataset by allowing approximately 8 data sample collections for each subject, each with around 50 repetitions, resulting in a total of 400 samples. The training data is always taken from the first 200 data of the user. Two latent values are generated for each typing event of the password string. Model evaluation based on

EER and ZMFAR is conducted in the current setup. The threshold acceptance rate is determined based on the FAR and FRR in accordance with the set EER and ROC ratings. A trade-off between FRR and FAR is generally expected. The decision is made based on threshold modeling, with threshold values determining the claimant's authenticity. The Area Under the Curve (AUC) classifies the system's performance.

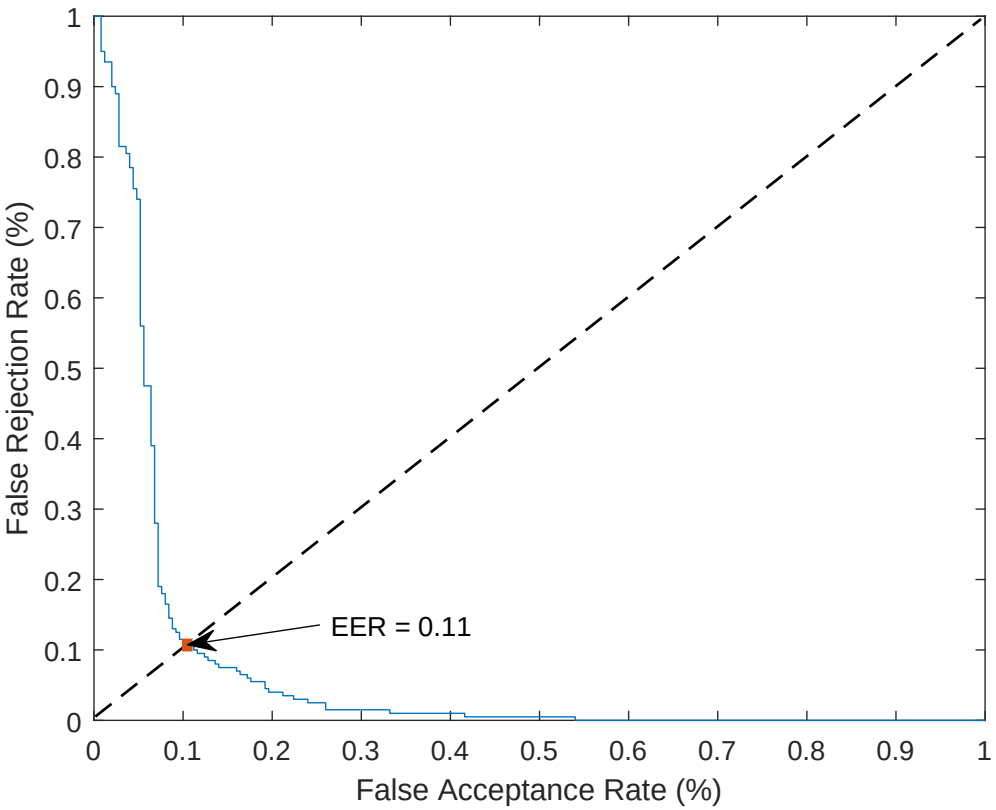

Figure 2.3. Subject 08 - equal error rate calculation.

2.4. Partially Observable Hidden Markov Model Method

Keystroke dynamics biometrics is a versatile method for user authentication, with numerous models available in the literature and widely used in practice. These models are primarily classified into two types: generative and discriminative models. The Support Vector Machine (SVM) as a discriminative model has been used for over a decade as a classifier. It is known for its high precision and suitability for big datasets without errors. On the other side, Hidden Markov Model (HMM) as one of generative models renowned for its efficiency in feature extraction across various applications, notably

speech recognition. However, despite its effectiveness in speech recognition, HMM performs less effectively in keystroke detectors compared to other classifiers. To address this, HMM replaced with Partially Observable Hidden Markov Model (POHMM) has been proposed by John V. Monaco and Charles C. Tappert [21] in 2018 to enhance its applicability. POHMM demonstrates improved performance over HMM, particularly in handling incomplete and infrequent data.

M. L. Ali, J. V. Monaco and C. C. Tappert in their study [22, 23], proposed a hybrid model combining the characteristics of POHMM and SVM for user authentication. This hybrid model leverages the strengths of both discriminative and generative models. Feature extraction is performed using POHMM, while anomaly detection is conducted using a one-class support vector machine. The Equal Error Rate (EER) is employed to evaluate the precision and accuracy of the biometric models. The proposed model achieves an EER of approximately 0.086, with a standard deviation of 0.063, based on tests using the CMU benchmark keystroke dataset. Combining generative models and discriminative models can significantly benefit biometric applications. This report outlines the dynamics of keystroke-based authentication, the principles of the proposed hybrid model.

Supervised machine learning involves understanding datasets from past observations and making predictions based on sensed data trends, despite uncertainties. Both generative and discriminative models are versatile and widely used. Common generative models include Gaussian models [24] proposed by Jiang, Cheng-Huang and Shieh, Shiuhpyng and Liu, Jen-Chien in 2007. Discriminative models used in applications include regression, SVM and neural networks. HMM is used in various applications, such as speech recognition, natural language processing, bioinformatics, face biometrics, character recognition, anomaly detection [16], and image database retrieval. HMM has shown remarkable performance in speech recognition and other applications like signature and gesture recognition. However, HMM remains less effective in keystroke biometric systems. POHMM, an extension of HMM, has demonstrated improved performance in user authentication for short fixed-input applications. SVM is extensively

applied in speech recognition, image classification, and keystroke biometrics, effectively distinguishing imposters from genuine samples due to its inherent ability to separate anomalies from genuine data. SVM is particularly advantageous in terms of accuracy, computational efficiency, large dataset analysis, and handling missing data.

Keystroke dynamics is an accessible biometric system based on the unique typing patterns of individuals. Each person exhibits different typing speeds, keystroke pressures, and intervals between consecutive key presses. Additionally, finger positions on keys vary among individuals. Capturing and analyzing this data can effectively differentiate imposters from genuine users. This method is non-intrusive, transparent, and respects user privacy, as typing rhythms are monitored continuously without interfering with the user's activities. The neuro-physiological aspects guiding typing rhythms are key to distinguishing genuine users from imposters. The simplicity of the technique and the minimal equipment required—merely an electronic keyboard—make it widely applicable. The model creates a unique profile for each user based on the following aspects:

2.4.1. Features Extraction

Based on typing rhythms, key features such as key hold time, key press time, key release delay, and key intervals are extracted for each user to create a unique profile. According to [25], these timing measurements are integrated into the system to gather data. The specific features extracted and used for validating test samples depend on the particular case. However, common features include key-to-key press latencies, keystroke timings, and key pressure applications, which are recorded as necessary.

The training of the model is crucial; the features extracted from user-supplied samples form the framework for testing. The number of test samples used will influence the subsequent accuracy of the hybrid model's performance.

2.4.2. Analytical Methodology

Statistical methods are utilized to classify key features extracted from keystroke biometrics data. Various statistical techniques are employed, including mean, median, standard deviation. Additionally, methods such as weighted probability strategies and distance measures like Euclidean, Manhattan, Bhattacharya [26], Mahalanobis, and Cosine Similarity [27] are used.

Currently, many researchers are focusing on machine learning techniques such as neural networks, fuzzy logic, and decision trees. Other classification models in use include evolutionary computing and support vector machines.

2.4.3. Decision

The decision is based on comparing the classifier outputs to a predetermined threshold to determine conformance or non-conformance. When a user profile matches to the reference patterns, a decision is made to either accept or reject the matching.

2.4.4. Retraining

Retraining is a technique used to capture evolving user patterns over time and under different environmental conditions. It ensures that the unique user template remains accurate despite these changes. The retraining process involves updating the existing user profile by recapturing data using methods such as growing windows, moving windows, intelligent modes, retraining with imposter patterns, and adaptive threshold levels for classification.

2.4.5. Verification

Verification or acceptance is the phase where the authentication of a genuine user is performed by evaluating a set of unknown user data samples. This process is

based on the actual system performance characteristics, with the EER determining system efficiency. In an optimal system, both the FAR and FRR should be minimal. FAR indicates the rate of which imposter profiles are erroneously accepted, while FRR represents the rate at which genuine users are erroneously rejected. The AUC measures the system's performance, with a value of one indicating optimal performance.

The new approach combines generative and discriminative models. Generative approaches focus on model classification, while discriminative models emphasize learning class boundaries. One-class and two-class SVMs are effective in understanding these boundaries, with differences between them arising from programming variations and differences in quadratic equation formation during training. SVMs are well-suited for large datasets. HMM has been a popular method for speech and pattern recognition for several years, utilizing hidden datasets linked to observed values dependent on underlying latent values. POHMM, a more advanced and customized version of HMM, outperforms other models by smoothing data distribution issues caused by missing and infrequent data [22].

POHMM generally outperforms SVM and other discriminative models, although SVM remains effective for continuous authentication with free text datasets. Few hybrid models have been applied to keyboard dynamics. Yu and Cho [28] proposed a GA-SVM (Genetic Algorithm) based model, combining General Algorithm and particle swarm optimization with SVM. Karnan and Akila [29] introduced a keystroke detector using Ant Colony Optimization (ACO) integrated with BPNN.

In this process, POHMM acts as an unsupervised feature extractor and then the results supplied to a SVM to correlate with test data and distinguish genuine user keystroke dynamics. The model training process involves POHMM extracting approximately 130 parameters per session, which are used to generate the SVM and develop a test model for comparison and evaluation. The hybrid model's performance is evaluated based on EER and related parameters, determining its ability to differentiate imposters from genuine users [30].

The POHMM/SVM hybrid model is found to be highly effective, with an EER of 0.086 and a standard deviation of 0.063 as seen in Table 2.3. The testing procedures involved training the hybrid model using CMU dataset. It uses key press-press and hold times which are modeled by a log-normal distribution. The claimant's typing profile likelihood is normalized by likelihoods of other users in the dataset. Then the normalized value is compared with a threshold to decide whether accept or reject the user based on correlation levels. The mean and standard deviation of the EER for each model are computed for comparison.

Table 2.2. POHMM with n=11 features.

POHMM Detector	**Mean**	**Std**
Equal-Error Rate	0.058	0.067
Area Under Curve	0.020	0.040
Accuracy	0.942	0.066

While pure POHMM performs best in terms of EER as seen in Table 2.2, the POHMM/SVM hybrid model outperforms SVM alone in short text testing which we evaluated later in Section 2.17.

Table 2.3. POHMM/SVM.

POHMM/SVM Detector	**Mean**	**Std**
Equal-Error Rate	0.086	0.063
Area Under Curve	0.038	-
Accuracy	0.914	0.062

2.4.6. Conclusion

The performance statistics demonstrated the efficiency of the POHMM/SVM hybrid model for continuous authentication using free text. The results highlighted the

effectiveness of POHMM as a feature extractor combined with a SVM as a classifier, showing superior performance compared to other models discussed in this thesis. These findings were based on training with a single password event. However, in real-world applications with more extensive datasets, the performance is expected to improve significantly, yielding even more promising results.

2.5. Bayesian Method

Bayesian method assumes that in an n dimensional vector space, X is a random variable consist of n different random variables from $[X_1, X_2, \cdots, X_n]$ and has a Multivariate Normal Distribution that is in another world, Joint Gaussian Distribution. We remember that single Gaussian Distribution represented by scalar mean μ and variance σ^2 values. In contrast to that Multivariate Normal Distribution represented as

$$P(x) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-(x-\mu_X)^T \Sigma^{-1} (x-\mu_X)/2}, \tag{2.7}$$

where x is a feature vector represented as $[x_1, x_2, \cdots, x_n]$, mean vector μ_X in length of n and Covariance Matrix $\Sigma_{n\times n}$, respectively.

First 200 feature vectors for training data are used to calculate μ_X and Σ. Suppose that M is a training matrix which each row contains individual samples and columns represent timing features. Remember that in the CMU dataset, it has been provided only 2 independent features per key pressing event, $H.key1$ and $DD.key1.key2$ or $H.key1$ and $UD.key1.key2$. Two out of these three parameters mathematically tied to each other. Therefore only 21 independent features available out of 31 features. The mean vector μ_X can be estimated as

$$\mu_j = \frac{1}{N} \sum_{i=1}^{N} M_{ij} \qquad j = 1 \ldots n, \tag{2.8}$$

and the covariance matrix Σ through N samples that each has n features can be estimated as

$$\Sigma_{ij} = \frac{1}{N} \sum_{k=1}^{N} (M_i - \mu_i) \times (M_k - \mu_k)^T \qquad i, j = 1 \ldots n. \tag{2.9}$$

Table 2.4. Bayesian with n=21 features.

Bayesian Detector	**Mean**	**Std**
Equal-Error Rate	0.110	0.065
Zero-Miss False-Alarm Rate	0.482	0.273
Accuracy	0.890	0.064

In the network, applications like TELNET, SSH use only key press event so if this is the case only available features that we can use in our detector is $DD.key1.key2$. If we limit the algorithm to use only these features, we obtain results provided in Table 2.5.

Table 2.5. Bayesian with n=10 DD features only.

Bayesian Detector	**Mean**	**Std**
Equal-Error Rate	0.153	0.067
Zero-Miss False-Alarm Rate	0.703	0.276
Accuracy	0.847	0.068

2.6. Euclidean Method

In this classical anomaly detection algorithm [31], features in the data-set are taken as points in an n dimensional vector space where n is the cardinality of features. It considers training data-set as if they were bunch of points and calculates center of these points. Later, anomaly score is calculated based on how far each test points to the center. S, the anomaly score is calculated as

$$S = \sum_{i=1}^{n} (y_i - x_i)^2, \tag{2.10}$$

where x is the center of training vectors and y is the test vector.

We have verified performance results of Euclidean Anomaly Detector suggested by K. Killourhy [16] shown in Table 2.6.

Table 2.6. Euclidean with n=31 features.

Euclidean Detector	**Mean**	**Std**
Equal-Error Rate	0.171	0.095
Zero-Miss False-Alarm Rate	0.875	0.200
Accuracy	0.829	0.095

Instead of all features, using selective futures can improve the EER. Lets assume that P is $1 \times n$ feature selection row vector those each elements consist of 0 or 1 and defined as

$$P = \arg\min_{p} \left[S(p) = \sum_{i=(p==1)} \left(y_i - x_i\right)^2 \right], \tag{2.11}$$

where x is the center of training vectors and y is the test vector. Solution to P gives us following results:

$$P = [1\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1]_{1\times 31}. \tag{2.12}$$

EER and Accuracy parameters are improved over feature selection but not Zero-Miss False-Alarm Rate as shown in Table 2.7.

Table 2.7. Euclidean with feature selection n=14 features.

Euclidean Detector	**Mean**	**Std**
Equal-Error Rate	0.156	0.073
Zero-Miss False-Alarm Rate	0.917	0.154
Accuracy	0.843	0.073

In the network, applications like TELNET, SSH use only key press event so if this is the case only available features that we can use in our detector is $DD.key1.key2$. If we limit the algorithm to use only these features, we obtain results provided in Table 2.8.

Table 2.8. Euclidean with n=10 DD features only.

Euclidean Detector	**Mean**	**Std**
Equal-Error Rate	0.187	0.087
Zero-Miss False-Alarm Rate	0.968	0.053
Accuracy	0.813	0.087

2.7. Normalized Euclidean Method

This method was suggested by Bleha [32] in 1990 and called as Normalized Minimum Distance Classifier. Normalized Euclidean model consists in calculating the same Euclidean distance; however it will be normalized by dividing the same with the product of the norms of the two vectors. S, the anomaly score is calculated as

$$S = \frac{\sum_{i=1}^{n} (y_i - x_i)^2}{\|x\| \cdot \|y\|}, \tag{2.13}$$

where x is the center of training vectors and y is the test vector. We have also obtained same performance results of Euclidean Anomaly Detector suggested by S. Bleha, C. Slivinsky and B. Hussien [32] shown in Table 2.9.

Table 2.9. Normalized Euclidean with n=31 features.

Normalized Euclidean Detector	**Mean**	**Std**
Equal-Error Rate	0.215	0.119
Zero-Miss False-Alarm Rate	0.911	0.148
Accuracy	0.785	0.118

Using selective futures can improve the EER. Lets assume that P is $1 \times n$ feature selection row vector those each elements consist of 0 or 1 and defined as

$$P = \arg\min_{p} \left[S(p) = \sum_{i=(p==1)} \frac{(y_i - x_i)^2}{\|x\| \cdot \|y\|} \right], \tag{2.14}$$

and a solution to P gives us following results:

$$P = [1\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0]_{1\times 31}. \quad (2.15)$$

EER, Accuracy and Zero-Miss False-Alarm Rate parameters are improved over feature selection as shown in Table 2.10.

Table 2.10. Euclidean with feature selection n=16 features.

Normalized Euclidean Detector	**Mean**	**Std**
Equal-Error Rate	0.171	0.073
Zero-Miss False-Alarm Rate	0.893	0.165
Accuracy	0.829	0.073

As we can see in the Table 2.10, Accuracy improved 4.4% comparing to non-selective Normalized Euclidean Method. If we limit the algorithm to use only DD features, we obtain results provided in Table 2.11.

Table 2.11. Normalized Euclidean with n=10 DD features only.

Normalized Euclidean Detector	**Mean**	**Std**
Equal-Error Rate	0.219	0.116
Zero-Miss False-Alarm Rate	0.927	0.136
Accuracy	0.781	0.116

2.8. Manhattan Method

Manhattan principle is same like Euclidean [31] and is classic as well. However the actual distance measured here is the Manhattan (city block) distance but not the Euclidean distance. The anomaly score S is calculated as

$$S = \sum_{i=1}^{n} |y_i - x_i|, \quad (2.16)$$

where x is the center of training vectors and y is the test vector.

Table 2.12. Manhattan with n=31 features.

Manhattan Detector	**Mean**	**Std**
Equal-Error Rate	0.153	0.092
Zero-Miss False-Alarm Rate	0.843	0.242
Accuracy	0.847	0.092

Instead of using all features, selective features can improve the EER. Lets assume that P is $1 \times n$ feature selection row vector those each elements consist of 0 or 1 and it is defined as

$$P = \arg\min_{p} \left[S(p) = \sum_{i=(p==1)}^{n} |y_i - x_i| \right], \tag{2.17}$$

and a solution to P gives us following results:

$$P = [1\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0]_{1\times 31}. \tag{2.18}$$

EER, Accuracy and Zero-Miss False-Alarm Rate parameters are improved over feature selection as shown in Table 2.13.

Table 2.13. Manhattan with feature selection n=14 features.

Manhattan Detector	**Mean**	**Std**
Equal-Error Rate	0.131	0.065
Zero-Miss False-Alarm Rate	0.836	0.227
Accuracy	0.869	0.065

As we can see in the Table 2.13, with the selection of only 14 features accuracy improved 2.2% comparing to non-selective Manhattan Method. If we limit the algorithm to use only $DD.key1.key2$ features, we obtain results provided in Table 2.14.

Table 2.14. Manhattan with n=10 DD features only.

Manhattan Detector	**Mean**	**Std**
Equal-Error Rate	0.172	0.095
Zero-Miss False-Alarm Rate	0.911	0.154
Accuracy	0.829	0.095

2.9. Filtered Manhattan Method

Filtered Manhattan is a detector that employs the training data filtered off the outliers. It is suggested by Joyce and Gupta [33] in 1990. Any training data with more than three standard deviations will be filtered out. Important factor here is that filtering is not done column-wise but each elements of columns. Rest procedure is same like Manhattan. After filtering the outliers, the anomaly score S is calculated as it is shown in Equation (2.16).

Table 2.15. Filtered Manhattan with n=31 features.

Filtered Manhattan Detector	**Mean**	**Std**
Equal-Error Rate	0.136	0.083
Zero-Miss False-Alarm Rate	0.755	0.283
Accuracy	0.864	0.083

In order to improve the performance of the detector, we can apply same feature selection procedure as it is shown in Equation (2.17) and obtained P selection vector as

$$P = [1\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 1]_{1\times 31}. \tag{2.19}$$

EER, Accuracy and Zero-Miss False-Alarm Rate parameters are improved over feature selection as shown in Table 2.16.

Table 2.16. Filtered Manhattan with feature selection, n=17 features.

Filtered Manhattan Detector	**Mean**	**Std**
Equal-Error Rate	0.119	0.063
Zero-Miss False-Alarm Rate	0.728	0.262
Accuracy	0.881	0.063

As we can see in the Table 2.16, with the selection of only 14 features accuracy improved 1.2% comparing to non-selective Filtered Manhattan Method. If we limit the algorithm to use only $DD.key1.key2$ features, we obtain results provided in Table 2.17.

Table 2.17. Filtered Manhattan with n=10 DD features only.

Filtered Manhattan Detector	**Mean**	**Std**
Equal-Error Rate	0.154	0.089
Zero-Miss False-Alarm Rate	0.843	0.212
Accuracy	0.846	0.089

2.10. Scaled Manhattan Method

Scaled Manhattan detector which was suggested by Araújo [34] in 2004 employs calculation of the mean absolute deviation (MAD) of each feature as well as mean vectors calculated. The average absolute deviations from the training phase are computed here.

Assume that T is a sample count in training set and X is a training matrix which has a dimension T-by-n where n represents feature set cardinality. Anomaly score S is calculated as

$$S = \sum_{i=1}^{n} \frac{|y_i - x_i|}{a_i}, \tag{2.20}$$

where a_i is given as

$$a_i = \frac{1}{n} \sum_{j=1}^{T} \left| X_{j,i} - \mu_{X_j} \right|. \tag{2.21}$$

If we applied same feature selection method to improve the detector's performance,

$$P = \arg\min_{p} \left[S(p) = \sum_{i=(p==1)}^{n} \frac{|y_i - x_i|}{a_i} \right], \tag{2.22}$$

and solving the P selection vector yields

$$P = [1\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1]_{1\times 31}. \tag{2.23}$$

Table 2.18. Scaled Manhattan with n=31 features.

Scaled Manhattan Detector	**Mean**	**Std**
Equal-Error Rate	0.096	0.069
Zero-Miss False-Alarm Rate	0.601	0.337
Accuracy	0.904	0.069

Table 2.19. Scaled Manhattan with feature selection n=21 features.

Scaled Manhattan Detector	**Mean**	**Std**
Equal-Error Rate	0.090	0.070
Zero-Miss False-Alarm Rate	0.532	0.328
Accuracy	0.910	0.069

As we can see in the Table 2.19, with the selection of only 21 features out of 31, accuracy improved 0.06% comparing to non-selective Scaled Manhattan Method. If we limit the algorithm to use only $DD.key1.key2$ features, we obtain results provided in Table 2.20.

Table 2.20. Scaled Manhattan with n=10 DD features only.

Scaled Manhattan Detector	**Mean**	**Std**
Equal-Error Rate	0.151	0.079
Zero-Miss False-Alarm Rate	0.847	0.211
Accuracy	0.849	0.079

2.11. Mahalanobis Method

Mahalanobis is classic anomaly detection algorithm [31] and employs the same principles of Manhattan and Euclidean, however the distance measuring procedures are more and more complex. Co-variance matrix of the timing vectors is also computed along with the mean vector in this system. To distinguish the points closer or far away to the mean vector, it uses correlation between the timing feature samples. The anomaly score S is calculated as

$$S = (x - y)^T C^{-1} (x - y), \tag{2.24}$$

where x is the center of training vectors, y is the test vector and C is the covariance matrix of training vectors.

In practice, due to dependency between the timing features, C covariance matrix is high likely to be numerically singular or ill-conditioned therefore it is not invertable properly. To achieve this problem, two well known solutions available in the literature. Singular Value Decomposition (SVD) and QR factorization.

Assume that X is a training set matrix, Y is a test set matrix and QR factorization can be followed as

$$\mu_X \;=\; \left[\; \mu_{x_1} \quad \mu_{x_2} \quad \cdots \quad \mu_{x_n} \;\right], \tag{2.25}$$

$$M = \begin{bmatrix} \cdots & \mu_X & \cdots \\ \cdots & \mu_X & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \mu_X & \cdots \end{bmatrix}, \tag{2.26}$$

and QR decomposition is defined as

$$(X - M) = Q.R, \tag{2.27}$$

where Q is a orthogonal matrix, R is a upper triangular matrix after factorization and we can obtain a symmetric matrix by removing null rows and therefore it can be invertable. After inverting the R, if we define r as

$$r = R^{-1}(Y - M)^T, \tag{2.28}$$

then the anomaly score of the Mahalanobis detector can be calculated as

$$S = r.r^T. \tag{2.29}$$

Another approach is to use Singular Value Decomposition (SVD) to invert ill conditioned covariance matrix C by defining it as

$$C = U\Sigma V^T, \tag{2.30}$$

where covariance matrix C can be decomposed to U and V those are unitary matrices and Σ is a diagonal matrix. Null items or near null values of Σ-diagonal matrix which are smaller then ε, can be removed and it becomes easily invertable using

$$C^{-1} = (V\Sigma^{-1})U^T. \tag{2.31}$$

Therefore the anomaly score S can be calculated using SVD as

$$S = (X - Y)^T(V\Sigma^{-1})U^T(X - Y). \tag{2.32}$$

Both approaches yield same results as it is provided in K. S. Killourhy and R. A. Maxion's study [16]. Defining feature selection vector P as

$$P = \arg\min_p \left[S(p) = S \cdots, \forall i = (p == 1)\right], \tag{2.33}$$

yields following solution,

$$P = [1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0]_{1\times 31}. \tag{2.34}$$

As we can see in the Table 2.22, with the selection of only 16 features, accuracy is improved only amount of 0.007% comparing to non-selective Mahalanobis Method. If we limit the algorithm to use only $DD.key1.key2$ features, we obtain results provided in Table 2.23.

Table 2.21. Mahalanobis with n=31 features.

Mahalanobis Detector	**Mean**	**Std**
Equal-Error Rate	0.110	0.065
Zero-Miss False-Alarm Rate	0.482	0.273
Accuracy	0.890	0.064

Table 2.22. Mahalanobis with feature selection n=16 features.

Mahalanobis Detector	**Mean**	**Std**
Equal-Error Rate	0.103	0.062
Zero-Miss False-Alarm Rate	0.481	0.290
Accuracy	0.897	0.062

Table 2.23. Mahalanobis with n=10 DD features only.

Mahalanobis Detector	**Mean**	**Std**
Equal-Error Rate	0.153	0.067
Zero-Miss False-Alarm Rate	0.703	0.276
Accuracy	0.847	0.068

2.12. Normalized Mahalanobis Method

Normalized Mahalanobis [32] employs normalization of the Mahalanobis distance using the same divisor as employed in other normalized detection algorithm models. The method also known as Normalized Bayes Classifier [32]. The anomaly score S can

be calculated as

$$S = \frac{(x-y)^T.C^{-1}.(x-y)}{\|x\| \cdot \|y\|}, \tag{2.35}$$

where x is the center of training vectors and y is the test vector. With QR factorization method, it can be simplified as

$$S = \frac{r.r^T}{\|x\| \cdot \|y\|}, \tag{2.36}$$

where r is defined in Equation (2.28) previously. We have also obtained same performance results of Normalized Mahalanobis Detector suggested by S. Bleha and C. Slivinsky and B. Hussien [32] shown in Table 2.24. The feature selection procedure can be applied by solving P,

$$P = \arg\min_{p} \left[S(p) = S \cdots, \forall i = (p == 1)\right], \tag{2.37}$$

and we obtained P as an optimum solution as

$$P = [0\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0]_{1\times 31}. \tag{2.38}$$

Table 2.24. Normalized Mahalanobis with n=31 features.

Normalized Mahalanobis Detector	**Mean**	**Std**
Equal-Error Rate	0.162	0.100
Zero-Miss False-Alarm Rate	0.648	0.295
Accuracy	0.837	0.100

As per the results provided in the Table 2.24 and Table 2.25, with the selection of only 16 features, accuracy is improved 3.2% comparing to non-selective Normalized Mahalanobis Method. If we limit the algorithm to use only $DD.key1.key2$ features, we obtain results provided in Table 2.26.

Table 2.25. Normalized Mahalanobis with feature selection n=16 features.

Normalized Mahalanobis Detector	**Mean**	**Std**
Equal-Error Rate	0.131	0.078
Zero-Miss False-Alarm Rate	0.614	0.275
Accuracy	0.869	0.078

Table 2.26. Normalized Mahalanobis with n=10 DD features only.

Normalized Mahalanobis Detector	**Mean**	**Std**
Equal-Error Rate	0.168	0.070
Zero-Miss False-Alarm Rate	0.778	0.262
Accuracy	0.832	0.069

2.13. Nearest Neighbor with Mahalanobis Method

The Nearest Neighbor with Mahalanobis Method, first suggested by Cho Sung-zoon, Han Chigeun, Hee Dae and Kim Hyung-Il [35], calculates the covariance matrix of training vectors and the anomaly score is then determined by minimum value of the difference between the test vector and the training vector after multiplying with the inverse of the covariance matrix. As before, feature selection procedure can be applied by solving Equation (2.37), and we obtained P as an optimum solution:

$$P = [1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 0]_{1\times 31}. \tag{2.39}$$

With the $DD.key1.key2$ features, we obtain results provided in Table 2.29.

Table 2.27. Nearest Neighbor Mahalanobis with n=31 features.

Nearest Neighbor with Mahalanobis Detector	**Mean**	**Std**
Equal-Error Rate	0.100	0.064
Zero-Miss False-Alarm Rate	0.468	0.272
Accuracy	0.900	0.064

Table 2.28. Nearest Neighbor Mahalanobis with feature selection n=15.

Nearest Neighbor with Mahalanobis Detector	**Mean**	**Std**
Equal-Error Rate	0.092	0.059
Zero-Miss False-Alarm Rate	0.435	0.248
Accuracy	0.908	0.059

Table 2.29. Nearest Neighbor Mahalanobis with n=10 DD features only.

Nearest Neighbor with Mahalanobis Detector	**Mean**	**Std**
Equal-Error Rate	0.121	0.064
Zero-Miss False-Alarm Rate	0.523	0.248
Accuracy	0.878	0.064

2.14. Standard Neural Network Method

Standard Neural Network Method works on a feed-forward neural network trained with the back propagation algorithm. The method proposed by S. Haider, A. Abbas and A. K. Zaidi [36]. The process consists in building with about p input nodes and one output node as well. Also consists of $2n/3$ hidden nodes. If s is the output of the network, the anomaly S, is computed as $1-s$ in this model, when s is 1 the test vector is similar to the original and if not it is considered dissimilar.

As it is explained in [36], all weights in hidden layer and output layer are set to 0.1 at initialization. And all bias values are assigned randomly. The network trained to yield 1.0 as output for each training vector. Learning parameter chosen as 0.0001 with 500 epochs. And we used Scaled Conjugate Gradient Back Propagation Algorithm [37] for training.

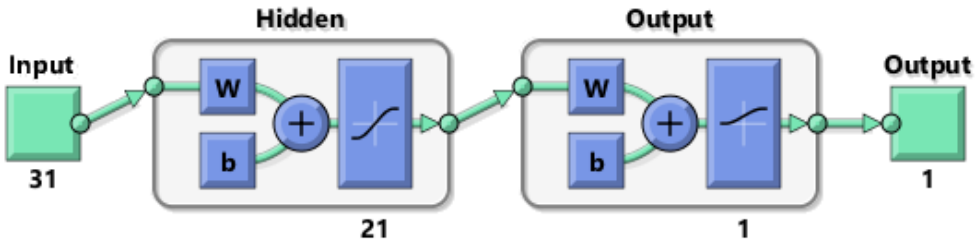

Figure 2.4. Feed forward NN trained with the back-propagation, generated using MATLAB [38].

Table 2.30. Standard NN with n=31 features.

Standard Neural Network Detector	**Mean**	**Std**
Equal-Error Rate	0.154	0.119
Zero-Miss False-Alarm Rate	0.776	0.292
Accuracy	0.846	0.118

Feature selection procedure can be applied by solving P defined as

$$P = \arg\min_{p} \left[S(p) = S \cdots, \forall i = (p == 1) \right], \tag{2.40}$$

and we obtained P as an optimum solution,

$$P = [0\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0]_{1\times 31}. \tag{2.41}$$

If we limit the algorithm to use only $DD.key1.key2$ features, we obtain results provided in Table 2.32.

Table 2.31. Standard NN with feature selection n=23 features.

Standard Neural Network Detector	**Mean**	**Std**
Equal-Error Rate	0.148	0.130
Zero-Miss False-Alarm Rate	0.718	0.342
Accuracy	0.852	0.129

Table 2.32. Standard NN with n=10 DD features only.

Standard Neural Network Detector	**Mean**	**Std**
Equal-Error Rate	0.201	0.154
Zero-Miss False-Alarm Rate	0.848	0.264
Accuracy	0.799	0.156

2.15. Multilayer Perceptron Neural Network Method

In 2023, Pelin Damla Ateş, Çağatay Ateş, Mutlu Koca and Emin Anarım showed that they could detect unknown types of attacks on the network using autoencoders and extreme value theory [39]. The neural network used in this study is an auto associative, multilayer perceptron. The process consists of feed-forward neural network trained with back propagating algorithm as well. Euclidean distance is the criteria and employed as anomaly score in this case. As it is described by Cho Sungzoon, Han Chigeun, Hee Dae and Kim Hyung-Il [35], n feature vector and same amount of hidden nodes used and the network was trained to generate same training vector on its output therefore it is called as auto associative. As a anomaly score, standard Euclidean distance is used between test set and output of the neural network.

Training parameters used are that learning rate is 0.0001, momentum is 0.0003 and for 500 epochs.

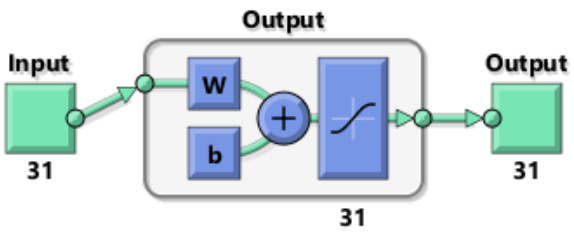

Figure 2.5. Auto associative NN trained with the back-propagation, generated using MATLAB [38].

Table 2.33. Auto Associative Neural NN with n=31 features.

Auto Associative Neural Network Detector	**Mean**	**Std**
Equal-Error Rate	0.147	0.082
Zero-Miss False-Alarm Rate	0.751	0.266
Accuracy	0.854	0.082

If we limit the algorithm to use only $DD.key1.key2$ features, we obtain results provided in Table 2.34.

Table 2.34. Auto Associative NN with n=10 DD features only.

Auto Associative Neural Network Detector	**Mean**	**Std**
Equal-Error Rate	0.201	0.092
Zero-Miss False-Alarm Rate	0.909	0.128
Accuracy	0.799	0.092

2.16. z-score Method

Outlier counting suggested by S. Haider, A. Abbas and A. K. Zaidi [36] consists of counting the z-scores and comparing the same with the threshold. We calculate standard deviation and mean value of each features. z-score calculated as

$$z_i = \frac{|x_i - \mu_i|}{\sigma_i}, \tag{2.42}$$

where μ_i is mean value of the i-th feature. And σ_i is standard deviation of the i-th feature. Anomaly score S is calculated as

$$S = \sum_{i=1}^{n} (z_i > 1.96). \tag{2.43}$$

Table 2.35. z-score with n=31 features.

z-score Detector	**Mean**	**Std**
Equal-Error Rate	0.102	0.077
Zero-Miss False-Alarm Rate	0.782	0.306
Accuracy	0.895	0.079

Threshold value picked as 1.96 in Equation (2.43) is provided directly in S. Haider, A. Abbas and A. K. Zaidi's study [36]. But it is not mentioned that how to reach this value or whether it is optimum. If we define η_x as the best threshold value, it can be expressed as

$$\eta_x = \arg\min_x \left[S(x) = \sum_{i=1}^{n} (z_i > x) \right]. \tag{2.44}$$

By optimizing η_x, we obtained 1.6% improvement in the accuracy and Zero-Miss False-Alarm Rate dropped from 0.782 to 0.618 which was significant.

Table 2.36. z-score with n=31 features and $\eta_x = 1.449$.

z-score Detector	**Mean**	**Std**
Equal-Error Rate	0.087	0.064
Zero-Miss False-Alarm Rate	0.618	0.302
Accuracy	0.911	0.065

If we limit the algorithm to use only $DD.key1.key2$ features, we obtain results provided in Table 2.37.

Table 2.37. z-score with n=10 DD features only.

z-score Detector	**Mean**	**Std**
Equal-Error Rate	0.189	0.106
Zero-Miss False-Alarm Rate	0.982	0.127
Accuracy	0.808	0.103

2.17. One Class Support Vector Machines Method

Support Vector Machines (SVM) employs support vector machine algorithm which usually finds a separator between two different classes provided in a data-set. But for anomaly-detection, another usage is suggested [40, 41] that one-class or unsupervised SVM looks for a best separator between the projection of the data-set onto a higher dimension plane and the origin. The negated distance of the linear separator forms the anomaly score in this case. The method proposed by Yu and Cho [28].

Table 2.38. One Class SVM with n=31 features.

One Class Support Vector Machines Detector	**Mean**	**Std**
Equal-Error Rate	0.100	0.064
Zero-Miss False-Alarm Rate	0.486	0.312
Accuracy	0.900	0.064

Feature selection procedure can be applied by minimizing anomaly score, we obtained P as a optimum solution as

$$P = [1\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 0]_{1\times 31}. \tag{2.45}$$

Table 2.39. One Class SVM with feature selection n=21 features.

One Class Support Vector Machines Detector	**Mean**	**Std**
Equal-Error Rate	0.095	0.066
Zero-Miss False-Alarm Rate	0.495	0.321
Accuracy	0.905	0.066

If we limit the algorithm to use only $DD.key1.key2$ features, we obtain results provided in Table 2.40.

Table 2.40. One Class SVM with n=10 DD features only.

One Class Support Vector Machines Detector	**Mean**	**Std**
Equal-Error Rate	0.138	0.065
Zero-Miss False-Alarm Rate	0.733	0.291
Accuracy	0.862	0.064

2.18. k-means Method

k-means identifies the clusters in the training vectors and evaluates whether the test vector has any cluster similar to this. As it is suggested by by Kang Pilsung, Hwang Seong-seob and Cho Sungzoon [19], on the training phase, k-means algorithm ran to find k=3 centroids. The anomaly score is computed as the Euclidean distance between the test vector and the nearest of these centroids.

Table 2.41. k-means with n=31 features.

k-means Detector	**Mean**	**Std**
Equal-Error Rate	0.154	0.070
Zero-Miss False-Alarm Rate	0.663	0.282
Accuracy	0.846	0.070

Feature selection procedure can be applied by minimizing anomaly score, we obtained P as an optimum solution as

$$P = [1\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1]_{1\times 31}. \tag{2.46}$$

Table 2.42. k-means with feature selection n=21 features.

k-means Detector	**Mean**	**Std**
Equal-Error Rate	0.134	0.058
Zero-Miss False-Alarm Rate	0.692	0.233
Accuracy	0.866	0.058

If we limit the algorithm to use only $DD.key1.key2$ features, we obtain results provided in Table 2.43.

Table 2.43. k-means with n=10 DD features only.

k-means Detector	**Mean**	**Std**
Equal-Error Rate	0.156	0.070
Zero-Miss False-Alarm Rate	0.725	0.271
Accuracy	0.843	0.070

2.19. Nearest Neighbor with Principle Square Root Method

This method suggested by Y. Zhong, Y. Deng and A. K. Jain [42] in 2012. They have combined stability of Manhattan Distance and flexibility of Mahalanobis Distance by using principle square root (PSR) of inverted covariance matrix C of training data-set by defining,

$$\Phi = C^{-\frac{1}{2}}, \tag{2.47}$$

where Φ is the inverse of the principle square root of the covariance matrix such that

$$\Phi^T \cdot \Phi = C^{-1}, \tag{2.48}$$

satisfies. We will apply correction on each samples before measuring the distances and it will be done based on linear operator as

$$x' = \Phi x, \tag{2.49}$$

where x' is a corrected sample. And the anomaly score S calculated after the correction as

$$\begin{aligned} S &= \|x - y\|' \\ &= \|x' - y'\|_1 \\ &= \left\| C^{-\frac{1}{2}}(x - y) \right\|_1. \end{aligned} \tag{2.50}$$

Table 2.44. Nearest Neighbor with PSR with n=31 features.

Nearest Neighbor with PSR Detector	**Mean**	**Std**
Equal-Error Rate	0.090	0.060
Zero-Miss False-Alarm Rate	0.442	0.275
Accuracy	0.910	0.060

Table 2.45. Nearest Neighbor with PSR with n=10 DD features only.

Nearest Neighbor with PSR Detector	**Mean**	**Std**
Equal-Error Rate	0.115	0.062
Zero-Miss False-Alarm Rate	0.518	0.264
Accuracy	0.885	0.062

3. USER AUTHENTICATION USING SIAMESE NETWORKS

3.1. Introduction

In contemporary times, many institutions resort to information technologies to efficiently continue their operations and to easily manage the services they provide to their customers. A multi-employee firm has numerous departments, each with individuals possessing different usage rights, competencies, and privileges. Where there is a human factor, security vulnerabilities are inevitable. To mitigate these vulnerabilities, Privileged Access Management (PAM) systems are being developed. These systems aim to prevent unauthorized access to relevant services by monitoring the usage of network, programs, disks, memory, computer processors, mice, and keyboards by authorized personnel, and detecting deviations in these usages. This study focuses on examining users' keyboard usage behavior, which is a subcomponent of the desired PAM system.

Studies on users' keyboard usage behavior began in the late 1970s. In 1977, G. Forsen [13] investigated whether the user's identity could be verified using various techniques based on the speed of typing their own name. K. S. Killourhy [16], in his 2009 study, uniquely focused on detecting the user during password entry. L. Ali [23], with his 2017 study on the Partially Observable Hidden Markov Model, significantly improved performance criteria in identifying users during fixed-length password entries. However, he still could not achieve the less than 1% false alarm rate and the 0.001% zero-miss rate required by the European Union standards [12] for entry control systems. Currently, no known anomaly detector meets these rates. Both single-class and dual-class detectors are intensely evaluated on mobile devices and desktop computers. E. Davarcı [43], in his 2022 study, demonstrated that dual-class detectors are more successful than single-class detectors. In 2019, M. Yıldırım [44] showed that similar verification could also be done through computer mouse movements.

In this study, we will examine user's keyboard usage behavior, the features extracted from the user's timing data during a short-fixed text entries will be converted into images and fed into a Gated Recurrent Unit (GRU) in a new Siamese Network topology will be proposed [45]. The results will be examined within the scope of group leakage, and a new leakage-free method will be proposed and evaluated.

3.2. Pre-processing and Feature Extraction

Normalization is crucial in the input layer to prevent the activation functions of the neurons from saturating. Therefore, each value in the dataset was divided by 255 before any transformation.

In the study conducted by Medvedev Viktor, Budžys Arnoldas and Kurasova Olga in 2023 [46], the Recurrence Plots (RP) method, which yielded the best result among Gramian Angular Summation Field (GASF), Gramian Angular Difference Field (GADF), Markov Transition Fields (MTF), and Recurrence Plots (RP) methods with a 0.078 Equal Error Rate (EER), was used as the basis and applied to the input vectors. The purpose of this transformation is to extract meaningful features from the 31-dimensional feature vector to assist the deep learning algorithm and facilitate easier classification of the data. This technique transforms a time series into two-dimensional images. This image is calculated by measuring the similarity of each point in the input vector to other points in the same vector. After this transformation, the 31-dimensional feature vector becomes a 31x31 dimensional image, which is then flattened and entered into the system as a single 961-dimensional vector.

3.3. System Description

It is believed that the keystroke dynamics of each user are unique, similar to a fingerprint. The relationship between successive key press and release events, such as typing speeds and pauses, are among the defining characteristics that enable authentication and distinguish one user from another. Recurrent Neural Networks have been

shown to be one of the best algorithms for handling temporally varying data [47]. In their study, Alejandro Acien, Aythami Morales, Ruben Vera-Rodriguez, Julian Fierrez and John V. Monaco [48] tested a similar structure on texts of varying lengths and succeeded in identifying users with an EER of 8.6% in sequences of 30 keystrokes. As the sequence of pressed keys increases, the system's ability to distinguish the user improves significantly. In this study, since fixed-length short texts are used, the sequence length is 1, and the number of features is 31. In the proposed Siamese twin structure shown in Figure 3.1, the first 128 samples from the user's training set are compared with a single data vector of the tested user, thus embedding the averaging process into the sequential input of the system, allowing the proposed deep learning network to make more accurate predictions.

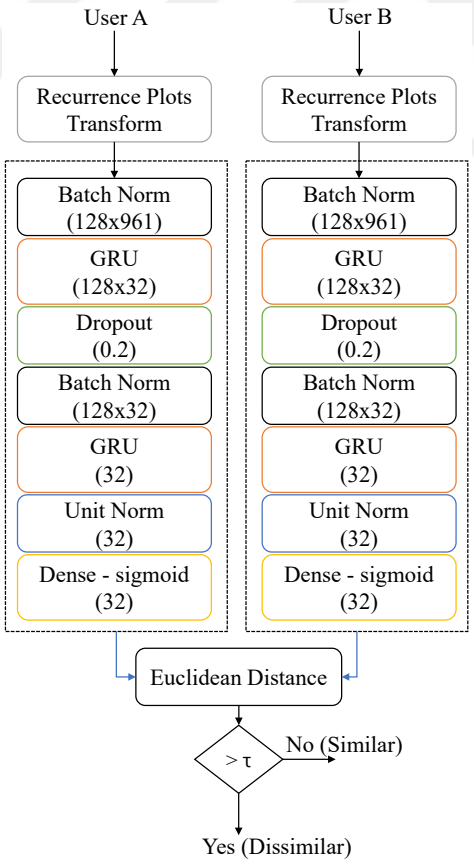

Figure 3.1. Siamese GRU network architecture suggested.

The GRU structure was preferred because it contains fewer training parameters compared to the Long Short-Term Memory (LSTM) structure and shows almost the same performance. Each GRU structure has 32 cells, a *tanh* activation function, and a recurrent dropout rate of 0.2. To prevent over-fitting, an additional dropout layer with a rate of 0.2 was added after the GRU structure. The final Dense layer was added to stabilize the validation loss function. The Siamese twin network constructed in this way has a total of 104,898 trainable parameters, which are shared between the right and left branches of the network [49].

3.4. Training Phase

The CMU dataset consists of a total of 51 users, with 400 samples for each user, and 31 temporal features of keystroke durations in each sample. While training the Siamese GRU network architecture created for each user, 200 out of 400 samples belonging to the owner of the network were used for training, and the remaining 200 were used for testing. In none of the studies conducted within the scope of this article were the training set and the test set mixed.

20% of the 200 data points used as the training set were reserved for validation and early stopping during the training phase. Additionally, in the normal training set, values that differed from the median by more than three times the mean absolute deviation were considered outliers and not used. For training the proposed network architecture, examples describing anomaly conditions, in addition to the positive labels belonging to the owner of the network, are required. This requirement was met by obtaining samples from the training sets of other users in the CMU dataset. Results were also obtained using 50 samples from five other users, 25 samples from 10 users, 10 samples from 25 users, and five samples from 50 users, along with 250 randomly generated samples uniformly distributed over an n-dimensional spherical surface, thus maintaining a balance between positive and negative labels during the training phase.

The contrastive loss function proposed by Hadsell Raia, Chopra Sumit and LeCun Yann in 2006 [50] was used as the loss function. Given input vector pairs x_i and x_j containing the temporal data of the user's keystrokes, $f(.)$ represents the network output shown in Figure 3.1, and L_{ij} is 0 for pairs x_i and x_j from the same class and 1 for different classes. The contrastive loss function $\mathcal{L}_{\text{CF}}$ is defined as follows:

$$\mathcal{L}_{\text{CF}} = (1 - L_{ij})\frac{\|f(x_i) - f(x_j)\|_2^2}{2} + L_{ij}\frac{\max^2\left\{0, \varepsilon - \|f(x_i) - f(x_j)\|_2\right\}}{2}. \tag{3.1}$$

During training, the margin value ε was set to 0.1 as demonstrated in Medvedev's study [46]. The best results were obtained 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 operations were conducted using an AMD Ryzen Threadripper 3990X 64-Core Processor, 256GB RAM, and Linux Kernel v4.19, using Keras-TensorFlow v2.15.0. For reproducibility, the seed values of the pseudo-random number generators were set to a fixed value of 7.

3.5. Group Leakage

In the study by Ayotte Blaine, Banavar Mahesh K, Hou Daqing and Schuckers Stephanie [51], it was noted that when group leakage is ignored, the performance metrics of user authentication procedures using keystroke dynamics and deep learning methods yield lower-than-expected results in real-world applications. The main reason for this is treating open-set identification problems as if they were closed-set. In anomaly detection problems, while there are many examples showing non-anomalous conditions that can be used during training, there are very few or no examples of anomalous conditions.

The CMU dataset contains samples from 51 users, and including one or some of the remaining 50 users as impostors in the training data of the system leads to leakage. In our study, we observed the impact on performance of increasing amounts of leakage, where the impostors included in the training phase were from 5, 10, 25, and 50 other users, and compared the results obtained with randomly generated impostor data on the surface of an n-dimensional sphere, which contains no leakage.

3.6. Minimum Enclosing n-Sphere Method

As one of our contributions, the proposed method generates the impostor data required during the training phase from the user's own normal data used during the training phase, without using any information belonging to other users in the dataset.

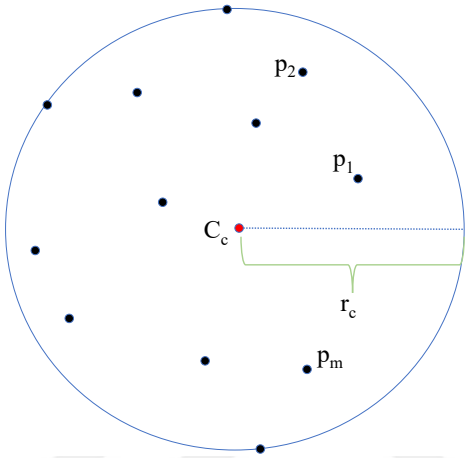

Figure 3.2. Minimum enclosing n-sphere.

To achieve this, first, the equation of the n-dimensional sphere with the smallest radius that encompasses all normal samples in the training set of the user to be modeled was solved, as shown in Figure 3.2. Let p_i represent the n-dimensional and m samples in the user's training set, r_c the radius, and C_c the center of the sphere to be found:

$$\min_{r_c \geq \left\| p_i - C_c \right\|_2} {}_{(r_c, C_c)} \left\{ r_c \right\}, \tag{3.2}$$

where

$$p_i, C_c \in R^n, r_c \in R, i = 1 \ldots m. \tag{3.3}$$

Elzinga [52] demonstrated that a unique solution exists for r_c, C_c. Practical results were easily obtained using the Gurobi Optimizer [53]. On this surface, with its calculated center and radius, 250 uniformly randomly generated points, as shown by G. Marsaglia [54], were labeled as impostor data and used for training the network.

3.7. Results

The mean and standard deviation values of EER and performance rankings of the implemented detectors are shown in Table 3.1. In the GRU (50 impostors) structure, where group leakage is at its maximum, the average system accuracy was observed to be 93.0%, yielding results close to the 92.2% obtained in the study by Medvedev Viktor, Budžys Arnoldas and Kurasova Olga with a similar level of leakage [46]. Acien Alejandro, Morales Aythami, Monaco John V, Vera-Rodriguez Ruben and Fierrez Julian [55], in their study without group leakage, achieved an EER of 10.7% using a comparative loss function in a structure involving thirty keystrokes, whereas the GRU (n-Sphere) method without leakage achieved an EER of 0.153 using only 10 keystrokes.

Table 3.1. Performances of suggested structures.

Detectors	**Results**	
	Mean EER (std)	**Mean ZMFAR (std)**
1. GRU (50 impostors)	0.070 (0.060)	0.273 (0.251)
2. GRU (25 impostors)	0.098 (0.068)	0.414 (0.291)
3. GRU (5 impostors)	0.125 (0.094)	0.467 (0.309)
4. GRU (10 impostors)	0.137 (0.104)	0.444 (0.293)
5. GRU (n-Sphere)	0.153 (0.088)	0.567 (0.260)

In our study, we proposed a new GRU-based model using a Siamese twin topology that eliminates the need for averaging. We shared the performance results in the presence of group leakage. By proposing a method that completely eliminates leakage, we compared the performance of the same model with and without leakage. Future studies may focus on the widely used triplet loss function in the literature, and the proposed method for creating anomaly-labeled data without causing leakage can be extended using the maximum enclosing sphere method.

4. EMPIRICAL STUDY

4.1. Introduction

We have studied many classifiers using CMU Keystroke data-set and obtained their performances under different feature selections and various parameter optimization. But we are missing one important factor that the aimed PAM project shall run on a cloud system therefore all its services shall be provided in a network environment rather than a direct physical access.

Keystroke Data-set provided by CMU captured locally through a notebook with a keyboard connected directly to it. But a cloud system usually stands behind a network topology and various protocols such as UDP, TCP ran to access its services.

In this part of the thesis, we will design a real time embedded platform to produce events exactly at the same time stamps with one provided in CMU's Keystroke data-set and then send them through UDP, TCP and some SSL/TLS enabled protocols to the cloud system. Events will be captured on the cloud system and anomaly classifiers performances investigated in the first part will be compared again with existence of the cloud system.

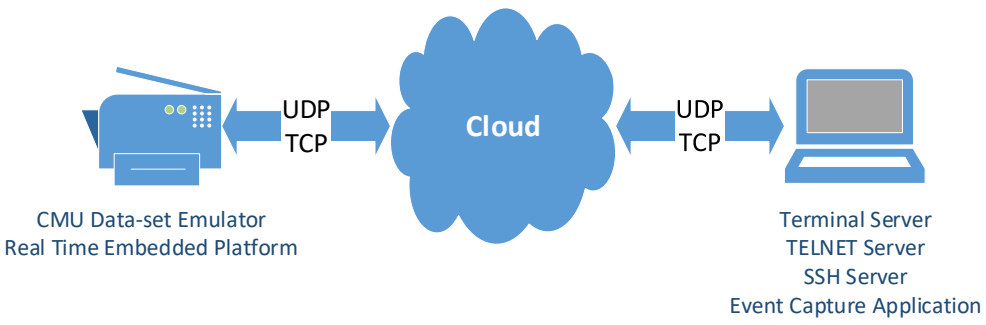

Figure 4.1. Realtime keystroke emulator.

4.2. Cloud Usage

Year after year the total number of people using internet services across the globe is getting increased. As of 2011 the total number of the same is 2 billion and it is expected to increase by 33% by 2016. However, this estimation may be somewhat overestimation, due to the reason that it is possible that one user might be using numerous devices and hence might be misleading the usage statistics. It is also expected that the number of internet connected devices may increase drastically in the coming years. By 2011, the total number of fixed and mobile networked devices as well as machine to machine connections is about 7 billion as per one realistic estimation. It is expected that this number will go up to as high as 20 billion by 2016. The number of users connected devices more specifically broad band devices will affect the volume of the traffic. This will result in non-productive consumption of the band capacity of the network connection. Further it will also consume the capacity on the multiple devices at the same time. Further each of the broad band devices will enable diverse applications, which cumulatively will consume higher and higher traffic generation. The total annual traffic can be about 372 Exa bytes as per latest estimations of 2011, and it is more likely to shoot up to 1.3 Zetta bytes by 2016 [56]. Such a huge traffic will call for challenges which do not exist before on the traffic and performance requirements on the communication protocols.

Local area network is very common type of network and it is very common observation. LANs are inevitable in almost all type of business organizations at present. The power and capacity consumption of LANs have grown tremendously in accordance with the changes in the power and the number of the computing devices. LANs as of now are proliferated in accordance with the standards which are developed on international framework and are being applied uniformly across all the networks. As of now Ethernet is one of the very dominant LAN architectures. However, the enterprise managers still can use both the wireless and wired LANs can be combined to make transmission rates from 100Mbps to 100Gbps speed rates. It is highly complicated and challenging to do interconnection and management of the diverse collection

of the LAN and computing devices in the current business network. In accordance with the changed business and operational requirements, communication is now part of the business and it is not confined to one particular office set-up or one specific LAN. It is a necessity of the entire business network to support voice, data, image and video traffic requirements. LAN Switches have improved their technology greatly. Also, diverse communication technologies developed in the recent years contributed to the great increase of the local area network transmission rates and capacities. Further the integration of communication devices is much improved at present. Further the integration has enabled to deal with voice, data, image and video all at the same time. A typical recent advancement in this direction can mean integration of a specific memo or report with an accompanies voice commentary, presentation graphics as well a video introduction. Further there can be demonstration, summary included automatically within the report. Enterprise networks need interconnections and the need for integration (interconnection) has increased tremendously in the recent years with the increase in the dispersion of the location of the enterprise networks. Some of the immediate consequences of such development include the necessity of the business organizations to keep in wide area network transmission and switching capabilities. Some of the factors that contributed to realize these developments include increase in the capacity of the fiber optic technologies, improvement in the wireless transmission services. These developments have actually working for meeting the communication needs of the recent business enterprises. The transmission links capacities and the switching systems performance to meet the ever-changing high-volume transmission capacities is a big challenge, with the current state of art and technology these challenges are not yet conquered fully, the deficiency still remains as it is. Enterprise networking is a versatile tool for organizations in current years. They can use this opportunity to improve the productivity of the organizations and also can work for slashing the costs. It is possible that if the business managers do have the necessary knowledge of these technologies and usage of the same, it is possible for them to work with these data communication equipment vendors more and service providers more effectively and this in turn can work on to improve the company's competitive position accordingly. The following part of the write-up works on to present the technologies like UDP protocols, TCP

protocols and Nagle algorithm.

4.3. Brief about UDP

UDP is also termed as user data-gram protocol. It is just an alternative protocol for TCP (Transmission control protocol). Sometime UDP in combination with IP (Internet Protocol) termed as UDP/IP. The similarities between UDP and TCP are contained in the procedures they do employ, like TCP, UDP also employ the IP for getting the Data unit (Data-gram) from one computer to the another. The following part of the discussion briefs the history of UDP and its development to the current state of art.

TCP was mainly responsible for provision for the network-layer connectivity, connection establishment and further is also responsible for the flow control and reliability aspects of the communication performance. However, during the further actions of improving the protocol effectiveness, reliability function is tried to get removed off. This is due to the fact that the reliability is considered to be an overhead and further it is not actually required by some applications and some applications do not need the same. UDP has become the solution to this issue. The original protocol is now divided into three elements, IP, TCP and UDP. IP is responsible for internet working; Reliability functions are now handled by TCP. UDP is simply a protocol that resides in the transport layer of the OSI model that can be seen in Figure 4.2. The original definition of UDP can be tracked to RFC 768 and UDP in 1980.

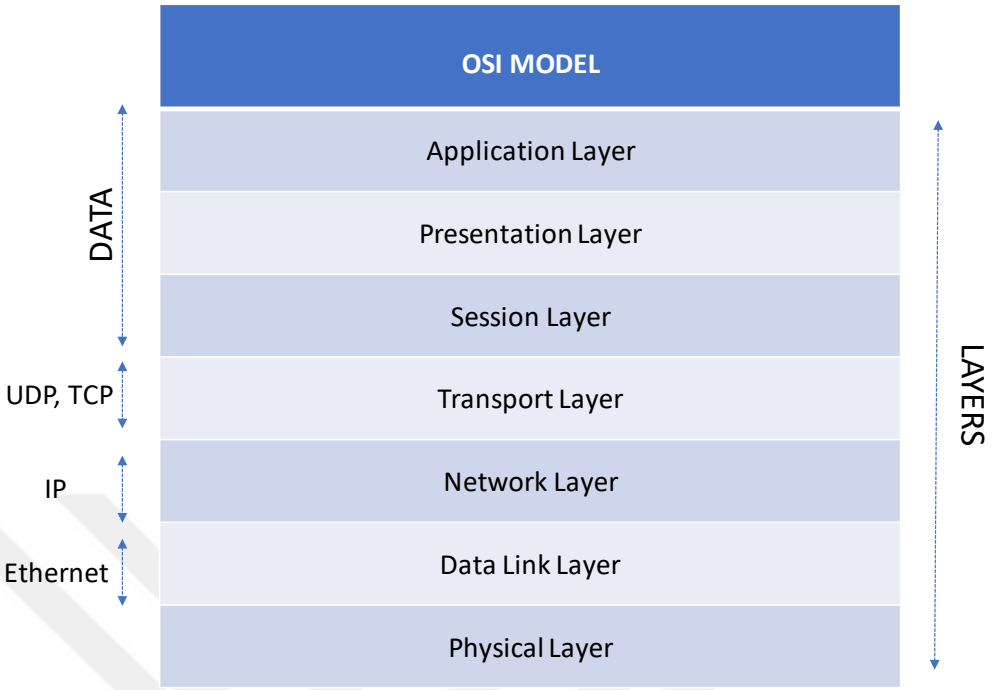

Figure 4.2. OSI model.

UDP is much simpler and lighter when compared with TCP. The most important characteristics of UDP that makes it distinct from TCP is the fact that UDP is not as complex as TCP. As UDP does not have the need to support the reliability as well as it has no need to support the acknowledgments, the delivery model is going to be more effort base. The most important objective of the protocol is to provide an interface between IP and the application process. The process corresponding to this is done in UDP by port numbers which is not available in the IP layer. The overall function of it contained is to take up the application layer data and pass the same to IP for transmission after packing the same into simplified message format.

UDP does not have any guarantee for message delivery to the upper layer of the protocols. That is why sometimes UDP is also referred to as Unreliable Data-gram protocol [57]. If there are any functions of the reliability to be handled, then obviously it is to be handed in the upper layers.

As a whole, the header of UDP contains 4 fields only as shown in UDP datagram [56]:

(i) The source port is the field that do identify the sending port which is meaningfully need to be assumed as the port that will be needed to reply to when there is necessity for the same. However, this field is an optional field and the value of the same can zero when it is not in usage.
(ii) Destination port, field is a mandatory field and it is a mandatory field in the complete UDP. The total length of the field is 16-bit and it will specify the total length of the bytes and the entire diagram as well. Header and the data in total make up the same. The minimum length of the field is 8 bytes. This is due to the reason that the length of the header will be 8 bytes.
(iii) Checksum is a 16-bit field. This pre-calculated value during the transmission, is used for validating the content of the header and the payload at the receiver side.
(iv) Data field is the field that actually contains the payload.

The checksum for UDP is defined using a pseudo header, which includes the IP source address, IP destination address, IP protocol field, and the UDP length field. The UDP stack runs on the destination device recalculates the pseudo header checksum and compares it with the transmitted checksum value in the UDP header [56].

UDP operations, ports, functions, and implementations are critical aspects to consider. The operations contained in UDP transmission can be summarized in three steps:

(i) Higher Layer Data Transfer: The application passes the data to the UDP stack.
(ii) UDP Message Encapsulation: The stacks adds the header to the data passed by the application. The UDP message headers include the source port and destination port, and the checksum value calculated.
(iii) Transfer Message Function: The UDP packed is transferred to the IP layer.

Upon reception, the stack (usually the kernel of an operating system) reverses this procedure and deliver the message to the application. Some properties of UDP include:

(i) UDP does not guarantee message delivery and does not provide acknowledgements for received data.
(ii) UDP does not establish a connection. It simply packs and sends it, therefore it is a connection-less protocol.
(iii) UDP does not guarantee the order of data arrival.
(iv) UDP does not detect nor re-transmit lost messages.
(v) UDP does not have mechanisms for managing data flow or handling congestion.
(vi) UDP provides an optional checksum capability to detect errors in transmission and verify message delivery to the correct destination.

Applications of UDP are recommended in the following scenarios:

(i) When performance speed is prioritized over reliable delivery.
(ii) For short data exchanges where the order of data-gram reception is not critical.
(iii) When delivery acknowledgements are not needed.
(iv) For applications requiring multi-cast or broadcast messages.

4.4. Brief Overview of TCP

TCP and IP are fundamental components of the TCP/IP suite. According to RFC793, TCP addresses reliable and stream-oriented connections, thereby mitigating many of IP's shortcomings. TCP enhances IP services by implementing several key features:

(i) Streams: TCP represents data as streams of bytes, similar to simple files, concealing the network datagram nature. It includes mechanisms for out-of-band data flagged as special.

(ii) Reliable Delivery: TCP uses sequence numbers to coordinate transmitted and received data, arranging for re-transmission if any data is lost.

(iii) Network Delay Adjustment: TCP adjusts network delay by trying to maximize network throughput.

(iv) Flow Control: TCP manages buffers available in the stack by throttling traffic, preventing buffer overflow and periodically stopping fast senders to keep pace with slower receivers.

TCP operations are inherently full-duplex, meaning they operate independently in both directions. A TCP session can be seen as two independent byte streams moving in opposite directions, without any mechanism to associate forward and reverse data streams.

TCP counts the number of bytes in the stream and stores this information in a 32-bit sequence number. The TCP packet includes the starting sequence number of the data and the acknowledgment number of the last byte received. A sliding window protocol is introduced for flow control. Each TCP peer tracks its own sequence numbering as well as the other endpoint's. Control flags, such as URG, SYN, and FIN, manage connection states and ensure reliable delivery, occupying a single byte in the sequence number space.

Each TCP endpoint has buffers for data storage before transmission over the network. The window size field in each packet, set by TCP, indicates the buffer space available, preventing buffer overflow. When the window size is zero, the remote TCP stops sending data. The protocol does not support larger window sizes, which can be challenging in networks with high bandwidth-delay products. This issue is addressed by LFN (Large Fat Networks) and RFC 1072 [58].

After transmitting a TCP packet, the host waits for an acknowledgment. If an acknowledgment cannot be received for a specified time, the packet is assumed to be lost and re-transmitted. Round Trip Time (RTT) estimation is crucial for this process,

as it affects performance parameters. Accurate RTT estimation is vital for maintaining efficient TCP exchanges.

4.4.1. Nagle Algorithm

The Nagle algorithm described in RFC896 [59] and RFC1122 [60] invented by John Nagle. The purpose of it is to reduce network congestion caused by inefficiency of TCP packets transmitted when the payload size is small relatively to the TCP header. Nagle's algorithm started being used in the 1990s, and it is still actively part of modern TCP implementations as of today [61]. This algorithm addresses the issue of applications, such as remote login, sending streams of one-octet data segments. To prevent deadlocks, TCP SEND calls must be pushed either by the application or TCP, resulting in inefficient internet usage and congestion. The Nagle algorithm solves this problem by discouraging the sending of tiny segments and accumulating small messages into larger TCP packets before transmission.

The Nagle algorithm complements the Send Silly Window Syndrome (SWS) [62] avoidance algorithm, which prevents small segments from advancing by incrementing the window size with small steps. Nagling targets small payloads and combines them into larger packets by delaying the physical transmission, therefore eliminates the inefficiency caused by header length / data length ratio. RFC896 [59], published in 1984, provides the technical specifications for the Nagle algorithm. By reducing the transmission of small packets, the Nagle algorithm significantly saves bandwidth, though at the cost of added latency. For example, character dumping over a telnet [63] connection, sends a byte with each keystroke but in the physical layer we observe that 40 bytes for header including IP's, plus 1 byte for the data itself, 41 bytes in total transmitted for a single byte payload.

Nagling can reduce up to 95% of header information and 50% or more of actual data from the sender's keyboard. TCP socket programming options control the use of the Nagle algorithm, which is enabled by default in Windows, Linux, and Java

systems. For applications requiring faster responses, the Nagle algorithm may need to be disabled, as it was originally intended for lower-bandwidth applications. The algorithm's relevance has diminished in modern high-bandwidth network environments, limiting its current applications compared to its original use cases.

4.5. Real-time Keystroke Emulator

Windows or Linux as a operating system are known as Non Real Time Operating Systems. They schedule many task to be done simultaneously. Therefore they can not guarantee that the exact time the code is executed. It can suddenly jump to another task by suspending current task for a while and suspension time is non deterministic, that is the reason they are called as Non-Real Time OS. Recent Windows timer accuracy can be as low as about 1 ms [64] after some kernel-mode development. Even that, it wouldn't be sufficient to emulate CMU's data-set event timings that is provided in a precision of 100 µs.

Designed Embedded Platform occupied with all necessary hardware to overcome all timing problems required for CMU's timings. To obtain a deterministic event generation, we didn't use any operating system, such as applications are called bare-metal application development.

The processor we picked has MIPS core [65] and runs at 250 MHz. The processor surrounded with Hardware peripherals such that a Real Time Clock (RTC) to be able to log events with a accurate time-stamps, 10 MHz temperature compensated crystal oscillator (TXCO) as a timer base, 100 Mbps Ethernet, one A-typed and one B-typed USB 2.0 ports to connect a keyboard directly, one LCD for visual presentation, flash memory as a storage and 2 UARTs for old PS/2 type keyboards, some LEDs and buttons.

For Ethernet protocols, we used open source TCP stack called Lightweight IP (lwIP) socket in raw mode. All codes developed for embedded platform are in ANSI-C

programming language.

The platform has verified with a oscilloscope that it is capable to produce keystroke events in a precision of 10 µs which is 10 times accurate than those required by CMU's timings.

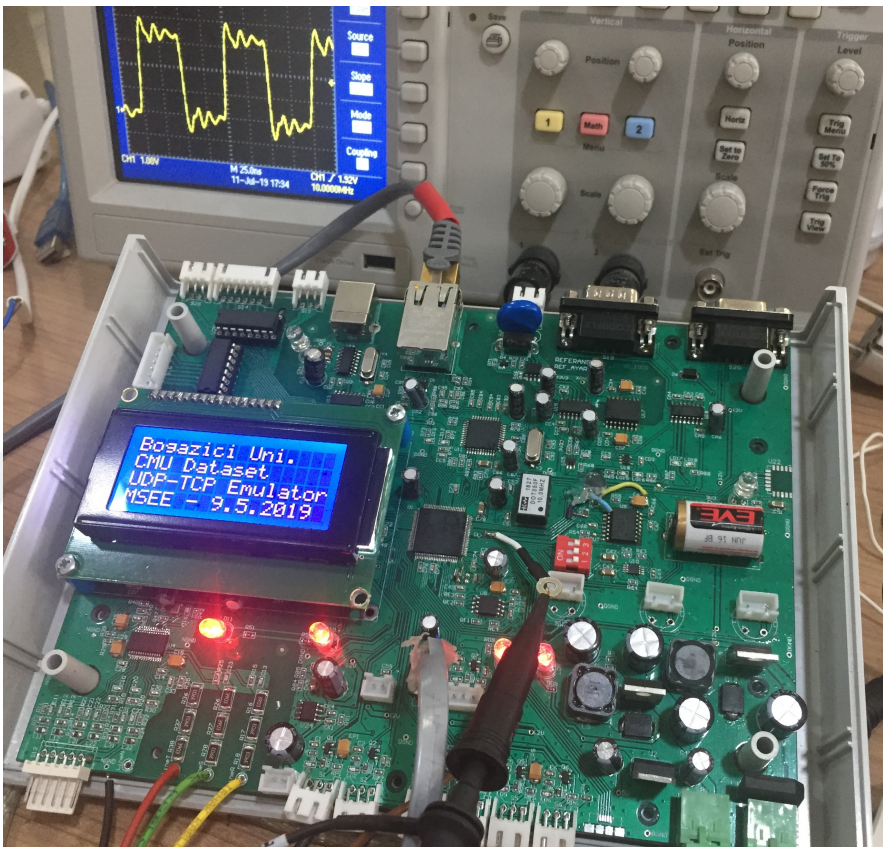

Figure 4.3. Designed embedded platform - CMU dataset emulator.

Embedded program binds port number 55550 for TCP connections and 55551 for UDP statically and wait for client requests in following protocols shown in Figure 4.4.

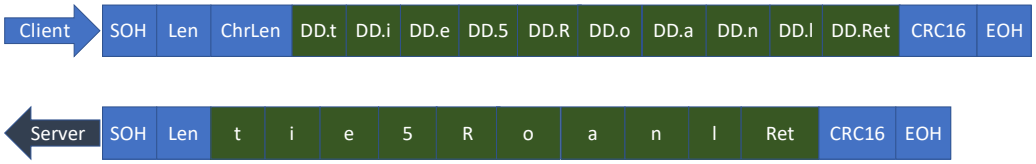

Figure 4.4. Designed protocol for CMU emulator.

4.6. Windows Application

CMU's key-stoke data-set holds timing data of 51 different users in 20400 records as total. Whole real time emulation takes more than 14 hours. Developed Windows Application shown in Figure 4.5, emulates CMU's data-set for 5 different scenarios:

(i) UDP: UDP protocol is a connection-less protocol and used by applications requires low-latency. The application emulated CMU's keystrokes through UDP and stored for further investigation.

(ii) TCP with Nagle Algorithm: Nagle algorithm is enabled by default in a TCP socket which affects the performance of classifiers as we discussed in section 4.4.1. The application emulated CMU's keystrokes through TCP with Nagle Algorithm enabled and stored.

(iii) TCP without Nagle Algorithm: TCP_NODELAY flag has been set in both client and server side and above emulation repeated and results are stored.

(iv) TCP SSL-AES128: Many recent applications use secure layer to access cloud services such as SSH, HTTPS etc. SSL/TLS layer uses asymmetric encryption such as RSA at session initialization for symmetric key exchange. Symmetric encryption method is negotiated by Client and Server and later data transfer has done based on this negotiation. AES is one of the most widely used symmetric encryption method and CMU data-set emulated by using 128 bit symmetric encryption and results are stored.

(v) TCP SSL-AES256: Some application may require stronger encryption, so we emulated also in AES-256 bit mode and results are stored.

Windows Application has developed using Delphi Community Edition [66] that uses Object Pascal and compiled for x64 platforms in order to get rid of timing accuracy bug available in x86 mode of Intel CPU.

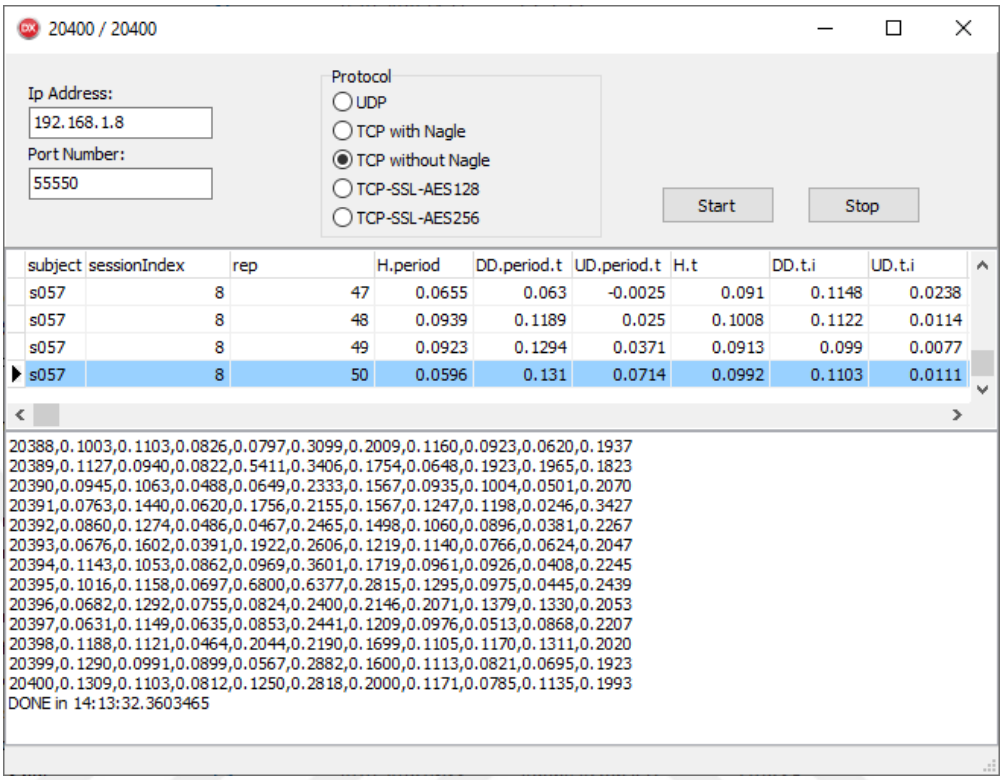

Figure 4.5. Windows application to simulate CMU keystroke dataset over the network.

4.7. Results

In this section we provided results of all classifiers as tables which are investigated and implemented and optimized from scratch in this thesis.

In the literature, classifiers utilizing the CMU dataset report their performances using all 31 features in the dataset [16]. We reproduced their results and extended with more classifiers using the same systematical approach. EER and ACC results of the original CMU dataset listed in the Table 4.1, as well as ZMFAR in Table 4.2.

Table 4.1. Original CMU dataset with all features, ordered by ACC.

CMU dataset with all features, ordered by mean ACC					
No	**Method**	**EER**	**Std EER**	**ACC**	**Std ACC**
1	z-score_optim_nu(1.449)	0.087	0.064	0.911	0.065
2	Nearest Neighbor with PSR	0.090	0.060	0.910	0.060
3	Scaled Manhattan	0.096	0.069	0.904	0.069
4	Nearest Neighbor with Mahal.	0.100	0.064	0.900	0.064
5	One-Class SVM	0.100	0.064	0.900	0.064
6	z-score	0.102	0.077	0.895	0.079
7	Bayesian	0.110	0.065	0.890	0.064
8	Mahalanobis	0.110	0.065	0.890	0.064
9	Filtered Manhattan	0.136	0.083	0.864	0.083
10	Multilayer Perceptron NN	0.147	0.082	0.854	0.082
11	Manhattan	0.153	0.092	0.847	0.092
12	k-means	0.154	0.070	0.846	0.070
13	Standard Neural Network	0.154	0.119	0.845	0.118
14	Normalized Mahalanobis	0.162	0.100	0.837	0.100
15	Euclidean	0.171	0.095	0.829	0.095
16	Normalized Euclidean	0.215	0.119	0.785	0.118

Table 4.2. Original CMU dataset with all features ordered by ZMFAR.

CMU dataset with all features, ordered by mean ZMFAR			
No	**Method**	**ZMFAR**	**Std ZMFAR**
1	Nearest Neighbor with PSR	0.442	0.275
2	Nearest Neighbor with Mahal.	0.468	0.272
3	Bayesian	0.482	0.273
4	Mahalanobis	0.482	0.273
5	One-Class SVM	0.486	0.312

Table 4.2. Original CMU dataset with all features ordered by ZMFAR. (cont.)

CMU dataset with all features, ordered by mean ZMFAR			
No	**Method**	**ZMFAR**	**Std ZMFAR**
6	Scaled Manhattan	0.601	0.337
7	z-score_optim_nu(1.449)	0.618	0.302
8	Normalized Mahalanobis	0.648	0.295
9	k-means	0.663	0.282
10	Multilayer Perceptron NN	0.751	0.266
11	Filtered Manhattan	0.755	0.283
12	Standard Neural Network	0.776	0.292
13	z-score	0.782	0.306
14	Manhattan	0.843	0.242
15	Euclidian	0.875	0.200
16	Normalized Euclidian	0.911	0.148

Since not all of these physical features are available during network transmissions, Table 4.3 lists the EER and Table 4.4 lists the ZMFAR 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.

Table 4.3. Original CMU dataset, *DD* features, ordered by ACC.

CMU dataset, DD.key1.key2 features, ordered by mean ACC					
No	**Method**	**EER**	**Std EER**	**ACC**	**Std ACC**
1	Nearest Neighbor with PSR	0.115	0.062	0.885	0.062
2	Nearest Neighbor with Mahal.	0.121	0.064	0.878	0.064
3	One-Class SVM	0.138	0.065	0.862	0.064
4	z-score_optim_nu(1.292)	0.145	0.081	0.848	0.082

Table 4.3. Original CMU dataset, *DD* features, ordered by ACC. (cont.)

CMU dataset, DD.key1.key2 features, ordered by mean ACC					
No	**Method**	**EER**	**Std EER**	**ACC**	**Std ACC**
5	Scaled Manhattan	0.151	0.079	0.847	0.079
6	Bayesian	0.153	0.067	0.847	0.068
7	Mahalanobis	0.153	0.067	0.847	0.068
8	Filtered Manhattan	0.154	0.089	0.846	0.089
9	k-means	0.156	0.070	0.843	0.070
10	Normalized Mahalanobis	0.168	0.070	0.832	0.069
11	Manhattan	0.172	0.095	0.829	0.095
12	Euclidean	0.187	0.087	0.813	0.087
13	z-score	0.189	0.106	0.808	0.103
14	Multilayer Perceptron NN	0.201	0.092	0.799	0.092
15	Standard Neural Network	0.201	0.154	0.799	0.156
16	Normalized Euclidean	0.219	0.116	0.781	0.116

Table 4.4. Original CMU dataset, *DD* features, ordered by ZMFAR.

CMU dataset, DD.key1.key2 features ordered by mean ZMFAR			
No	**Method**	**ZMFAR**	**Std ZMFAR**
1	Nearest Neighbor with PSR	0.518	0.264
2	Nearest Neighbor with Mahal.	0.523	0.248
3	Bayesian	0.703	0.276
4	Mahalanobis	0.703	0.276
5	k-means	0.725	0.271
6	One-Class SVM	0.733	0.291
7	Normalized Mahalanobis	0.778	0.262
8	Filtered Manhattan	0.843	0.212
9	Scaled Manhattan	0.847	0.211

Table 4.4. Original CMU dataset, *DD* features, ordered by ZMFAR. (cont.)

CMU dataset, DD.key1.key2 features ordered by mean ZMFAR			
No	**Method**	**ZMFAR**	**Std ZMFAR**
10	Standard Neural Network	0.848	0.264
11	Multilayer Perceptron NN	0.909	0.128
12	Manhattan	0.911	0.154
13	Normalized Euclidean	0.927	0.136
14	z-score_optim_nu(1.292)	0.946	0.186
15	Euclidean	0.968	0.053
16	z-score	0.982	0.127

Secondly, Table 4.5 and Table 4.6 present the EER and ZMFAR 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 4.5. UDP emulation, *DD* features, ordered by ACC.

UDP emulation, ordered by mean ACC					
No	**Method**	**EER**	**Std EER**	**ACC**	**Std ACC**
1	Nearest Neighbor with PSR	0.115	0.062	0.885	0.062
2	Nearest Neighbor with Mahal.	0.121	0.065	0.878	0.065
3	One-Class SVM	0.137	0.063	0.862	0.063
4	Scaled Manhattan	0.151	0.079	0.849	0.079
5	z-score_optim_nu(1.291)	0.145	0.081	0.847	0.082
6	Bayesian	0.153	0.068	0.847	0.068

Table 4.5. UDP emulation, *DD* features, ordered by ACC. (cont.)

UDP emulation, ordered by mean ACC					
No	**Method**	**EER**	**Std EER**	**ACC**	**Std ACC**
7	Mahalanobis	0.153	0.038	0.847	0.068
8	Filtered Manhattan	0.154	0.089	0.846	0.089
9	k-means	0.156	0.070	0.843	0.070
10	Normalized Mahalanobis	0.169	0.070	0.832	0.069
11	Manhattan	0.172	0.095	0.829	0.095
12	Euclidean	0.187	0.087	0.813	0.087
13	z-score	0.190	0.106	0.808	0.103
14	Standard Neural Network	0.202	0.153	0.797	0.155
15	Multilayer Perceptron NN.	0.218	0.101	0.783	0.101
16	Normalized Euclidean	0.219	0.116	0.781	0.116

Table 4.6. UDP emulation, *DD* features, ordered by ZMFAR.

UDP emulation, ordered by mean ZMFAR			
No	**Method**	**ZMFAR**	**Std ZMFAR**
1	Nearest Neighbor with PSR	0.522	0.262
2	Nearest Neighbor with Mahal.	0.527	0.248
3	Bayesian	0.706	0.275
4	Mahalanobis	0.706	0.275
5	k-means	0.730	0.268
6	One-Class SVM	0.739	0.289
7	Normalized Mahalanobis	0.780	0.260
8	Filtered Manhattan	0.843	0.211
9	Scaled Manhattan	0.848	0.209
10	Standard Neural Network	0.849	0.264
11	Multilayer Perceptron NN.	0.887	0.174

Table 4.6. UDP emulation, *DD* features, ordered by ZMFAR. (cont.)

UDP emulation, ordered by mean ZMFAR			
No	**Method**	**ZMFAR**	**Std ZMFAR**
12	Manhattan	0.910	0.154
13	Normalized Euclidean	0.926	0.136
14	z-score_optim_nu(1.291)	0.946	0.186
15	Euclidean	0.968	0.052
16	z-score	0.982	0.127

Thirdly, Table 4.7 and Table 4.8 list the changes in EER and ZMFAR values when using TCP. The impact of the Nagle algorithm [67], 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 significantly affects the physical characteristics of a user's keystroke dynamics.

Table 4.7. TCP with Nagle emulation, *DD* features, ordered by ACC.

TCP with Nagle emulation, ordered by mean ACC					
No	**Method**	**EER**	**Std EER**	**ACC**	**Std ACC**
1	Nearest Neighbor with Mahal.	0.189	0.069	0.811	0.069
2	Bayesian	0.189	0.078	0.811	0.078
3	Mahalanobis	0.189	0.078	0.811	0.078
4	Nearest Neighbor with PSR	0.189	0.067	0.810	0.067
5	k-means	0.225	0.083	0.775	0.083
6	Filtered Manhattan	0.231	0.087	0.769	0.087
7	Scaled Manhattan	0.234	0.090	0.766	0.090

Table 4.7. TCP with Nagle emulation, *DD* features, ordered by ACC. (cont.)

TCP with Nagle emulation, ordered by mean ACC					
No	**Method**	**EER**	**Std EER**	**ACC**	**Std ACC**
8	Manhattan	0.239	0.090	0.761	0.090
9	z-score_optim_nu(1.195)	0.243	0.097	0.754	0.106
10	Normalized Mahalanobis	0.247	0.091	0.753	0.091
11	z-score	0.259	0.103	0.745	0.101
12	Standard Neural Network	0.279	0.145	0.720	0.150
13	Multilayer Perceptron NN	0.286	0.090	0.715	0.089
14	Euclidean	0.325	0.097	0.677	0.095
15	Normalized Euclidean	0.332	0.088	0.669	0.088
16	One-Class SVM	0.267	0.078	0.544	0.142

Table 4.8. TCP with Nagle emulation, *DD* features, ordered by ZMFAR.

TCP with Nagle emulation, ordered by mean ZMFAR			
No	**Method**	**ZMFAR**	**Std ZMFAR**
1	Bayesian	0.872	0.173
2	Mahalanobis	0.872	0.173
3	k-means	0.892	0.162
4	Normalized Mahalanobis	0.893	0.162
5	Nearest Neighbor with PSR	0.906	0.116
6	Standard Neural Network	0.907	0.175
7	Nearest Neighbor with Mahal.	0.913	0.103
8	One-Class SVM	0.926	0.122
9	Scaled Manhattan	0.938	0.122
10	Filtered Manhattan	0.950	0.097
11	Manhattan	0.959	0.083
12	Multilayer Perceptron NN	0.959	0.106

Table 4.8. TCP with Nagle emulation, *DD* features, ordered by ZMFAR. (cont.)

TCP with Nagle emulation, ordered by mean ZMFAR			
No	**Method**	**ZMFAR**	**Std ZMFAR**
13	z-score_optim_nu(1.195)	0.976	0.074
14	Normalized Euclidean	0.980	0.042
15	Euclidean	0.987	0.016
16	z-score	1.000	0.000

Table 4.9. TCP emulation, *DD* features, ordered by ACC.

TCP emulation, ordered by mean ACC					
No	**Method**	**EER**	**Std EER**	**ACC**	**Std ACC**
1	Nearest Neighbor with PSR	0.116	0.061	0.884	0.061
2	Nearest Neighbor with Mahal.	0.124	0.064	0.876	0.064
3	One-Class SVM	0.138	0.064	0.862	0.064
4	Scaled Manhattan	0.152	0.078	0.849	0.078
5	z-score_optim_nu(1.277)	0.145	0.080	0.847	0.080
6	Bayesian	0.153	0.067	0.846	0.067
7	Mahalanobis	0.153	0.067	0.846	0.067
8	Filtered Manhattan	0.154	0.089	0.846	0.089
9	k-means	0.157	0.070	0.843	0.070
10	Normalized Mahalanobis	0.169	0.069	0.830	0.069
11	Manhattan	0.172	0.095	0.828	0.095
12	Euclidean	0.187	0.088	0.814	0.088
13	z-score	0.191	0.105	0.807	0.102
14	Standard Neural Network	0.211	0.160	0.785	0.165
15	Multilayer Perceptron NN.	0.218	0.097	0.783	0.097
16	Normalized Euclidean	0.219	0.116	0.781	0.116

Table 4.10. TCP emulation, *DD* features, ordered by ZMFAR.

TCP emulation, ordered by mean ZMFAR			
No	**Method**	**ZMFAR**	**Std ZMFAR**
1	Nearest Neighbor with PSR	0.529	0.259
2	Nearest Neighbor with Mahal.	0.535	0.245
3	Bayesian	0.705	0.277
4	Mahalanobis	0.705	0.277
5	k-means	0.723	0.270
6	One-Class SVM	0.738	0.288
7	Normalized Mahalanobis	0.781	0.259
8	Filtered Manhattan	0.842	0.212
9	Scaled Manhattan	0.845	0.212
10	Standard Neural Network	0.868	0.242
11	Multilayer Perceptron NN	0.899	0.159
12	Manhattan	0.911	0.151
13	Normalized Euclidean	0.927	0.135
14	z-score_optim_nu(1.277)	0.937	0.193
15	Euclidean	0.969	0.050
16	z-score	0.982	0.127

And finally, Table 4.11, Table 4.12, Table 4.13, Table 4.14 present the EER and ZMFAR 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.

Table 4.11. TCP-SSL-AES128, *DD* features, ordered by ACC.

TCP-SSL-AES128, ordered by mean ACC					
No	**Method**	**EER**	**Std EER**	**ACC**	**Std ACC**
1	Nearest Neighbor with PSR	0.116	0.061	0.884	0.061
2	Nearest Neighbor with Mahal.	0.122	0.064	0.878	0.064
3	One-Class SVM	0.138	0.063	0.862	0.063
4	Scaled Manhattan	0.152	0.079	0.848	0.079
5	z-score_optim_nu(1.273)	0.146	0.079	0.848	0.079
6	Bayesian	0.152	0.068	0.847	0.068
7	Mahalanobis	0.152	0.068	0.847	0.068
8	Filtered Manhattan	0.154	0.089	0.846	0.089
9	k-means	0.156	0.070	0.844	0.069
10	Normalized Mahalanobis	0.169	0.068	0.831	0.068
11	Manhattan	0.172	0.094	0.829	0.095
12	Euclidean	0.186	0.087	0.814	0.087
13	z-score	0.190	0.105	0.807	0.103
14	Multilayer Perceptron NN	0.210	0.093	0.791	0.093
15	Normalized Euclidean	0.218	0.116	0.782	0.115
16	Standard Neural Network	0.223	0.161	0.777	0.164

Table 4.12. TCP-SSL-AES128, *DD* features, ordered by ZMFAR.

TCP-SSL-AES128 emulation, ordered by mean ZMFAR			
No	**Method**	**ZMFAR**	**Std ZMFAR**
1	Nearest Neighbor with PSR	0.541	0.270
2	Nearest Neighbor with Mahal.	0.545	0.256
3	Bayesian	0.714	0.285
4	Mahalanobis	0.714	0.285
5	One-Class SVM	0.739	0.294

Table 4.12. TCP-SSL-AES128, *DD* features, ordered by ZMFAR. (cont.)

TCP-SSL-AES128 emulation, ordered by mean ZMFAR			
No	**Method**	**ZMFAR**	**Std ZMFAR**
6	k-means	0.743	0.265
7	Normalized Mahalanobis	0.773	0.273
8	Filtered Manhattan	0.837	0.216
9	Scaled Manhattan	0.840	0.224
10	Standard Neural Network	0.877	0.221
11	Multilayer Perceptron NN	0.882	0.180
12	Manhattan	0.907	0.160
13	Normalized Euclidean	0.929	0.135
14	z-score_optim_nu(1.273)	0.942	0.181
15	Euclidean	0.968	0.056
16	z-score	0.983	0.123

Table 4.13. TCP-SSL-AES256, *DD* features, ordered by ACC.

TCP-SSL-AES256, ordered by mean ACC					
No	**Method**	**EER**	**Std EER**	**ACC**	**Std ACC**
1	Nearest Neighbor with PSR	0.116	0.062	0.884	0.062
2	Nearest Neighbor with Mahal.	0.123	0.064	0.877	0.064
3	One-Class SVM	0.139	0.064	0.861	0.064
4	z-score_optim_nu(1.293)	0.146	0.082	0.849	0.083
5	Scaled Manhattan	0.152	0.079	0.848	0.079
6	Bayesian	0.153	0.068	0.846	0.068
7	Mahalanobis	0.153	0.068	0.846	0.068
8	Filtered Manhattan	0.154	0.088	0.846	0.088
9	k-means	0.157	0.069	0.842	0.069
10	Normalized Mahalanobis	0.169	0.068	0.831	0.068

Table 4.13. TCP-SSL-AES256, *DD* features, ordered by ACC. (cont.)

TCP-SSL-AES256, ordered by mean ACC					
No	**Method**	**EER**	**Std EER**	**ACC**	**Std ACC**
11	Manhattan	0.171	0.094	0.829	0.094
12	Euclidean	0.186	0.087	0.814	0.086
13	z-score	0.190	0.105	0.808	0.103
14	Standard Neural Network	0.209	0.158	0.791	0.159
15	Multilayer Perceptron NN	0.216	0.092	0.784	0.092
16	Normalized Euclidean	0.219	0.116	0.781	0.116

Table 4.14. TCP-SSL-AES256, *DD* features, ordered by ZMFAR.

TCP-SSL-AES256, ordered by mean ZMFAR			
No	**Method**	**ZMFAR**	**Std ZMFAR**
1	Nearest Neighbor with PSR	0.499	0.263
2	Nearest Neighbor with Mahal.	0.507	0.251
3	Bayesian	0.701	0.287
4	Mahalanobis	0.701	0.287
5	k-means	0.730	0.268
6	One-Class SVM	0.737	0.288
7	Normalized Mahalanobis	0.771	0.267
8	Scaled Manhattan	0.838	0.236
9	Filtered Manhattan	0.840	0.212
10	Standard Neural Network	0.869	0.233
11	Manhattan	0.907	0.158
12	Multilayer Perceptron NN	0.922	0.118
13	Normalized Euclidean	0.926	0.136
14	z-score_optim_nu(1.293)	0.946	0.186
15	Euclidean	0.969	0.052

Table 4.14. TCP-SSL-AES256, *DD* features, ordered by ZMFAR. (cont.)

TCP-SSL-AES256, ordered by mean ZMFAR			
No	**Method**	**ZMFAR**	**Std ZMFAR**
16	z-score	1.000	0.000

5. CONCLUSION

In the first part of the study, we implemented and evaluated 15 different anomaly detectors from the literature using the CMU keystroke dataset, ranking them from best to worst based on performance across all subjects. Statistical analysis was performed to identify the top performers and to ensure that the observed differences were statistically significant. The results showed that for the original CMU dataset, the z-score_optim_nu (1.449) detector achieved the best equal error rate (EER) of 0.087%. The Nearest Neighbor (PSR) and Scaled Manhattan detectors also performed well in terms of EER. The Nearest Neighbor (PSR) detector had the best zero-miss false alarm rate at 0.442. Other top detectors included the classical Mahalanobis and one-class SVM algorithms. The Nearest Neighbor detector was particularly robust. The top performers shared similar scaling characteristics, indicating potential areas for future research. However, some error rates in this study were higher than those reported in the literature, and some top performers differed from previous findings.

In the second part of the study, we proposed a GRU-based Siamese Network, which eliminates the need for averaging. We reported the performance of the model in the presence of group leakage. Additionally, we introduced a method that completely eliminates leakage and compared the performance of the same model with and without leakage.

In the third part of the study, we 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 classifiers, which are 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. The study revealed that classifiers demonstrating the best performance on the CMU dataset

in a local computer do not maintain the same performance levels in remote network environments, particularly under conditions involving Nagle's algorithm. Conversely, some classifiers that perform poorly on local datasets exhibited improved performance in network environments. Under UDP, the Nearest Neighbor with PSR was the best, followed by Nearest Neighbor with Mahalanobis and one-class SVM. With the Nagle algorithm in TCP, the Nearest Neighbor with Mahalanobis achieved the highest accuracy at 81.1%. Disabling Nagle and enabling encryption, the Nearest Neighbor with PSR again performed best, with 88.4% accuracy. This study also showed that a cloud-based PAM system can achieve about 90% accuracy with just one layer of data input. Accuracy can be further improved by incorporating additional layers, as mentioned in Section 1.4.

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 network environments, such as WiFi [68], 3G, 4G, and 5G [69], can provide a comprehensive understanding of their effectiveness under different conditions.

REFERENCES

1. Hovav, A. and R. Berger, "Tutorial: Identity Management Systems and Secured Access Control", *Communications of the Association for Information Systems*, Vol. 25, No. 42, pp. 531–570, 2009.

2. Cleven, A. and R. Winter, "Regulatory Compliance in Information Systems Research – Literature Analysis and Research Agenda", *Proceedings of the 11th International Conference on Enterprise, Business-Process and Information Systems Modeling*, pp. 183–194, Berlin, Heidelberg, 2009.

3. Joseph Pato, O. C. C., "Identity Management: Setting Context", *Hewlett-Packard, Cambridge, MA*, Vol. 29, No. HPL-2003-72, pp. 1–12, 2003.

4. Ludwig Fuchs, R. S., Günther Pernul, "Roles in Information Security–A Survey and Classification of the Research Area", *Computers & Security*, Vol. 30, No. 8, pp. 748–769, 2011.

5. Mendling, J., M. Strembeck and J. Recker, "Factors of Process Model Comprehension—Findings from a Series of Experiments", *Decision Support Systems*, Vol. 53, No. 1, p. 195–206, 2012.

6. Bakshi, A. and Y. B. Dujodwala, "Securing Cloud from Distributed Denial of Service Attacks Using Intrusion Detection System in Virtual Machine", *Second International Conference on Communication Software and Networks*, pp. 260–264, Singapore, 2010.

7. Hlavacs, H. and G. Kotsis, "Modeling User Behavior: a Layered Approach", *Proceedings of the Seventh International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 218–225, College Park, Maryland, 1999.

8. Dhanalakshmi, D. and J. K. Lakshmi, "A Survey on Web User Personalization Techniques", *IEEE International Conference on Computational Intelligence and Computing Research*, pp. 1–5, Coimbatore, India, 2014.

9. Zhu, X., S. Wu and G. Zou, "User Behavior Detection for Online Survey via Sequential Pattern Mining", *Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control*, pp. 493–497, Qinhuangdao, China, 2015.

10. Kaur, P., A. Singhal and J. Kaur, "Spam Detection on Twitter: a Survey", *3rd International Conference on Computing for Sustainable Global Development*, pp. 2570–2573, New Delhi, India, 2016.

11. IEE, "IEE Colloquium on 'Kalman Filters: Introduction, Applications and Future Developments"', *Digest No.27*, pp. 1/1–1/6, London, UK, 1989.

12. CENELEC, *European Standard EN 50133-1: Alarm Systems. Access Control Systems for Use in Security Applications. Part 1: System Requirements*, European Committee for Electrotechnical Standardization, Brussels, Belgium, 2002.

13. Forsen, G., M. Nelson and R. S. Jr., "Personal Attributes Authentication Techniques", *Rome Air Development Center Technical Reports*, Vol. RADC-TR-77, No. 333, pp. 1–331, 1977.

14. Peacock, A., X. Ke and M. Wilkerson, "Typing Patterns: a Key To User Identification", *IEEE Security Privacy*, Vol. 2, No. 5, pp. 40–47, 2004.

15. Gaines, R. S., W. Lisowski, S. J. Press and N. Shapiro, *Authentication by Keystroke Timing: Some Preliminary Results*, RAND Corporation, Santa Monica, CA, 1980.

16. Killourhy, K. S. and R. A. Maxion, "Comparing Anomaly-Detection Algorithms for Keystroke Dynamics", *IEEE/International Federation for Information Processing International Conference on Dependable Systems Networks*, pp. 125–134, Lisbon,

Portugal, 2009.

17. Software, P. T., "Security Guide for Windows—Random Password Generator", 2007, `https://www.pctools.com/guides/password`, accessed in 2008.

18. Microsoft, "Microsoft Password Checker", 2011, `https://www.microsoft.com/security/pc-security/password-checker.aspx`, accessed on January 2, 2012.

19. Kang, P., S.-S. Hwang and S. Cho, "Continual Retraining of Keystroke Dynamics Based Authenticator", *Proceedings of the International Conference on Advances in Biometrics*, pp. 1203–1211, Berlin, Heidelberg, 2007.

20. Conrad, E., S. Misenar, J. Feldman and K. Riggins, *Certified Information Systems Security Professional Study Guide*, Syngress, Amsterdam, Netherlands, 2010.

21. Monaco, J. V. and C. C. Tappert, "The Partially Observable Hidden Markov Model and Its Application To Keystroke Dynamics", *Pattern Recognition*, Vol. 76, No. C, pp. 449–462, 2018.

22. Ali, M. L., J. V. Monaco and C. C. Tappert, "Biometric Studies with Hidden Markov Model and Its Extension on Short Fixed-Text Input", *IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference*, pp. 258–264, New York, USA, 2017.

23. Liakat Ali, M. and C. C. Tappert, "Partially Observable Hidden Markov Model/Support Vector Machine: a Hybrid Approach for Keystroke Biometric User Authentication", *IEEE International Conference on Real-Time Computing and Robotics*, pp. 612–617, Kandima, Maldives, 2018.

24. Jiang, C.-H., S. Shieh and J.-C. Liu, "Keystroke Statistical Learning Model for Web Authentication", *Proceedings of the 2nd Association for Computing Machinery Symposium on Information, Computer and Communications Security*, p. 359–361, Singapore, 2007.

25. Azevedo, G. L. F. B. G., G. D. C. Cavalcanti and E. C. B. C. Filho, "An Approach To Feature Selection for Keystroke Dynamics Systems Based on Particle Swarm Optimization and Feature Weighting", *IEEE Congress on Evolutionary Computation*, pp. 3577–3584, Singapore, 2007.

26. Nielsen, F. and S. Boltz, "The Burbea-Rao and Bhattacharyya Centroids", *IEEE Transactions on Information theory*, Vol. 57, No. 8, pp. 5455–5466, 2011.

27. Tan, P.-N., M. Steinbach and V. Kumar, *Introduction To Data Mining, (First Edition)*, Addison-Wesley Longman Publishing Co., Inc., USA, 2005.

28. Yu, E. and S. Cho, "Genetic Algorithm-Support Vector Machine Wrapper Approach for Feature Subset Selection in Keystroke Dynamics Identity Verification", *Proceedings of the International Joint Conference on Neural Networks*, pp. 2253–2257 vol.3, Portland, OR, USA, 2003.

29. Karnan, M. and M. Akila, "Personal Authentication Based on Keystroke Dynamics Using Soft Computing Techniques", *Second International Conference on Communication Software and Networks*, pp. 334–338, Singapore, 2010.

30. Liakat Ali, M., C. Tappert, M. Qiu and V. Monaco, "Keystroke Biometric Systems for User Authentication", *Journal of Signal Processing Systems*, Vol. 86, No. 2–3, pp. 175–190, 2016.

31. Duda, R. O., P. E. Hart and D. G. Stork, *Pattern Classification (2nd Edition)*, Wiley-Interscience, USA, 2000.

32. Bleha, S., C. Slivinsky and B. Hussien, "Computer-Access Security Systems Using Keystroke Dynamics", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, No. 12, pp. 1217–1222, 1990.

33. Joyce, R. and G. Gupta, "Identity Authentication Based on Keystroke Latencies", *Commun. Association for Computing Machinery*, Vol. 33, No. 2, pp. 168–176, 1990.

34. C. F. Araújo, L., L. Sucupira Jr, M. Lizarraga, L. Ling and J. Baptista T. Yabu-Uti, "User Authentication through Typing Biometrics Features", *Signal Processing, IEEE Transactions On*, Vol. 53, No. 2, pp. 851 – 855, 2005.

35. Cho, S., C. Han, D. Hee and H.-I. Kim, "Web-Based Keystroke Dynamics Identity Verification Using Neural Network", *Journal of Organizational Computing and Electronic Commerce*, Vol. 10, No. 4, pp. 295–307, 2000.

36. Haider, S., A. Abbas and A. K. Zaidi, "A Multi-Technique Approach for User Identification through Keystroke Dynamics", *IEEE International Conference on Systems, Man and Cybernetics. 'Cybernetics Evolving To Systems, Humans, Organizations, and their Complex Interactions' (Cat. No.0*, pp. 1336–1341 vol.2, Nashville, TN, USA, 2000.

37. Møller, M. F., "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning", *Neural Networks*, Vol. 6, No. 4, pp. 525 – 533, 1993.

38. MathWorks, "MATLAB version: 24.2.0 (R2024b)", `https://www.mathworks.com`, accessed in 2024.

39. Ateş, P. D., Ç. Ateş, M. Koca and E. Anarım, "Autoencoder and Extreme Value theory Based Unknown Intrusion Detection System", *31st Signal Processing and Communications Applications Conference*, pp. 1–4, İstanbul, Türkiye, 2023.

40. Schölkopf, B., J. C. Platt, J. C. Shawe-Taylor, A. J. Smola and R. C. Williamson, "Estimating the Support of a High-Dimensional Distribution", *Neural Computation*, Vol. 13, No. 7, pp. 1443–1471, 2001.

41. Schölkopf, B., R. C. Williamson, A. J. Smola, J. Shawe-Taylor and J. C. Platt, "Support Vector Method for Novelty Detection", *Advances in Neural Information Processing Systems 12*, pp. 582–588, 2000.

42. Zhong, Y., Y. Deng and A. K. Jain, "Keystroke Dynamics for User Authentica-

tion", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 117–123, Providence, RI, USA, 2012.

43. Davarci, E. and E. Anarim, "User Identification on Smartphones with Motion Sensors and Touching Behaviors", *30th Signal Processing and Communications Applications Conference*, pp. 1–4, Safranbolu, Türkiye, 2022.

44. Yıldırım, M. and E. Anarım, "Session-Based User Authentication via Mouse Dynamics", *27th Signal Processing and Communications Applications Conference*, pp. 1–4, Sivas, Türkiye, 2019.

45. Fide, M. and E. Anarım, "User Authentication with Gated Recurrent Units Based Siamese Networks Using Keyboard Usage Behaviour", *32nd Signal Processing and Communications Applications Conference*, pp. 1–4, Mersin, Türkiye, 2024.

46. Medvedev, V., A. Budžys and O. Kurasova, "Enhancing Keystroke Biometric Authentication Using Deep Learning Techniques", *18th Iberian Conference on Information Systems and Technologies*, pp. 1–6, Aveiro, Portugal, 2023.

47. Tolosana, R., R. Vera-Rodriguez, J. Fierrez and J. Ortega-Garcia, "Biotouchpass2: Touchscreen Password Biometrics Using Time-Aligned Recurrent Neural Networks", *IEEE Transactions on Information forensics and Security*, Vol. 15, No. 9, pp. 2616–2628, 2020.

48. Acién Ayala, A., A. Morales Moreno, R. Vera Rodríguez, J. Fiérrez Aguilar, J. V. Monaco *et al.*, "Typenet: Scaling Up Keystroke Biometrics", *2020-IEEE/International Association for Pattern Recognition International Joint Conference on Biometrics*, pp. 1–7, Houston, TX, USA, 2020.

49. Deb, D., A. Ross, A. K. Jain, K. Prakah-Asante and K. V. Prasad, "Actions Speak Louder Than (Pass)Words: Passive Authentication of Smartphone Users via Deep Temporal Features", *International Conference on Biometrics*, pp. 1–8,

Crete, Greece, 2019.

50. Hadsell, R., S. Chopra and Y. LeCun, "Dimensionality Reduction by Learning An Invariant Mapping", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1735–1742, New York, NY, USA, 2006.

51. Ayotte, B., M. K. Banavar, D. Hou and S. Schuckers, "Group Leakage Overestimates Performance: a Case Study in Keystroke Dynamics", *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1410–1417, Nashville, TN, USA, 2021.

52. Elzinga, D. J. and D. W. Hearn, "The Minimum Covering Sphere Problem", *Management Science*, Vol. 19, No. 1, pp. 96–104, 1972.

53. Gurobi Optimization, L., "Gurobi Optimizer Reference Manual", 2008, `https://www.gurobi.com`, accessed on June 14, 2024.

54. Marsaglia, G., "Choosing a Point from the Surface of a Sphere", *The Annals of Mathematical Statistics*, Vol. 43, No. 2, pp. 645–646, 1972.

55. Acien, A., A. Morales, J. V. Monaco, R. Vera-Rodriguez and J. Fierrez, "Typenet: Deep Learning Keystroke Biometrics", *IEEE Transactions on Biometrics, Behavior, and Identity Science*, Vol. 4, No. 1, pp. 57–70, 2021.

56. Stallings, W., *Data and Computer Communications*, Prentice Hall Press, Upper Saddle River, NJ, USA, 2013.

57. IPv6.com, "UDP Protocol Overview", 1999, `https://ipv6.com`, accessed on August 17, 2011.

58. Jacobson, V., R. T. Braden and D. A. Borman, *Request for Comments 1072 - TCP Extensions for Long-Delay Paths*, Internet Engineering Task Force, Reston, VA, USA, 1988.

59. Nagle, J., *Request for Comments 896 - Congestion Control in IP/TCP Internetworks*, RFC Editor, Reston, VA, USA, 1984.

60. Braden, R. T., *Request for Comments 1122 - Requirements for Internet Hosts - Communication Layers*, RFC Editor, Reston, VA, USA, 1989.

61. Benvenuti, C., *Understanding Linux Network Internals*, O'Reilly Media, Inc., USA, 2005.

62. Comer, D. E., *Internetworking with TCP/IP, Volume 1: Principles, Protocols, and Architectures, Fourth Edition*, Prentice Hall PTR, USA, 2000.

63. Sack, H. and C. Meinel, *Internetworking - Technical Foundations and Applications*, Heidelberg, Germany, Springer, 2014.

64. Microsoft, "Microsoft High-Resolution Timers", 2022, `https://learn.microsoft.com`, accessed on July 3, 2024.

65. MIPS, "M-Class M51xx Core Family", 2015, `https://imgtec.com/mips/warrior/m-class-m51xx-core-family`, accessed on March 8, 2016.

66. Embarcadero, "Delphi Community Edition", 2018, `https://www.embarcadero.com`, accessed on July 18, 2018.

67. Peterson, L. L. and B. S. Davie, *Computer Networks: A Systems Approach*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

68. IEEE, *IEEE Standard for Information Technology–Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control and Physical Layer Specifications*, Institute of Electrical and Electronics Engineers, Piscataway, NJ, USA, 2016.

69. 3GPP, *Technical Specification Group Radio Access Network; NR; Physical Layer Procedures for Data (Release 16)*, 3rd Generation Partnership Project, Sophia Antipolis, France, 2021.

APPENDIX A: COPYRIGHT FOR THE REUSE OF THE FIGURES

The figures generated within the scope of this thesis study for which copyright has been transferred to the publisher have been incorporated into the thesis book in compliance with the publisher's applicable publication policy on the reuse of the author's original writings and graphics as specified on the publisher's official website.