A NOVEL ONLINE APPROACH TO DETECT DDOS ATTACKS USING MAHALANOBIS DISTANCE AND KERNEL-BASED LEARNING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY
BY

SALVA DANESHGADEH ÇAKMAKÇI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

NOVEMBER 2019

A NOVEL ONLINE APPROACH TO DETECT DDOS ATTACKS USING MAHALANOBIS DISTANCE AND KERNEL-BASED LEARNING

Submitted by SALVA DANESHGADEH ÇAKMAKÇI in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Information Systems Department, Middle East Technical University** by,

Prof. Dr. Deniz Zeyrek Bozşahin
Dean, **Graduate School of Informatics** ____________________

Prof. Dr. Sevgi Özkan Yıldırım
Head of Department, **Information Systems** ____________________

Prof. Dr. Nazife Baykal
Supervisor, **Information Systems Dept., METU** ____________________

Assoc. Prof. Thomas Kemmerich
Co-Supervisor, **Information Security and Communication Technology Dept., NTNU** ____________________

Examining Committee Members:

Prof. Dr. Sevgi Özkan Yıldırım
Information Systems Dept., METU ____________________

Prof. Dr. Nazife Baykal
Information Systems Dept., METU ____________________

Prof. Dr. Kemal Bıçakcı
Computer Engineering Dept., TOBB ETÜ ____________________

Assoc. Prof. Pekin Erhan Eren
Information Systems Dept., METU ____________________

Prof. Dr. Ali Aydın Selçuk
Computer Engineering Dept., TOBB ETÜ ____________________

Date: ______22.11.2019________

Name, Last name: Salva DANESHGADEH ÇAKMAKÇI

Signature : ____________________________

ABSTRACT

A NOVEL ONLINE APPROACH TO DETECT DDOS ATTACKS USING MAHALANOBIS DISTANCE AND KERNEL-BASED LEARNING

Daneshgadeh Çakmakçı, Salva

Ph.D., Department of Information Systems

Supervisor: Prof. Dr. Nazife Baykal

Co-Supervisor: Assoc. Prof. Thomas Kemmerich

November 2019, 90 pages

Distributed denial-of-service (DDoS) attacks are continually evolving as the computer and networking technologies and attackers' motivations are changing. In recent years, several supervised DDoS detection algorithms have been proposed. However, these algorithms require a priori knowledge of the classes and cannot automatically adapt to the frequently changing network traffic trends. This emphasizes the need for the development of new DDoS detection mechanisms that target zero-day and sophisticated DDoS attacks. To fulfill this need, an online sequential DDoS detection scheme that is suitable for use with multivariate data was proposed. The proposed algorithm utilizes a kernel-based learning algorithm, the Mahalanobis distance, and a Chi-square test. The algorithm is fully automated and does not require a pre-defined setting of any thresholds or baseline normal network traffic for training. Initially, four entropy-based and four statistical-based features were extracted from network flows as detection metrics per minute. Then, the Enhanced Kernel based Online Anomaly Detection Algorithm (E-KOAD) was employed to detect entropy-based input feature vectors that were suspected to be DDoS. This algorithm assumes no model for network traffic or DDoS in advance; then, it constructs and adapts a *Dictionary* of features that approximately span the subspace of normal behavior. Every T minutes, the Mahalanobis distance between suspicious vectors and the distribution of *Dictionary* members is measured. Subsequently, the Chi-square test is used to evaluate the Mahalanobis distance. The proposed DDoS detection scheme was applied to the CICIDS2017 dataset and the performance of the algorithm was measured using different performance metrics including accuracy, recall, precision and ROC-Curve. Finally, the results were compared with those by existing algorithms.

Keywords: DDoS, Machine algorithm, KOAD, Mahalanobis distance, Chi-square test

ÖZ

MAHALANOBIS UZAKLIĞI VE KERNEL TABANLI ÖĞRENME KULLANILARAK DDOS SALDIRILARINI TESPİT ETMEK İÇİN ÖZGÜN VE ÇEVRİMİÇİ BİR YAKLAŞIM

Daneshgadeh Çakmakçı, Salva

Ph.D., Bilişim Sistemleri Bölümü

Danışman: Prof. Dr. Nazife Baykal

Eş Danışman: Doç. Prof. Thomas Kemmerich

Kasım 2019, 90 sayfa

Bilgisayar, ağ teknolojileri ve saldırganların motivasyonları değiştikçe DDoS saldırıları sürekli olarak dönüşüm geçirmektedir.DDoS saldırılarını tespit etmek için geçtiğimiz yıllarda, birçok denetimli makine öğrenmesi algoritması önerilmiştir. Fakat bu algoritmalar sınıflarla ilgili ön bilgiye ihtiyaç duymakta ve sürekli değişen ağ trafiği trendlerine otomatik olarak uyum sağlayamamaktadırlar.Bu durum, sıfır günlük ve gelişmiş DDoS saldırılarını hedef alan yeni DDoS tespit etme mekanizmalarının geliştirilmesine olan ihtiyacı öne çıkmaktadır.Bu ihtiyacı karşılamak için bu çalışmada, çok değişkenli verilerle çalışmaya uygun olan çevrimiçi ve sıralı bir DDoS tespit etme şeması önerilmiştir.Önerdiğimiz algoritma; kernel tabanlı bir öğrenme algoritması, Mahalanobis uzaklığı ve Chi-square testinden yararlanmaktadır.Algoritma tamamen otomatiktir ve önceden tanımlanmış herhangi bir eşik değere veya normal ağ trafiğine ihtiyacı yoktur.Yapılan çalışmada öncelikle, ağ akışlarından, dakika başına dört adet entropi tabanlı ve dört adet de istatistiksel tabanlı özellik elde edilmiştir. Sonrasında, DDoS saldırısı olarak şüphelenilen, entropi tabanlı girdi özellik vektörlerini tespit etmek için kernel tabanlı öğrenme algoritması çalıştırılmıştır.Bu algoritma, ağ trafiği veya DDoS için herhangi bir modeli temel olarak varsaymamaktadır. Bunun yerine, normal davranışın çerçevesini yaklaşık olarak tanımlayan bir özellik kütüphanesi oluşturmakta ve bu kütüphaneyi kullanmaktadır. Şüpheli vektörler ve kütüphane üyelerinin dağılımı arasındaki Mahalanobis uzaklığı belirli bir periyotta ölçülmektedir. Sonrasında, bu mesafenin değerlendirilmesi için Chi-square testi kullanılmaktadır. Önerilen DDoS algılama yapısı CICIDS2017 veri setine uygulanmış ve doğruluk, anımsama, duyarlılık ve ROC eğrisini de içeren birçok parametre kullanılarak algoritmanın performansı ölçülmüştür. Son olarak, elde edilen sonuçlar mevcut algoritmaların performanslarıyla karşılaştırılmıştır.

Anahtar kelimeler: DDoS, Makine algoritması, KOAD, Mahalanobis uzaklığı, Chi-square

To My Love Şendoğan Çakmakçı

ACKNOWLEDGMENTS

First of all, I would like to express my sincere gratitude to my supervisors Prof. Dr. Nazife Baykal and Assoc. Prof. Thomas Kemmerich for their support, encouragement, feedback and guidance throughout this study.

I would wish to thank Assoc. Prof. Tarem Ahmed for his professional mentoring and his endless support, co-operation and patience. He generously shared his professional experiences in all steps of my Ph.D. research.

I would also like to express my gratitude to the examination committee members; Prof. Dr. Sevgi Özkan Yıldırım, Prof. Dr. Kemal Bıçakcı, Assoc. Prof. Pekin Erhan Eren, and Prof. Dr. Ali Aydın Selçuk, for their engagement and invaluable recommendations.

A very special appreciation to my husband, Şendoğan Çakmakçı for all his support. I could not accomplish my goal without your help, understanding, encouragement and patience throughout my Ph.D. study.

I am also wholeheartedly thankful to my parents, Hossein Daneshgadeh and Parvin Ghalichebaf, for their unconditional love and care throughout. I am so lucky for having parents like you.

Many thanks to my lovely sister, Samira Daneshkadeh, for her emotional support and being my best companion and closest confidant.

I would like to thank my grandmother, Mehranghiz Khabaze Azari, for her love and prayers throughout my studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BIRCH	**Balanced Iterative Reducing and Clustering Hierarchies**
CharGEN	**Character Generator**
CID	**Generalized Information Distance**
CLDAO	**Connection-less Lightweight Directory Access**
DBSCAN	**Density-Based Spatial Clustering of Applications with Noise**
DDoS	**Distributed Denial of service**
DGSOT	**Dynamically Growing Self-Organizing Tree**
DNS	**Domain Name System**
DoS	**Denial of Service**
DT	**Decision Tree**
EAC	**Evidence Accumulation Clustering**
ELM	**Extended Learning Machine**
FE	**Flash Event**
FPR	**False Positive Rate**
GE	**Generalize entropy**
HR-DDoS	**High Rate-DDoS**
IDS	**Intrusion Detection System**
IP	**Internet Protocol**
IPS	**Intrusion Prevention System**
ISP	**Internet Service Provider**
KOAD	**Kernel Online Anomaly Detection**
E-KOAD	**Extended Kernel Online Anomaly Detection**
KPCA	**Kernel Principal Component Analysis**
LAN	**Local Area Network**
LR-DDoS	**Low Rate-DDoS**
NTP	**Network Time Protocol**
PCA	**Principal component analysis**
PLS	**Partial Least Square**
R2U	**Remote to User**
RF	**Random Forest**
ROC Curve	**Receiver Operating Characteristic Curve**
ROC	**Receiver Operating Characteristics**

RST	**Rough Set Theory**
SDN	**Software Defined Network**
SIEM	**Security Information and Event Management**
SSC	**Sub-Space Clustering**
SSDP	**Simple Service Discovery Protocol**
SVM	**Support Vector Machine**
TCP	**Transmission Control Protocol**
TPR	**True Positive Rate**
U2R	**User to Root**
UDP	**User Datagram Protocol**
UPnP	**Universal Plug and Play**

CHAPTER 1

INTRODUCTION

The evolution of intelligent computer networks, distributed processing facilities, range of communication protocols, and arrays of smart devices has significantly revolutionized all modern critical infrastructures and business models. Today's technologies are firmly relying on network and communication facilities, which in turn makes them dependent on network security. The growing number of internet-based services and applications along with increasing adoption rate of connected wired and wireless devices present opportunities as well as technical challenges and threads. Therefore, security concerns increase exponentially for both individuals and service providers.

A cyber-attack is a malicious and deliberate activity by an individual or organization toward the computer and network system of another induvial or organization to compromise the secure operation of their information systems. Attackers peruse some gains by devastating the victim's network. Therefore, cybersecurity has become one of the primary concerns of the organizations to prevent or reduce the potentially severe consequences of the cyber-attacks. The terms cybersecurity and information security are often used interchangeably in the literature. In general, information security concentrates on protecting information assets, while cybersecurity encompasses more dimensions, including human resources and ethics (Reid and Van Niekerk, 2014). Information security aims to protect the triad of confidentiality, integrity, and availability (CIA) of data (ISO/IEC 27002:2013, 2013). Availability means that information should be available whenever authorized people need it. Integrity means that only authorized people can alter information or otherwise, would be the same as was produced or sent by a sender. Confidentiality means that information should only be accessible by authorized people. Cybersecurity also supports the authenticity and non-repudiation of data. Authenticity ensures the identity of the source. Non-repudiation means that the party of the communication cannot deny the authenticity of her/his signature. Moreover, cybersecurity can protect the privacy of the users.

Ahmed et al. (2016) defined network security as a subset of cybersecurity which deals with planning and implementing network security mechanisms. It protects the confidentiality and integrity of data while also ensure the availability of the resources. They categorized network attack types in four groups, including DoS/DDoS, probe, User to Root (U2R), and Remote to User (R2U). Many organizations today implement a variety of intrusion detection and prevention systems and employ cybersecurity experts to protect their network against different types of mentioned attacks. Not surprisingly, as the complexities and sophistication of attack vectors increase, the need for more robust and sophisticated detection methods also increase simultaneously.

This chapter begins with elaborating the problem and the importance of the subject within the global research community. The problem statement is defined, the gaps in the existing literature are presented, and the original contributions of this dissertation are highlighted.

1.1. Motivation

Many organizations today implement a variety of intrusion detection and prevention systems and employ cybersecurity experts to protect their Internet-enabled interests. Preventing revenue loss and data breach are the dominant motivations of the organizations to employ security solutions.

Gartner reported the approximate amount of $114 billion for the worldwide expenditure on information security in 2018. Gartner also predicted the 8.7 percent of growth to $124 billion for the market in 2019.

According to the recent report by Akamai [aka, 2019] money is the main concentration of all cyber-attacks. Subsequently, the financial services industry is fascinating for attackers. Distributed Denial of Service (DDoS) attack can cause millions of dollars lost for each minute of downtime for commercial organizations. Nevertheless, the financial services industry was the target for 40% of all the unique DDoS attacks. The report mentions to TCP SYN-ACK as the most common DDoS attack type against financial organizations in 2019.

The report by Verizon (2019) demonstrates the volume of DDoS attacks in different industries and indicates that the median of DDoS attack bandwidth does not change dramatically among various sectors. The report discusses DDoS protection as an essential control for information entities by considering the massive number of DDoS attacks in different industries.

According to the Arbor report (2016) The DDoS attack was the second most commonly experienced attacks in 2016 after Ransomware. The DDoS attack was reported as the top observed threat by service providers in 2017. Additionally, over two-thirds of the respondents cited the DDoS attack as a high threat to IPv6 networks. The vast majority of service providers who participated in the Arbor's survey represented the DDoS attack as the dominant threat. Firewall logs, Intrusion Detection System (IDS) and Security Information and Event Management (SIEM) were addressed as the top three most utilized tools to detect threats. However, half of the enterprise, government, and education respondents stated that their firewalls and IDS failed to detect DDoS attacks, or the event contributed to an outage during DDoS attack. Respondents also see online DDoS detection/mitigation systems as the most effective ways to detect threats. Additionally, there is an increasing demand toward best-practice hybrid and online automatic DDoS detection/mitigation systems in the market.

1.2. Problem Statement

There are hundreds of studies regarding DDoS detection in the literature. Many frameworks have been presented in academia and industry to predict, detect, and defend against DDoS attacks. Machine learning, knowledge-based, soft computing and statistical methods are examples of techniques which have been adopted to detect DDoS attacks. However, the nature of DDoS attack makes it very difficult to propose a method to cover the detection of all different types of DDoS attack. Modern firewalls and IDSs have some examples of flooding protection that enable them to mitigate some DoS/DDoS attacks such as volumetric DDoS. On the other hand, today's high-speed networks not only empower attackers to bombard their victims with high rate and volume of packets but also to configure themselves in a manner which can escape traditional firewall and IDS.

It is thus essential to develop a new DDoS detection mechanism which incorporates the best feature of existing practices while automating the detection process the most. This is the objective of this dissertation.

1.3. Gaps in the Existing Literature and Original Contributions

Despite the abundance of DDoS attack detection approaches available in the literature, the following significant gaps have revealed in existing knowledge. Based on the literature survey, several critical issues remain unresolved.

- The lack of new benchmark datasets for the validation of detection schemes, which leaded almost all authors used either old datasets or simulation data in a strictly controlled environment (Behal and Kumar, 2017).
- The lack of real-time DDoS attack detection systems (Behal and Kumar, 2017).
- The fixed threshold setting for statistical and entropy-based DDoS detection approaches.
- The limited number of unsupervised DDoS attack detection schemes that satisfy all the requirements of a real-world online DDoS detection algorithm (Ahmed et al., 2016).

This thesis makes the following original contributions:

- A novel algorithm is proposed for performing automated detection of DDoS attacks in the network of the organizations. The time complexity and memory requirements of the proposed algorithm is independent of time, which makes it naturally suitable for real-time use.
- Validation of the proposed algorithm with the recent benchmark dataset from the Canadian Institute for Cybersecurity (CICID2017) (Sharafaldin et al., 2018).
- Improving the well-known kernel online anomaly detection algorithm (Ahmed et al., 2007a, Ahmed et al., 2007b, Ahmed, 2009), which is cited in more than 150 studies, by automating threshold selection.
- Construction of the original feature vector by combining a small subset of traffic features based on the recommendations for both statistical and entropy-based features in the DDoS detection literature.

1.4. Scope and Limitations

The primary concentration of this dissertation is to propose an online and fully automated DDoS detection algorithm. The proposed algorithm can detect protocol-based (TCP, UDP, DNS, ICMP) flooding attacks at the victim's network. The network administrator can determine the time required to alarm a DDoS attack after suspicious traffic enters the network of the organization. The time can be defined based on the severity level of the system and the network bandwidth to tolerate large traffic volumes.
It is hypothesized that this algorithm can detect flooding DDoS attacks in any environment where the victim is located, such as LAN, WAN, ISP, cloud, and SDN networks. However, it requires future investigation, which is preserved for future work.
The algorithm is not able to differentiate different types of DDoS attacks, including High Rate-DDoS (HR-DDoS) and Low Rate-DDoS (LR-DDoS) from Flash Event (FE) traffic. The algorithm cannot be employed to detect reflection DDoS attacks. However, it is expected that the update of second feature vector (Please refer to section 4.2 for more details) will be enough to make algorithm applicable for detection of different reflection DDoS attacks such as DNS-based reflection DDoS.

1.5. Outline of Dissertation

The remainder of this dissertation is organized as follows. Chapter 2 describes the recent related work and surveys the literature on the topic. Chapter 3 presents the mathematical background of the proposed DDoS detection system. Chapter 4 presents the research design and presents the experimental setup. Chapter 5 presents the experimental results and empirical comparisons of the proposed scheme with existing methods. Chapter 6 concludes with directions for future work.

CHAPTER 2

LITERATURE REVIEW

This chapter begins with presenting an overview of the DDoS attack, taxonomy of DDoS attacks, and DDoS protection mechanisms. Subsequently, this chapter discusses the methods that have been recently proposed by various researchers to detect DDoS attacks. These works are organized based on different behavioral approaches to detect DDoS attacks, including statistical methods, information theory, supervised machine learning, unsupervised machine learning, and hybrid methods.

2.1. DDoS attack Overview

Denial-of-service (DoS) attack is a classic method of bringing down a victim network by preventing legitimate users (clients) of a service from accessing that service. Distributed DoS (DDoS) attacks are sophisticated, many-to-one version of DoS attacks, where the attacker overwhelms victim's resources by sending streams of packets to the victim. DDoS attackers not only aim to render a service inaccessible but also may pursue to gain unlimited access to the victim machine and thus cause more damage. According to Shameli-Sendi et al. (2015), the taxonomy of DDoS attacks involves six categories: the degree of automation (manual, semi-automatic, and automatic); the degree of attack rate (continuous, fluctuating, and increasing); the network scanning strategy (random, hit-list, topological, permutation, and local subset); the adopted strategy (protocol and brute-force attacks); the propagation mechanism (central, back-chaining, and autonomous); and finally the degree of impact (disruptive and degrading).

Currently, bot networks are usually utilized to increase the impact of the attacks. These bot networks consist of handler (master) and agent (bot/slave/zombie) machines. The attackers scan the network and compromise vulnerable machines by using them as bots. These bots are then commanded to bombard the victims with packets by specifying the attack type and the victim's address (Mirkovic and Reiher, 2004). Akamai Technologies (2019) recorded 7,822 DDoS attacks between November 1, 2017 and April 30, 2018, indicating the prevalence of such attacks despite the presence of multilevel Internet security measures. During the last six months of this period, companies confronted with DDoS attacks 41 times on average. The United States was identified as the source of 30% of the recorded DDoS attacks, that is, 46,137 source IP addresses (Akamai Technologies, 2019). Table I displays the summary of DDoS attacks statistics in 2016 based on the ATLAS data and the Arbor report (2016).

Table I Summary of DDoS attacks Statistics in 2016

DDoS attack	**2016**	
	First-Place	Second-Place
Peak Attack Size	500 Gbps	400 Gbps
DDoS Attack Types	Volumetric	State-Exhaustion
Attack duration	1% of attacks took more than 1 day	86% less than 30 minutes
DDoS Attack Motivations	Criminals demonstrating DDoS attack capabilities	Online gaming-related
Ports Targeted by DDoS Attacks	80	53
Protocols Used for Reflection/Amplification	DNS	NTP

Flash Event (FE) is another type of network traffic which deliberately causes a denial of service problem for legitimate users of an internet-based service or an application. FE resembles High-Rate DDoS (HR-DDoS) attacks, and it is difficult for network administrators to distinguish DDoS attack from FE traffic. This increase in the volume of legitimate traffic might be unpredicted, like spreading a piece of breaking news around the world. It also could be predicted like a pre-scheduled introducing of a new product by Apple company (Behal and Kumar, 2017).

2.2. Taxonomy of DDoS attack

DDoS attack vectors can be categorized in three groups:

- Volumetric Attacks: The fundamental goal of these types of attacks is to consume bandwidth of the network and causing congestion in the network. Bhuyan et al. (2015a) classified volumetric DDoS attacks into HR-DDoS which are similar to FE and Low Rate DDoS (LR-DDoS) which are similar to legitimate network traffic. According to Moor et al. (2006) if the rate of DDoS attack is more than 1000 packets per second, it will be considered as High Rate DDoS (HR-DDoS) attack.

- TCP State-Exhaustion Attacks: The ultimate aim of these attacks is to consume the whole connection state tables of network infrastructure components such as load balancers, firewalls, IDS, IPS, and the application.

- Application-Layer Attacks: In these types of attacks target the critical aspect of an application or service. They are a relatively effective attack because it consumes not only server resources but also network resources. In general, it is difficult to detect them with traditional flow-based monitoring methods.

DDoS attacks based on exploited vulnerability:

- SYN Flood: An attacker sends a massive number of spoofed SYN packets to a victim to establish a connection. The attacker never completes the three-way TCP communication process. The victim waits for the packet, but it never receives the SYN-ACK packet until the connection request becomes timeout. It results in an increasingly large number of half-open connections in the network. SYN packets are often used aa a powerful attack means because they are least likely to be rejected by a simple firewall rule.

- UDP Flood: An attacker floods the massive number of spoofed UDP packets to random servers or a specific server (with a particular IP address and port number). Finally, the attacker consumes all available bandwidth in a victim's network.

- UDP Fragment Attack: It encompasses fragmentation of different UDP packets in protocols like DNS and NTP.

- DNS Amplification Attack: It is a type of reflection attack which an attacker sends many DNS requests to several DNS servers with a spoofed IP address (IP address of victim). As a result, DNS servers simultaneously send DNS response packets to the victim. Finally, the victim's network is exhausted by the sheer number of DNS responses.

- Connection-less Lightweight Directory Access Protocol (CLDAP) Attack: CLDAP is an attack on port 389 which is used to retrieve server information by clients. In CLDAP attack, a malicious party sends LDAP/CLDAP requests to several servers with a spoofed sender IP address (victim's IP). Servers respond with their data inside a large response packet. As a result, the victim's network is overwhelmed with a significant number of LDAP response packets.

- NTP Attack: NTP is a simple UDP protocol which is used by connected machines to set their clocks automatically. It can be misused to start a reflected DDoS attack against a computer because the small request can result in a tremendous response.

- Character Generator (CharGEN) Flood: It is an old-fashioned protocol either TCP or UDP on port 19. When a TCP connection is started, the server initiates sending arbitrary characters to the connecting host until the connection is closed. In UDP based communication, when a host sends a datagram to the server, the server sends back a UDP datagram containing a random number of characters (0-512). This protocol can be misused to perform a CharGEN amplification DDoS attack. The attack is started by sending small packets carrying a spoofed sender IP address (victim's IP) to the connected devices running CharGEN. Then the connected devices which receives the packets start to send large UDP packets to the victim on port 19. Therefore, the victim is overwhelmed with a massive amount of response packets (between 0- 512 bytes depending on request).

- SSDP Attack: The Simple Service Discovery Protocol (SSDP) which is a part of the Universal Plug and Play (UPnP) Protocol has started to be miss-used as a new reflection DDoS attack. SSDP is enabled by default on most of home/office devices such as PCs, wireless access points, modems, routers, web cameras, smart TVs, scanners, and printers. The protocol is used to enable machines to seamlessly discover each other for communication and data transferring on the network.

- DNS Attack: It is an application layer attack towards DNS server using UDP floods. An attacker sends a large amount of spoofed DNS request packets from a massive set of source IP addresses. It is difficult for DNS server to differentiate a fake DNS request from the real one, so fraudulent requests drown the DNS server and make it out of service.

2.3. DDoS Attack Defence Mechanisms

A DDoS attack can adversely influence an organization at various levels, ranging from financial, prestige, and customer loss to data exfiltration. Therefore, an effective DDoS protection system is critical for preserving revenue, productivity, reputation, and user loyalty. Shameli-Sendi et al. (2015) defined four DDoS defense phases: prevention, monitoring, detection, and mitigation. Prevention includes the ideal protection mechanisms against all security concerns including DDoS attacks. However, DDoS prevention has become even more challenging as DDoS attacks are becoming increasingly scaled and sophisticated. They are designed so that they can bypass traditional prevention tools such as anti-viruses, firewalls, and intrusion prevention systems. Consequently, it is impossible to prevent every potential DDoS attack. DDoS detection is another protection mechanism for distinguishing attack traffic from normal network traffic. Shameli-Sendi et al. (2015) categorized DDoS attack detection algorithms into two main groups:

- Signature-based: where the characteristics of the captured traffic are compared with well-defined characteristics of previous and precisely modeled DDoS attacks. Signature-based attack detection systems are perfect in detecting known-attacks while they cannot prevent the novel attacks.

- Anomaly-based (Behavioral-based): Anomaly is defined as any deviation from the known or expected behavior of the systems. Anomaly-based attack detection systems require the usual profile of the undertaken system to compare with observed events in order to recognize significant attacks. This profile is developed based on many attributes such as network connections, hosts, users and so on. This profile should be updated whenever there is a change in any of the mentioned attributes. Anomaly-based detections systems are effective to detect unforeseen and nocel DDoS attacks. On the other hand, these systems are usually difficult to operate in a real-time manner (Liao et al., 2013). Jyothsna et al. (2011) summarized different anomaly detection algorithms including: "Statistical based, Operational or threshold metric model, Markov Process or Marker Model, Statistical Moments or Mean and Standard Deviation Model, Univariate Model, Multivariate Model, Time Series Model, Cognition based, Finite State Machine Model, Description Script Model, Adept System Model, Machine Learning based, Baysian Model, Genetic Algorithm Model, Neural Network Model, Fuzzy Logic Model, Outlier Detection Model, Computer Immunology based, User Intention based "

2.4. Review of DDoS Detection Studies

This section provides a brief review of DDoS detection studies based on behavioral approaches. These studies use either statistical methods, information theory, supervised machine learning, unsupervised machine learning, or hybrid methods, and they may be classified accordingly.

2.4.1 Works based on Statistical Algorithms

Bhuyan et al. (2015b) proposed the partial rank correlation scheme to detect both low-rate and high-rate DDoS attacks. The detection was based on two heuristically estimated thresholds for partial rank correlation (r). The detection accuracy of their algorithm is highly dependent on threshold selection. Normal and DDoS-attack traffic was selected from the DARPA-2000 (MIT Lincoln Laboratory, 2000) and CAIDA-2007 datasets (CAIDA, 2007) respectively. However, the integration of network traffic samples with different underlying network characteristics and typologies can influence the results.

Hoque et al. (2016) developed a Feature Feature Source (FFSc) to measure the degree of similarity in terms of standard deviation and mean value between input feature vectors. Each vector was composed of three network features. They computed a similarity value using FFSc for each vector of network traffic, and when the score passed a threshold, the attack alarm was generated. They evaluated their method using CAIDA DDoS 2007 and MIT DARPA datasets. Their proposed algorithm was able to classify normal, LR-DDoS, and HR-DDoS attacks. The empirical threshold setting and requiring normal background network traffic are shortcomings of their algorithm.

Nezhad et al. (2016) extracted two features, namely, the number of packets and the number of source IP addresses, from network traffic per minute to construct the detection feature vector. Subsequently, the Box-Cox transformation (Montgomery et al, 2015) was used to fix the variance of the time series representing the number of packets and thereby increase the prediction power of the Auto-Regressive Integrated Moving Average (ARIMA) model. In the next step, the local Lyapunov exponent was used to classify chaotic and non-chaotic errors. A detection rate of 99.5% was reported for their algorithm. This approach investigates only two well-known traffic features to detect DDoS attacks. Sophisticated attackers can mimic legitimate traffic and evade the detection tools by reducing the number of bots or the number of the sent packets per each bot. Additionally, time-series forecasting models such as ARIMA are supervised learning algorithms and are not suitable for online attack detection.

Behal et al. (2017) classified the different characteristics of FE and DDoS attacks based on the distribution of requests among source IPs, change in the rate of new IPs, change in the rate of traffic, number of different source IPs, number of different geographical distribution of source, duration and so. They utilized arithmetic mean, geometric mean and standard deviation to model the average decrease in growth pattern of new IPs for FE and DDoS traffic. They also measured different statistics for the average request per source IP. They validated their model using simulated data, FIFA'98, ACM SIGCOMM, DARPA and CAIDA datasets.

Zhou et al. (2017) used the expectation of packet size (EPS) to distinguish LR-DDoS-attack traffic from legitimate traffic. Their mechanism was based on the assumption that the volume of the sent/received bytes in LR-DDoS traffic is considerably lower than that in normal traffic. Considering the size of sent/received packets as the only detection feature is the limitation of this method, which also requires obtaining and storing the normal EPS in advance using a normal background traffic without any attack data.

David and Thomas (2019) used the number of packets, unique source IP addresses, unique destination IP addresses, and unique protocols (which were aggregated every T seconds) to construct the feature vector. Moreover, a slicing time window was used to calculate the mean and variance of these attributes. Subsequently, the threshold values were adaptively filtered for each feature based on its mean and variance. Their algorithm issues a DDoS attack alarm if and only if all four attributes exceed their thresholds. The limitation of this algorithm is that an attacker can slightly increase the traffic in the victim's network so that the threshold means, and variance may be smoothly increased over time, thus deceiving the detection scheme.

2.4.2 Works based on Unsupervised Machine Learning algorithms

Lee et al. (2008) constructed a feature vector of 9 traffic characteristics including entropy of source IP address and port number, entropy of destination IP address and port number, entropy of packet type, occurrence rate of packet type (ICMP, UDP, TCP SYN) and number of packets. Then they applied a clustering analysis based on Euclidean distance. Their algorithm aimed to group network traffic into normal, phase 1, phase 2, attack and post-attack classes. They used DARPA-2000 dataset to validate their algorithm. This algorithm is not suitable for online detection as the construction of the clusters is incremental, so any change in the trend of network traffic requires re-clustering of data from the beginning.

Casas et al. (2012a, 2012b) proposed an unsupervised network intrusion detection system that can detect different network attacks without requiring any type of signature, labeled traffic, normal traffic profile, or training to construct the normal profile of the network traffic. Initially, captured packets in consecutive time slots of fixed length T were aggregated in IP flows. Flow aggregation was performed at nine different resolution levels using different aggregation keys based on the network prefix in either the source or destination IP addresses of the flows. An algorithm running in three consecutive steps was developed. In the first step, anomalous time slots were detected using three simple and traditional volume metrics and the corresponding dynamic threshold values. In the second step, subspace clustering, density clustering, and evidence accumulation clustering were used to construct an outlier ranking mechanism. Finally, in the third step, a simple threshold detection approach was used to select anomalies among top-ranked outlying flows. It appears that this algorithm is not suitable for real-time detection because clustering algorithms must re-partition the entire space when points were added to or deleted from the system. Thus, it is difficult to satisfy real-time detection requirements. This method is not a dedicated DDoS-attack detecting algorithm.

Papalexakis et al. (2014) applied a co-clustering algorithm to isolate specific parameters which are indicators of abnormal connections. They validated their algorithm using KDD CUP'99. They mentioned that their algorithm was too slow to run over the full dataset.

Ahmed and Mahmood (2014) proposed a variation of the k-means algorithm, which was called *x*-mean algorithm. The algorithm did not require to set the number of the clusters (*k*) in advance. They used DoS data from DARBA dataset to train and test their algorithm. Their algorithm is not suitable for online detection as the construction of the clusters is sequential.

Ahmed and Mahmood (2015) proposed a collective anomaly detection method using a partitioned clustering technique. Initially, they used *x*-means algorithm to cluster the dataset and sort the clusters based on their size. Then, they summed the traffic features of the clusters to make a single new traffic feature and re-cluster the newly constructed feature vectors. They considered the cluster with the minimum variance as a collective anomaly. They used the KDD CUP’99, DARPA, and Kyoto datasets to train and test their method.

Dromard et al. (2016) proposed an Online and Real-time Unsupervised Network Anomaly Detection Algorithm (ORUNADA) based on the incremental grid clustering algorithm and a discrete time sliding window. The application of incremental grid clustering is the novel part of their approach, as incremental clustering algorithms require updating only the previous feature space partition instead of re-clustering the entire space whenever a point is added or removed. Subsequently, these updated partitions are merged to recognize the most dissimilar outliers. Incremental grid clustering makes the proposed algorithm more suitable for real-time detection. ORUNADA is not designed to detect only DDoS-attack traffic. However, its high computational power requirements limit its applicability (Roudiere and Owezarski, 2017).

Roudiere and Owezarski (2017) proposed an unsupervised DDoS detector called autonomous algorithm for traffic anomaly detection. This approach aimed at reducing the use of computational resources required to process the traffic. The algorithm has two steps: online and offline processing. In online processing, the continuous part quickly handles flow-based feature extraction and uses histograms to model the flow feature distribution for source/destination IP/port. Additionally, traffic-wide densities are calculated for the number of SYN, UDP, and ICMP packets. In offline processing, snapshots of the traffic are taken every T seconds, and the traffic is modeled accordingly. Finally, the k-NN algorithm is used to compare a snapshot with the last N snapshots to detect significant deviations from the usual traffic profile. The algorithm exhibited promising performance on a simulated dataset, but the approach was not validated using a benchmark dataset. Additionally, the algorithm requires the experimental setting of certain parameters, that is, the number of nearest neighbors (k), the total number of snapshots (N) for the k-NN analysis, and the density factor (λ). Moreover, network traffic sampling (snapshots) may affect the detection accuracy.

2.4.3 Works based on Supervised Machine Learning Algorithms

Seo et al. (2005) computed the TRA (Traffic Rate Analysis) to analyze the characteristics of network traffic for DDoS attacks and then they employed a multi-class SVM classification to detect different types of DDoS attacks. TRA examines the occurrence rate of a specific kind of packet within the stream network traffic based on a TCP flag rate and a protocol rate. This approach is not applicable as online learning detection as it requires prior knowledge of the normal network traffic and labeled data to train SVM classifier in offline mode.

Xu et al. (2007) proposed a group of relative values features (RLT features) based on characteristics of DDoS attack to increase the precision of distinguishing normal streams from DDoS attack streams. RLT feature vector was composed of six features including one-way connection density, average length of IP flow, incoming and outgoing ratio of IP packets, entropy of length in IP flow, entropy of protocols, and ratio of protocols. They used multi-class SVM to detect various DDoS attacks. Their work suffers two limitations. First, they only used emulated network traffic data to validate their method. Second, SVM is an offline learning algorithm, so it is not suitable for online DDoS detection.

Yu et al. (2008) proposed a lightweight and fast attack detection mechanism using the SVM-based hierarchical structure. The proposed method had two-levels: A one-class SVM was used at first level to detect attack traffic from normal traffic. Then at the second level, the attack traffic was classified into several attack types. They validated their method using simulated attack data. They utilized the features of Simple Network Management Protocol- Management Information Base (SNMP-MIB) data instead of raw data to detect DDoS traffic.

Yang et al. (2008) developed a support vector machine using a wavelet kernel (WSVM). They demonstrated that WSVM outperformed original SVM by about 4% less false positive rate while increasing the detection accuracy. It seems that the time complexity of WSVM is higher than SVM because WSWM tries different wavelet kernel functions and then compare and select the best kernel function. WSVM also suffers the similar shortcomings of the original SVM algorithm.

Cheng et al. (2009) proposed the concept of IAFV (IP Address Feature Value) to reflect the four features of DDoS attack flows including the abrupt traffic change, flow dis-symmetry, distributed source IP addresses, and concentrated target IP addresses. They used IAFV time series to describe the state change features of network flow. Finally, they employed the SVM classifier to detect DDoS attacks. Both time-series algorithm and SVM are not suitable for online detection.

Wagner et al. (2011) developed a new kernel function for calculating the similarities between NetFlow windows. The kernel function calculates the similarity between two windows (W1, W2) by summing up the similarity values between five features in two windows including (prefix, suffix-length of source/destination IP addresses and volume of the traffic. Then they used one-Class SVM with this new kernel function to detect different attacks in the real ISP traffic data. The one-class SVM classifier also has to be retrained when the network environment changes however its computation time is less than original SVM as it has a smaller number of support vectors.

Chitrakar and Huang (2014) proposed a DDoS detection algorithm named the candidate support vector-based incremental SVM. The incremental SVM is cost and time effective than original SVM because only support vectors are transferred to the next re-training process of SVM classifier while all other data samples are removed. They used Kyoto 2006+ dataset to validate their algorithm. Incremental SVM still have some shortcomings of original SVM such as requiring labeled data. Therefore, it is not entirely meet the requirements of online learning algorithm by Ahmed et al. (2016).

Sahi et al. (2017) proposed a classification mechanism for the prevention of TCP ping flood attacks in a cloud environment by detecting DDoS-attack traffic before sending incoming packets to the cloud service provider. Ostensibly, this is unrelated to the present research. However, if we consider only the detection of DDoS traffic in the internal network before the traffic is sent to the cloud, the aim of that study will be similar to our research question. Their feature vectors were the number of total transmitted packets and the number of packets from the same source to the same destination every 60 seconds. Four classification algorithms were employed, namely, Least Square-SVM (LS-SVM), *k*-nearest neighborhood, Naive-Bayes (NB), and multilayer perception, to distinguish DDoS and normal traffic. This system achieved the highest detection rate when LS-SVM was adopted. The proposed mechanism is an example of an old-fashioned offline attack detection scheme in a cloud environment.

Daneshgadeh et al. (2017) proposed a DDoS detection approach using the sequential minimal optimization (SMO) algorithm with the polynomial kernel function. Real network traffic and simulated DDoS traffic were used for validation. The synthetic minority over-sampling technique was employed for synthetically increasing the number of DDoS attacks in their dataset. Therefore, the perfect accuracy rate of 100% may have been achieved owing to over-fitting or a bias in the training and validation datasets.

Chen et al. (2018) developed a model based on the Random Forest (RF) algorithm to classify the traffic on top-level domain servers. The aim was to detect DDoS attacks on major recursive DNS servers. The authors applied a simple supervised learning algorithm that is not suitable for online attack detection.

2.4.4 *Works based on Information Theory Metrics*

Nychis et al. (2008) investigated the detection power of entropy-based analysis using flow-header features (IP addresses, ports, and flow-sizes) and behavioral features (e.g; the number of distinct destination/source IPs that each host communicates with). They demonstrated that precise detection of DDoS attacks needs more further investigation in addition to analyzing port and IP address distributions. Additionally, they suggested bi-directional analyses of the flows when it comes for computing traffic distribution in order to prevent bias.

Li et al. (2009) They used probability metrics such as source IP distribution to characterize DDoS and FE traffic. They mentioned that DDoS attacks are originated from a limited geographical area where bots are located, so the geographic diversity of source IPs follow Gaussian distribution. They also added that the source IP distribution of FEs is more disperse that follows a Poisson distribution. They also discussed the different trend of increase and decrease in the number of requests per second for DDoS and FE. During a DDoS attack a server encounters sudden increase/ decrease in the number of requests. Whereas, this change is smoother in FE. They used some information distance such as Jeffrey, Sibson, and Helinger distances to calculate the level of similarity among various DDoS and FE flows. They showed that Sibson distance is the best metric to separate DDoS from FE flows. Their experiment was performed on simulated data for both FE and DDoS traffic. The empirical threshold setting and requiring normal background network traffic are shortcomings of their algorithm.

Jun et al. (2014) proposed a detection mechanism using packet sampling and flow features. Two entropy-based and two statistical features were extracted, along with the corresponding thresholds. The algorithm initially compares the volume of the sample packet with a predefined threshold value and marks the traffic as suspicious if and only if the volume exceeds the threshold value. Then, other features are checked against their corresponding thresholds. If all features exceed the threshold values, a DDoS attack is detected. The reliance on a constant threshold is a drawback of this algorithm.

David and Thomas (2015) utilized fast entropy to detect DDoS-attack traffic. The authors claimed that the flow count entropy severely decreases in the case of attack flows, and it is stable otherwise. To detect DDoS attacks, a pre-defined threshold was compared with the difference between the flow count entropy and the mean entropy at each instant in the same time interval. The threshold was updated according to the packet traffic condition. If the fast entropy was 1.5 times as high as the mean flow count, then the threshold was increased by one. If the fast entropy was one half of the mean flow count, then the threshold was decreased by one. However, this detection method can be bypassed if the attacker knows the fixed update rule for the threshold value.

Bhuyan et al. (2015a) used several Generalized Entropy (GE) and Generalized Information Distance (GID) metrics with different α-order to distinguish DDoS attacks with different rates, where α refers to the value of α in Renyi's α-entropy (Rényi, 1965). Experiments demonstrated that using GID and GE with higher α values increases the dissimilarity between normal and both LR-DDoS and HR-DDoS traffic. The low computing overhead of these metrics facilitates real-time application. However, threshold setting is empirical and could be biased.

Behal and Kumar (2017) reported the significant increase in the entropy values of source IP addresses and source ports during a DDoS attack. They also reported a substantial decrease in the entropy values of destination IP addresses and destination ports for a DDoS attack. Generalized information distance metrics such as Reny, Sibson, Jeffery, Kullaback-leibler, Bhattacharyya Hellinger have been used to discriminate DDoS attacks from FE traffic. They also demonstrated that the Generalized Entropy (GE) and Generalized Information Distance (GID) metrics with higher order of alpha could discriminate legitimate, DDoS and FE traffic in significant manner than Shannon entropy and KullbackLeibler distance.

Behal et al. (2018) proposed a detection method called D-FACE to differentiate legitimate, LR-DDoS, HR-DDoS, and FE traffic. This algorithm compares the source IP entropy of normal traffic flow and current incoming traffic in each time window. The entropy difference is called Information Distance (ID) and is used as the detection metric. Two thresholds were defined according to the baseline behavior of a network without attacks. The major issue with this algorithm is defining normal network traffic in a continually changing environment.

2.4.5 Works based on hyprid approches

SVM is one of the powerful and well-known non-linear (using kernel function), non-parametric classification technique, which already showed good results in the cyber-attack detection. SVM were employed in combination with other methods by many researchers to detect anomalies and DDoS attacks in the network traffic data. All studies reported promising results to detect DDoS attacks. However, none of these hybrid algorithms are suitable for online DDoS detection because the original SVM classifier need to be re-trained sequentially and from scratch using labeled data when the network traffic changes.

Gan et al. (2013), Chen et al. (2009), Horng et al. (2011), Kuang et al. (2014) investigated the effect of SVM with Partial Least Square (PLS), Extended Learning Machine (ELM), Rough Set Theory (RST), Balanced Iterative Reducing and Clustering Hierarchies (BIRCH), Kernel Principal Component Analysis (KPCA) to classify DoS attacks in the KDD CUP'99 dataset. Agarwal and Mittal (2012) and Khan et al. (2007) have utilized SVM with Dynamically Growing Self-Organizing Tree (DGSOT) clustering algorithm and Shannon entropy respectively to detect DoS attacks in the DARPA dataset.

Gogoi et al. (2013) proposed a multi-level hybrid intrusion detection method based on supervised, unsupervised and outlier-detection algorithms to classify different types of attacks such as DoS/DDoS, Probe, R2L, U2R, and normal traffic. The classifier in the first level of this algorithm is responsible for detecting DDoS attacks. The classifier builds a set of representative clusters (DoS/DDoS, Probe, and rest) from the available labeled training data. Subsequently, the algorithm measures the similarity between each unlabeled test data and the predefined clusters and insert unlabeled data in the corresponding cluster. Finally, unlabeled data get the label of the clusters in which they are added. The performance of the algorithm is highly dependent on the availability and significant of initially labeled clusters.

Qin et al. (2015) defined a feature vector based on entropies of five different features of traffic flows and one TCP flag feature. Then they used the common k-means clustering algorithm to model normal patterns of the network and determine the detection threshold. The cluster number setting, and construction of the normal flow profile are challenging aspects of their approach.

Fernandes Jr et al. (2016) proposed a network anomaly detection algorithm based on Principal Component Analysis (PCA), ant colony optimization, and Dynamic Time Warping (DTW) to detect DoS, DDoS, port scan, and FE attacks. Three quantitative and four qualitative IP flow attributes were analyzed. The results demonstrated that the algorithm is successful in detecting different types of anomalies in the network traffic without attack type classification. However, the algorithm requires prior knowledge of the normal network traffic behavior for at least one day.

Hoque et al. (2017) defined a new correlation measure referred to as NaHiD for the distance between two feature vectors. These vectors were defined using three features: the entropy of source IP addresses, variation index of source IP addresses, and packet rate per second. The network monitor initially calculates the normal profile and threshold value for the detection algorithm. An attack is detected when the computed correlation value is smaller than a user-defined threshold. The NaHiD metric can be implemented in software as well as in hardware using field programmable gate arrays. Initially, a network monitor calculates the normal profile of the training dataset and determines the optimal threshold value that provides the highest classification accuracy for the attack and the normal instances in the training set. These values are stored in the profile log dataset. The normal profile and threshold value are updated incrementally and dynamically based on the previously stored values. The algorithm can operate in online mode and adapt to a changing traffic pattern; however, it requires the normal traffic profile in advance.

Idhammad et al. (2018) proposed a semi-supervised DDoS detection approach based on entropy estimation, co-clustering, information gain ratio, and extra-trees. The entropy of the header features was measured and analyzed using different time-based sliding windows. Then, a co-clustering algorithm was used to split network traffic into three clusters (normal traffic, DDoS traffic, and normal as well as DDoS traffic) based on the entropy features. Subsequently, the information gain ratio was measured for each cluster and computed with the average entropy of the network header features in the current time window. Finally, clusters with high information gain ratio were considered suspicious and the extra-trees algorithm was used to separate DDoS attack traffic from other abnormal or normal traffic. This algorithm is not suitable for online detection, as it requires labeled data and handling several thresholds and parameters for extra-trees. Moreover, it relies on a supervised classification tree, which is not suitable for online learning.

Daneshgadeh et al. (2018) proposed a hybrid method using the Kernel based Online Anomaly Detection (KOAD) algorithm with pre-defined threshold settings and the Mahalanobis distance metric to detect DDoS attacks. Normal and abnormal datasets were generated based on the KOAD algorithm. Subsequently, the Mahalanobis distance between abnormal data points and the normal traffic distribution was measured. This distance was statistically evaluated by means of the Chi-square test. Simulated data were used for validation.

Gu et al. (2019) proposed a semi-supervised k-means algorithm using hybrid feature selection to detect DDoS attacks. Initially, a set of nine candidate features was defined, where eight of them were entropy-based. Then, a hybrid feature selection method was applied to rank the candidate features and select the most effective one. A semi-supervised k-means algorithm was employed for model training and testing, and a small subset of labeled data was used to facilitate the selection of the initial center points for the k-means algorithm and resolve the outlier and local optimum issues. Subsequently, the classical k-means algorithm was applied to determine the similarity between unlabeled data and initial clusters. This algorithm requires labeled data for initialization, which limits the applicability of the algorithm in an unknown real-time environment. Additionally, a set of best features was selected for each dataset separately based on performance analysis (recall and false positive rate). However, calculating these performance metrics is not practical if there are no labeled data for unknown network traffic.

Daneshgadeh et al. (2019a) used Shannon entropy with the KOAD algorithm for online detection of DDoS and FE traffic. Subsequently, the Mahalanobis distance was used to differentiate between various types of DDoS attacks from FE traffic. This study did not use a benchmark validation dataset and provided neither a systematic method for obtaining the Mahalanobis distance nor a guideline for selecting thresholds for the KOAD algorithm.

Daneshgadeh et al. (2019b) proposed a hybrid algorithm to distinguish DDoS attacks from FE traffic. The algorithm detected abnormal network data points using the KOAD algorithm (Ahmed et al., 2007a, Ahmed et al., 2007b), and then used an SVM classifier to separate DDoS attacks from FE traffic. Simulated DDoS attack, real FE, and normal data were used to evaluate the accuracy of the algorithm. The predefined threshold setting of the KOAD algorithm, offline training of the SVM classifier, and application of simulated validation instrument are the drawbacks of this framework.

CHAPTER 3

RESEARCH METHODOLOGY

This chapter presents descriptions of the data used in this thesis. Subsequently, the research design and proposed scheme are given.

3.1. Dataset

In the experiments, the publicly available CICIDS2017 dataset, which contains normal and the most up-to-date common attacks was used. CICIDS2017 is a realistic IDS dataset because it is based on B-Profile and M-Profile components. The B-Profile is responsible for profiling the abstract behavior of human interactions and generating realistic benign background traffic (Sharafaldin et al., 2018a). The M-Profile is used to describe the details of attack scenarios. The dataset includes the abstract behavior of 25 users based on the HTTP, HTTPS, FTP, SSH, and e-mail protocols.

Sharafaldin et al. (2018a) created a comprehensive testbed which included two separated networks: Attack-Network and Victim-Network.

Figure 1 demonstrates the underlying network infrastructure of the CICIDS2017 dataset.

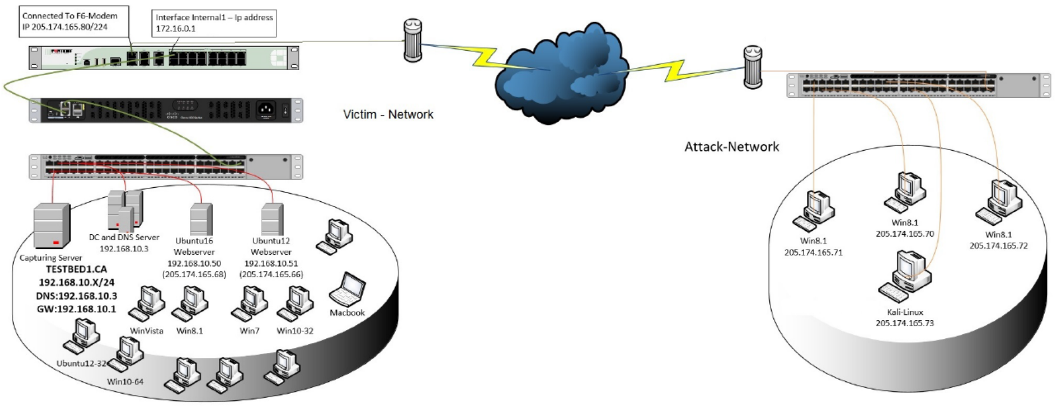


Figure 1 Testbed architecture of CICIDS2017 dataset (Sharafaldin et al. (2018b))

Sharafaldin et al. (2018a) defined 11 criteria for building a reliable benchmark dataset as following:

- **Complete Network Configuration**: The Victim-Network is a highly secure infrastructure and includes different networking tools such as firewalls, switches, and routers. Victim-Network composites of various operating systems such as Windows, Ubuntu and Mac OS X. The Attack-Network is wholly separated from the Victim-Network and consists of one router, one switch and four PCs with public IP addresses. The Kali and Windows 8.1 are available operating systems on mentioned four PCs.

- **Complete Traffic**: The testbed includes one user profiling agent,12 different machines in Victim-Network and all attacks are real.

- **Labelled Dataset**: All flows in the dataset are labeled as benign or the name of the attack.

- **Complete Interaction**: The dataset includes all communication between Victim-Network and Attack-Network as well as Internet communications.

- **Complete Capture**: The dataset covers all traffic in the testbed using a mirror port such as a tapping system.

- **Available Protocols**: The dataset has all standard protocols such as HTTP, HTTPS, FTP, SSH and email protocols.

- **Attack Diversity**: The dataset includes the most common attacks, including brute force, DoS, DDoS, data infiltration, Heart-bleed, Bot, port scan and web-based attacks.

- **Anonymity**: The payloads of the packets were removed because of privacy concerns.

- **Heterogeneity**: The dataset is heterogeneous because, all the network traffic is captured from the main switch, memory dump and system alarms of all victim machines during the attacks.

- **Feature Set**: The dataset includes 83 network traffic features, which were extracted using the CICFlowMeter software package (CICFLOWMETER)

- **Meta-Data**: Information related to time, attacks, flows and labels are entirely explained.

Sharafaldin et al. (2018a) defined the equation (3.1) to measure the reliability of their CICIDS2017 dataset.

$$\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{w_i}\left(\sum_{\mathrm{j}=1}^{\mathrm{m}} \mathrm{v_j} \times \mathrm{F_j}\right) \tag{3.1}$$

Where W is a weight of each feature (11 criteria), V is the coefficient of each sub-factor, F is a binary value which demonstrates the appearance/absence of a specific factor and sub-factor in the dataset, n is the number of features and m is the number of coefficients for each factor. The CICIDS2017 dataset achieved a score of 1, whereas KDD CUP'99 (KDD, 1999) achieved only 0.56. Therefore, using this dataset in the present is justifiable.

3.2. Feature Extraction

The entire CICIDS2017 dataset was divided into eight .csv files. For the experiments in this thesis, files that included DDoS and normal traffic flows (July 03 and 07, 2017) were used. Subsequently, the feature vectors were generated by aggregating network flows per minute.

Yu et al. (2012) analyzed DDoS attack network traffic and revealed that DDoS flows are more similar than normal flows. Attackers usually prefer to send general commands to bots instead of sending a specialized command to each bot. Therefore, the randomness degree of the attributes (such as number of source IPs, number of sent/received bytes, duration, etc.) in DDoS flows is less than FE flows. Therefore, the information theory-based detection metrics have gained popularity in the DDoS attack detection literature. This research incorporates the advantages of using entropy to the proposed framework by measuring the randomness degree of flows using Shannon entropy. A feature vector was constructed based on the Shannon entropy of Source/Destination IP addresses/Ports.

The Shannon entropy of a discrete distribution is a measure of uncertainty or randomness of a single random sample in a separate distribution based on the Boltzmann entropy of classical statistical mechanics (Rao et al., 2004). The entropy of random sample x is defined as:

$$H(X) = -\sum_{t=1}^{N} p(x_t) \cdot log_2\, p(x_t) \tag{3.2}$$

Where $X = \{x_1, x_2, x_3, \cdots, x_N\}$ and the probability distribution $P = \{p_1, p_2, p_3, \cdots, p_N\}$.

The value of Shannon entropy is always positive and equal to zero if and only if it is an individual event. Additionally, an increase in the number of independent components results in increasing the value of entropy and vice versa. As a result, the entropy values of Source IP addresses/Ports increase dramatically during DDoS attacks as the number of independent source IP addresses increases and the entropy values of Source IP addresses/Ports increase sharply. The entropy-based feature vector was constructed as the following:

$$\overrightarrow{\mathrm{F}_1} = \begin{pmatrix} Time\_Interval, Source\_IP\_Entropy, Destination\_IP\_Entropy, \\ Source\_Port\_Entropy, Destination\_Port\_Entropy \end{pmatrix}$$

Time_Interval represents the time stamp of the flow in minutes of the day. For example, Time_Interval=400 implies that the flow is related to the time 6:4 AM. Finally, 448 input vectors were obtained from 480,745 flows, where 21 of those vectors were related to DDoS attacks. Additionally, another feature vector was constructed and named statistical-based feature vector. It utilized the best candidate features to detect DDoS attacks by Sharafaldin et al. (2018b).

$$\overrightarrow{\mathrm{F}_2} = \begin{pmatrix} Time\_Interval, Flows\_IAT\_Std, Total\_BPacket\_Len\_Std, \\ Flows\_Duration, Total\_Avg\_Packet\_Size \end{pmatrix}$$

Table II shows the list of abbreviations for the features.

Table II List of abbreviations for features.

Feature Names	Feature Abbreviations
Flows_IAT_Std	FsIAT_Std
Total_BPacket_Len_Std	TBP_Len_Std
Flows_Duration	FsD
Flow_Duration	FD
Total_Avg_Packet_Size	TAvg_PS
Avg_Packet_Size	Avg_PS
Flow_IAT_Std	FIAT_Std
Flow_IAT_Mean	FIAT_Mean
BPacket_Len_Std	BP_Len_Std
BPacket_Len_Mean	BP_Len_Mean

The feature vector ($\overrightarrow{\mathrm{F}_2}$) has four attributes as follows:

- Flows_IAT_Std denotes the standard deviation of the time between packets that are sent in either direction in 1 minute.

- Total_BPacket_Len_Std denotes the standard deviation of packet length in the backward direction in 1 minute.
- Flows_Duration denotes the flow duration.
- Total_Avg_Packet_Size denotes the average packet size in 1 minutes.

The CICID2017 dataset provides the statistical attributes of traffic per flow. The flow-based features were converted into time-based feature as shown in equations (3.3) to (3.6).

$$FsD = \sum_{i=1}^{N} FD_i, \qquad (3.3)$$

where N is the number of flows in the corresponding $Time\_Interval$.

$$TAvg\_PS = \frac{\sum_{i=1}^{n}(n_i \times Avg\_PS_i)}{\sum_{i=1}^{n} n_i}, \qquad (3.4)$$

where n is the total number of packets in the corresponding $Time\_Interval$ and n_i is the number of packets in each flow.

$$FsIAT\_Std = \frac{\sum_{i=1}^{n}(n_i \times FIAT\_Std_i^2) + \sum_{i=1}^{n} n_i \times (FIAT\_Mean_i - FIAT\_Mean)^2}{\sum_{i=1}^{n} n_i}, \qquad (3.5)$$

where $FIAT\_Mean_I$ is the mean of the inter-arrival time for each flow and $FIAT\_Mean$ is the grand mean of all inter-arrival times for the corresponding $Time\_Interval$.

$$TBP\_Len_{Std} = \frac{\sum_{i=1}^{n}(n_i \times BP\_Len\_Std_i^2) + \sum_{i=1}^{n} n_i \times (BP\_Len\_Mean_i - BP\_Len\_Mean)^2}{\sum_{i=1}^{n} n_i}, \qquad (3.6)$$

where $BP\_Len\_Mean_i$ is the mean of the backward packet length and BP_Len_Mean is the grand mean of all backward packet lengths for the corresponding $Time\_Interval$.

3.3. Reseach Design

Nesselroade and Cattell (2013) defined experimental research design as “a recording of observations, quantitative or qualitative, made by defined and recorded operations and in defined conditions, followed by examination of the data, by appropriate statistical and mathematical rules, for the existence of significant relations."

This research completely follows the definition by Nesselroade and Cattell (2013). Therefore, it can be seen as entirely experimental research. According to Asadullah (2011), empirical research is commonly accepted research design in different areas of knowledge.

3.4. Proposed Architecture

The proposed framework combines the capabilities of Shannon entropy, k-means, KOAD, the Mahalanobis distance, and the Chi-square test for the online detection of DDoS attacks. According to Xiang et al. (2011), the entropy values are relatively uniform when the network traffic is normal, but the entropy values of one or more features would increase/drop significantly during the DDoS attacks. Therefore, Shannon entropy is utilized for feature construction. DDoS attack is regarded as a collective anomaly in the literature. Consequently, the online anomaly detection algorithm is suitable for spotting cumulative anomalies. The KOAD algorithm is selected because, in addition to network anomaly detection, it has already exhibited promising performance in diverse anomaly detection areas, such as medical monitoring (Ahmed et al., 2016b), surveillance systems (Anika et al., 2017, Ahmed et al., 2010) and image processing (Ahmed et al., 2013, Ahmed et al., 2017, Ahmed et al., 2014). Moreover, the KOAD algorithm is useful for modeling of normal network traffic behavior. However, the updated version of the original KOAD algorithm named E-KOAD is employed in this thesis in order to overcome some shortcomings of the original KOAD. The E-KOAD depreciate from KOAD in three aspects, including automated threshold settings, automated standard deviation setting for RBF kernel, and replacement of the "Usefulness Test" with "Utility Test".

The number of fully online anomaly detection algorithms are limited in the literature. Some researchers have used the terms "online" and "real-time" interchangeably in the literature, but there is a significant difference between them. An ideal online machine learning algorithm should be able to update itself and adapt to a frequently changing environment, in addition to operating immediately while ingesting one observation at a time.

Online algorithms have the ability to learn from a newly arriving data instance, without re-training the whole data obtained to-date from initiation. As online algorithms involve real-time operations, the computational and storage complexities of the algorithms (both in terms of time and memory) is required to not grow with time as the size of the whole (to-date) dataset grows, and preferably be small.

The KOAD algorithm meets all the requirements of real-world anomaly detection set forth by Ahmed et al. (2016a) as following:

- The KOAD algorithm instantly decides about an incoming datapoint by issuing an “Orange”/”Red” alarm. Moreover, it saves “Orange” alarms for further investigation to ensure that they correspond to anomalies and not to alterations in the trend of the traffic.

- The algorithm learns continuously and does not require storing the entire stream of data. All updates are performed recursively when a data point is added to or deleted from the *Dictionary*. Therefore, there is no need to store the entire stream.

- The algorithm runs in a completely unsupervised manner without requiring labeled data. The original KOAD algorithm requires the experimental setting of thresholds. However, the proposed algorithm overcomes this limitation.

- The algorithm can adapt to dynamic environments by using a *Dictionary* and a “Usefulness Test”.

- The algorithm has a "Red2" alarm to detect anomalies as early as possible. When the deviation from the normal traffic of the network is not significant to ensure that the incoming point is abnormal, the algorithm provides a calculated prediction by considering subsequent data. Nevertheless, the algorithm can be forced to make decisions as soon as it receives the traffic flow by setting the lag-time *(l)* to zero.

- The algorithm attempts to minimize false positives and false negatives by postponing the decision on less suspicious traffic flows.

The Mahalanobis distance is utilized for separating DDoS attacks from other abnormal traffic data points, which were initially detected by E-KOAD. Mahalanobis distance is used for different purposes in the literature, including similarity measurement, outlier detection, calibration samples selection and examination of representativity between two data sets (De Maesschalck et al., 2000). It is also used for detecting anomalies in network traffic (Santiago-Paz et al., 2012; Bayarjargal and Cho, 2014).

Semerci et al. (2018) obtained promising DDoS detection results in SIP networks by using a novel adaptive real-time change-point model that tracks the changes in the Mahalanobis distance. It was assumed that employing the Mahalanobis distance metric is most likely to be useful in detecting DDoS attacks in data networks as well.

Mahalanobis distance fits the data into uncorrelated and unit-variance Gaussian variables. If it is assumed that Mahalanobis distance measures the difference between each incoming feature vector and the mean vector of normal feature vectors, then the Mahalanobis distance values follow the Chi-squared distribution with d-degrees of freedom (Semerci et al., 2018). Therefore, the Chi-square test can be employed for evaluating the Mahalanobis distance values.

The proposed algorithm initially uses the entropy-based feature vectors to calculate the δ values of the KOAD for one hour. After one hour, the *k*-means algorithm is utilized to find the optimal threshold values for the kernel-based anomaly detection algorithm. Then the system continues its operation using new thresholds. Whenever the E-KOAD issues a "Red" or "Red2" alarm, the corresponding input data will be added to the suspicious dataset. Similarly, the input data related to *Dictionary*.

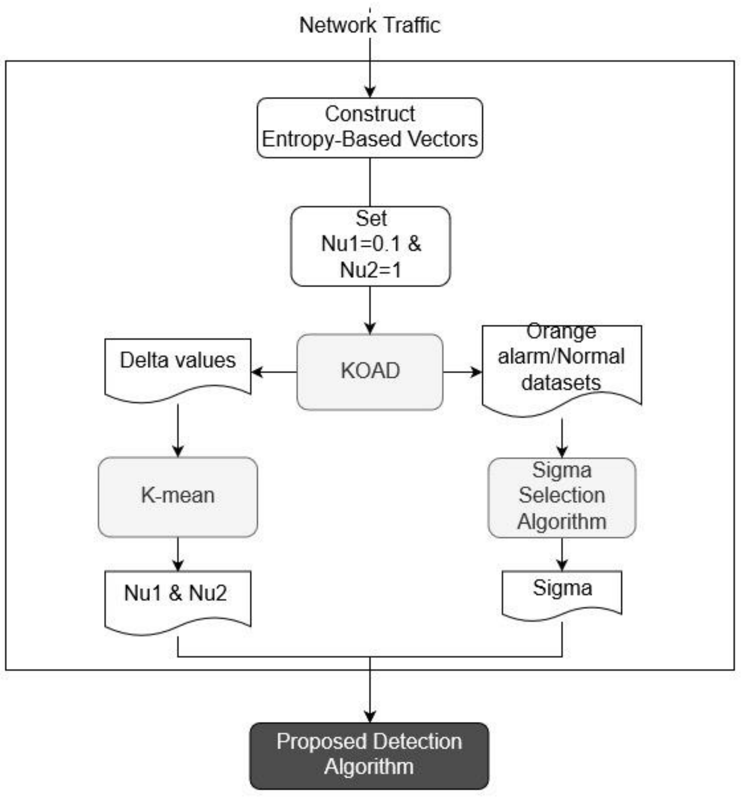

Network Traffic
Construct Entropy-Based Vectors
Set Nu1=0.1 & Nu2=1
Delta values
KOAD
Orange alarm/Normal datasets
K-mean
Sigma Selection Algorithm
Nu1 & Nu2
Sigma
Proposed Detection Algorithm

Figure 2 Initialization of proposed algorithm

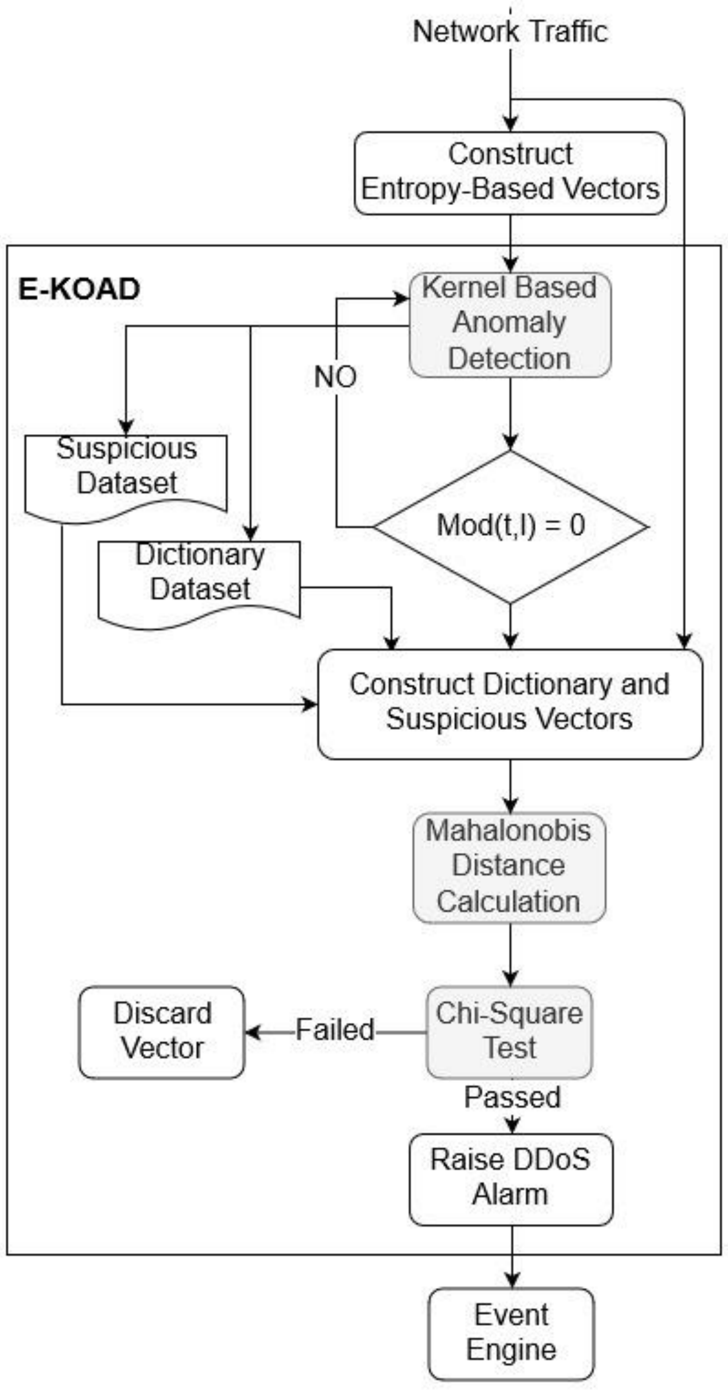


Figure 3 Proposed architecture of E-KOAD

CHAPTER 4

ALGORITHMIC FONDATION

This chapter presents the detailed mathematical background of the proposed scheme including KOAD, k-means, Mahalanobis distance and Chi-square test.

4.1. Kernal Function

Algorithms based on the so-called "kernel trick" involve using a kernel function that maps the input data onto a feature space of a much higher dimension (Scholkopf and Smola, 2001). This counterintuitive operation is performed owing to the expectation that points depicting similar behavior should form more pronounced clusters in the richer feature space. A suitable kernel function, when applied to a pair of input vectors, may be interpreted as an inner product in the feature space (Scholkopf and Smola, 2001). This subsequently allows inner products in the feature space (inner products of the feature vectors) to be computed without explicit knowledge of the feature vectors themselves, by only evaluating the kernel function:

$$k(x_i, x_j) = < \phi(x_i), \phi(x_j) >, \tag{4.1}$$

Where x_i, x_j denote the input vectors and Φ represents the mapping onto the feature space. Using kernel functions thus allows a simple comparison of higher-order statistics between the input vectors.

Subsequently, a kernel matrix is defined as $K := (k(x_i,x_j))_{i=j=1}^{n}$, where x_i is a set of observation $x_i \in X \; and \; i = \{1,2,\cdots,n\}$. The following are some popular kernels (Haasdonk and Burkhardt, 2007):

- Linear Kernel: $k(x, y)_{Linear} = x^T y + c$ (4.2)
- Polynomial Kernel of degree p: $k(x, y)_{Polynomial} = (ax^T y + c)^d$ (4.3)
- Gaussian/Radial Kernel: $k(x, y)_{RBF} = exp\left(\frac{-\|x-y\|^2}{2\,\sigma^2}\right)$ (4.4)
- Negative Kernel: $k(x, y)_{ne} = -\|x - y\|^{\beta}, \beta \in [0, 2]$ (4.5)

Figure 4 demonstrate the classification of circular-separable data based on different kernel functions.

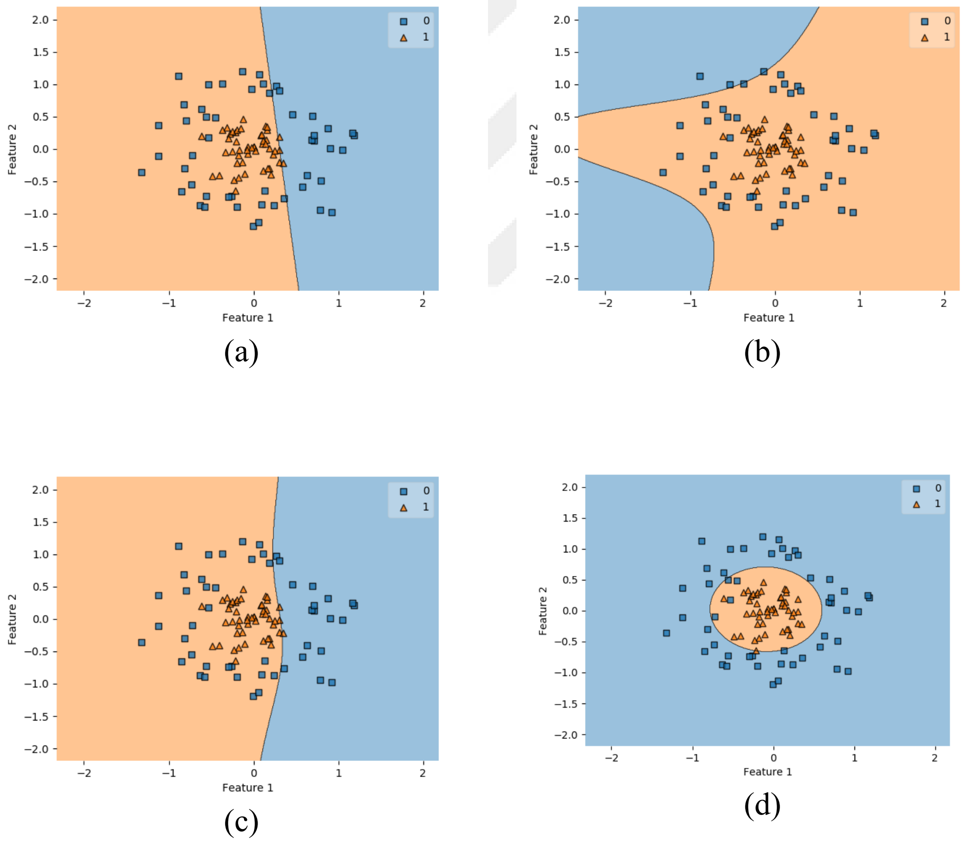


Figure 4 Decision boundries of SVM classification on 2-dimensional data using (a) linear kernel, (b) 2-polynomial kernel, (c) sigmioid kernel and (d) RBF kernel.

SVM, Kernel Principal Component Analysis (KPCA) and kernel regression are examples of offline algorithms that use kernel functions. Linear and non-linear classifiers employ different kernel methods. The Radial Basis Function (RBF) kernel method is suitable for data samples which are dependent non-linearly because it maps samples onto higher-dimensional space (d > 2) in a non-linear fashion.

RBF is the most widely used type of kernel function because it has a localized and finite response along the entire x-axis. Therefore, it is a general-purpose kernel and can be used when there is no prior knowledge about the relationships among data points (DataFlair Team, 2018).

A non-linear, kernel-based least squares algorithm was initially introduced in 2004 by Engel et al., (2004). Their algorithm took advantage of the kernel trick to perform linear regression in high dimensional feature space to recursively calculate minimum mean-squared- error solution to non-linear least-squares problems.

The Kernel Recursive Least Square (KRLS) algorithm tries to solve the problem of regularization and computational cost using online constructive sparsification. This sparsification method only selects data samples that cannot be represented as an appropriate linear combination of selected samples. The KRLS algorithm incrementally builds a *Dictionary* (basis) of approximate linearly independent samples.

Ahmed et al., (2007a), proposed a prolonged variant of the kernel-based least square algorithm, which they named the Kernel-based Online Anomaly Detection (KOAD) algorithm. The KOAD algorithm incorporates two thresholds of approximate linear independence, includes exponential forgetting to reduce the importance of past observations gradually, and allows the deletion of previous *Dictionary* members to enable the basis set to remain current dynamically. These features were absent in the foundation KRLS algorithm of Engel et al. (2004).

The postulate of Ahmed et al. is that if the multivariate data points $\{x_t\}_{t=1}^{T}$show normal behavior in the input space, then it is expected that the corresponding feature vectors $\{\Phi(x_t)\}_{t=1}^{T}$will construct a cluster. Consequently, the explanation of normality region should be possible using an almost small *Dictionary* of approximately linearly independent elements ($\{\Phi(\tilde{x}_j)\}_{j=1}^{m}$) in the feature space. If the projection error δ_t conciliates the equation (4.6), feature vector $\varphi(x_t)$ is issued to be linearly dependent on $\{\Phi(\tilde{x}_j)\}_{j=1}^{m}$, with approximation threshold ν.

$$\delta_t = \min_a \left\| \sum_{j=1}^{m} a_j \times \Phi(\tilde{x}_j) - \Phi(x_t) \right\|^2 < \nu, \tag{4.6}$$

where $a = \{a_j\}_{j=1}^{m}$ is the optimal coefficient vector. Here $\{\Phi(\tilde{x}_j)\}_{j=1}^{m}$ represents those $\{x_t\}_{t=1}^{T}$ that are entered the *Dictionary*. It is expected that the size of the *Dictionary* (m) will be considerably less than total time steps (T), which results in CPU usage and memory savings.
The equation (4.7) involves an L2 norm (the distance of the vector from the origin of the hyper-space vector), which could be demonstrated exclusively in the form of the inner products of $\Phi(\tilde{x}_j)$ and $\Phi(x_t)$. As a result, the kernel function could be used to evaluate it without requiring an exact knowledge of the feature vectors. Therefore, equation can be presented as:

$$\delta_t = \min_a \left\| a_t^T \tilde{K}_{t-1} a_t - 2\tilde{a}_t \tilde{k}_{t-1}(x_t) + k(x_t, x_t) \right\|^2 < \nu, \tag{4.7}$$

where $[\tilde{K}_{t-1}]_{i,j} = k(\tilde{x}_i, \tilde{x}_j)$ and $[\tilde{K}_{t-1}(x_t)]_j = k(\tilde{x}_t, x_t)$ for $i, j = 1,2,\cdots, m_{t-1}$.

The a_t in equation (4.8) is said to be the optimum sparsification coefficient vector and is used to minimize δ_t.

$$a_t = \tilde{K}_{t-1}^{-1} \tilde{k}_{t-1}(x_t) \tag{4.8}$$

Therefore, the error (δ_t) is simplified into:

$$\delta_t = k_{tt} - \tilde{K}_{t-1}(x_t)^T \tilde{a}_t \tag{4.9}$$

The KOAD algorithm operates at each time step t on a measurement vector x_t. It begins by evaluating the error δ_t in projecting the arriving observation x_t onto the current *Dictionary* (in the feature domain). This error measure δ_t is then compared with two thresholds ν_1 and ν_2, where $\nu_1 < \nu_2$.
If $\delta_t < \nu_1$, KOAD speculates that x_t is significantly dependent on the *Dictionary* members in linearly manner, and thus represents normal behavior.
If $\delta_t > \nu_2$, KOAD speculates that x_t is far away from the normal behavior of the system and immediately raises a “Red1” alarm to flag an anomaly.
If $\nu_1 < \delta_t < \nu_2$, KOAD speculates that x_t is not sufficiently linearly dependent on the *Dictionary* to be considered as a normal event. It might be caused by an anomaly, or it might be resulted because of a change in the normal behavior of the system (expansion or migration of the space of normality). In this situation, KOAD immediately signals the

existence of the abnormal input vector by raising an “Orange” alarm, then it keeps track of subsequent arrival inputs for the next *l* time steps and it investigates the contribution of the abnormal input vector x_t in explaining of the mentioned subsequent arrival inputs.
If the *Dictionary* element x_{t-l} is able to explain a noticeable number of input vectors between time steps $t - l$ to t, it should be kept in the *Dictionary*.
KOAD algorithm uses “Usefulness Test” to resolve the orange alarm. The usefulness of x_{t-l} is measured by equation (4.10).

$$\left[\sum_{i=t+1}^{t+l} \mathbb{I}\big(k(x_t, x_i)\big) > d\right] > \epsilon L \tag{4.10}$$

Particularly, at timestep t+ *l*, the KOAD performs a “Usefulness Test” and checks if a noticeable number of kernel values between x_{t-l} and *l* subsequent input vectors are more than threshold d. Finally, at time t+*l*, KOAD lowers “Orange” alarm into “Green” if the “Usefulness Test” is passed meaning normal behavior of x_{t-l} otherwise it elevates the “Orange” alarm to “Red2” alarm indicating anomalous observation and it removes the from the Dictionary. x_{t-l} otherwise it elevates the “Orange” alarm to “Red2” alarm indicating anomalous observation and it removes the x_{t-l} from the *Dictionary*.

Finally, at time t- *l*, KOAD lowers “Orange” alarm into “Green” meaning normal behavior of x_t or elevates the “Orange” alarm to “Red2” alarm indicating anomalous observation. Figure 5 presents the pseudocode for the KOAD algorithm.
Following sub-sections presents the parameters and attributes of the algorithm and the ways that they are set. Some numeric examples also are provided to visualize the formation of the various attributes in the algorithm.

4.2.1. Threshold Setting

The original KOAD algorithm does not provide automatic setting of the thresholds (ν_1, ν_2). Ahmed et al. (2007a) investigated different pairs of ν_1 and ν_2, and they demonstrated that the optimal setting varies for different metrics. They recommend that researchers can run the algorithm over a training dataset in a supervised fashion with pre-known anomalies and then, set the threshold values that result in a tolerable trade-off between True Positive Rate (TPR) and False Positive Rate (FPR). However, the original KOAD algorithm was improved by proposing a systematic and automated way to select optimal threshold values (Please refer to chapter 5, section 5.1 for detailed information).

```
Set thresholds: $\nu_1, \nu_2$ ;
for $t = 1, 2, \ldots$ do
    Data: $(x_t, y_t)$
    /* Evaluate current measurement */
    Compute projection error $\delta_t$ for $x_t$ using $\mathcal{D}_t$ ;
    if $\delta_t > \nu_2$ then
        Raise Red1 Alarm ;
    endif
    if $\delta_t > \nu_1$ then
        Raise Orange Alarm ;
        Store $x_t$ in $\Theta$ ;
    endif
    /* Process previous orange alarm */
    if Orange Alarm($x_{t-\ell}$) then
        Re-evaluate projection error $\delta$ for $x_{t-\ell}$ using $\mathcal{D}_t$
        if $\delta > \nu_1$ then
            Evaluate usefulness of $x_{t-\ell}$ over previous $\ell$
            measurements ;
            if NOT useful then
                Raise Red2 Alarm($x_{t-\ell}$) ;
            else
                Add $x_{t-\ell}$ to dictionary $\mathcal{D}$ ;
                Lower Orange Alarm($x_{t-\ell}$) ;
            endif
        else
            Lower Orange Alarm($x_{t-\ell}$) ;
        endif
        Remove $\Theta\{1\}$ ;
    endif
    /* Remove obsolete elements */
    Evaluate usefulness of each dictionary element over
    previous $L$ measurements;
    Remove any useless element from dictionary $D$;
endfor
```

Figure 5 Outline of KOAD algorithm (Ahmed et al., 2007a)

4.2.2. Parameter Setting (l, ε, L, d, γ)

- ***l*** is a lag-time parameter for resolving the "Orange" alarm. When $\nu_1 < \delta_t < \nu_2$ the algorithm waits for *l* time steps and then decides whether to elevate the existing "Orange" alarm to "Red", or to add the corresponding input vector to the *Dictionary*. Adding the vector to the *Dictionary* indicates a change in the basis for the sphere of normality. *l* should be selected in a manner that balances the waiting time for detecting the anomaly and the false positive rate. If it is too large, it will violate the principle of online detection. If it is too small, there will not be enough time to make an intelligent decision for suspicious cases.

- $\boldsymbol{\varepsilon}$ is also a parameter for resolving the "Orange" alarm. ε is a real number between zero and one ($\varepsilon \in (0,1)$). It determines what fraction of input vectors should lie within the region of usefulness. The effect of epsilon on algorithm should be run with different values of ε, and the impact of ε on performance should be investigated. It should be selected based on the user's sensitivity tolerance.
- **L** is a parameter for dropping obsolete elements. It determines the time when obsolete (useless) elements should be removed from the *Dictionary*. L should be set based on the long-term stationarity of the application data sphere.
- ***d*** is also a parameter for dropping obsolete elements. It determines the amount of closeness between a *Dictionary* element and an input vector to consider a *Dictionary* element as useful. In other words, it defines the region of usefulness. The value of d should be selected based on the kernel type and value because the kernel implicitly defines a distance measure.
- $\boldsymbol{\gamma}$ is the forgetting factor. The algorithm gradually and exponentially disregards past data. Parameter γ is a time-based weight which is systematically applied to past observations. A value of $\gamma = 1$ means that the most recent and previous input vectors have equal importance. The forgetting factor is set ($0 < \gamma < n\,, n = 1,\ 2, 3,\cdots$) for the *n*th most-recent observation, meaning that recent events are gradually more important than past events.

4.2.3. *Initialization Phase*

- $D = \{x_1\}$: the first input vector is added to the *Dictionary* at $t = 1$.
- $m_1 = 1$: the number of elements in *Dictionary* (in correspondence with preceding step) is one.

- $\widetilde{\mathrm{K}}_1 = [\mathrm{k}_{11}]$: the kernel matrix is set to the kernel value of the (as of now) sole element of the *Dictionary* with itself. In general, $\widetilde{K}_t$ keeps track of kernel values among the members of the *Dictionary* at time t.
    - ✓ Example: assume that *t* = 5 and $D = \{x_1, x_3, x_5\}$, $x_1 = dic_1,\ x_3 = dic_2\ and\ x_5 = dic_3$. That is, the 1st, 3rd and 5th arriving samples have been entered into the *Dictionary*, with a total of five time steps having elapsed since the algorithm began running, and thereby constitute the *Dictionary* composition at time *t* = 5. Then $\widetilde{K}_5 =$
$$\begin{bmatrix} k(dic_1, dic_1) = 1 & k(dic_1, dic_2) & k(dic_1, dic_3) \\ k(dic_2, dic_1) & k(dic_2, dic_2) = 1 & k(dic_2, dic_3) \\ k(dic_3, dic_1) & k(dic_3, dic_2) & k(dic_3, dic_3) = 1 \end{bmatrix}$$
- $\widetilde{\mathrm{K}}_1^{-1} = \left[\frac{1}{\mathrm{k}_{11}}\right]$: the inverse of kernel matrix.
- $\tilde{\alpha}_1 = \frac{y_1}{k_{11}}$: the coefficient least square vector α at t = 1.
- $P_1 = [1]$: P is the covariance matrix and equal to $[A^T A]^{-1}$.
- $A_t = []_{t\times m}$: is a matrix of least square coefficients $a = (a_1, a_2, \ldots, a_m)$.
    - ✓ Example 1: assume that *t* = 7 and $\mathcal{D} = \{x_1, x_3, x_5\}$. Then $A =$
$$\begin{bmatrix} a_{11} = 1 & a_{13} & a_{15} \\ a_{21} & a_{23} & a_{25} \\ a_{31} & a_{33} = 1 & a_{35} \\ a_{41} & a_{43} & a_{45} \\ a_{51} & a_{53} & a_{55} = 1 \\ a_{61} & a_{63} & a_{65} \\ a_{71} & a_{73} & a_{75} \end{bmatrix}$$
    - ✓ Example 2: x_4 and x_7 can be shown as below:
      $x_4 = a_{41}x_1 + a_{43}x_3$
      $x_7 = a_{71}x_1 + a_{73}x_3 + a_{75}x_5$

In order to obtain a recursive formula for P_t , the Matrix Inversion Lemma is used. Matrix inversion lemma assumes that $\begin{bmatrix} \mathrm{A} & \mathrm{B} \\ \boldsymbol{C} & \mathrm{D} \end{bmatrix}$ is an investable matrix and made of invertible blocks such as A, B, C, D. Subsequently, prove that

$$\begin{bmatrix} \mathrm{A} & \mathrm{B} \\ \boldsymbol{C} & \mathrm{D} \end{bmatrix}^{-1} = (A - B\,.D^{-1}.C)^{-1}$$
$$= A^{-1} + A^{-1}.B.(D - C\,.A^{-1}.B)^{-1}.C\,.A^{-1}$$

A and BCD have the same dimensions. It is linear algebra trick which is applicable in kernel theory (Strang et al., 1993). For finding the inverse of non-square matrix, the pseudo-inverse matrix is used. If the columns of a matrix A are linearly independent, so we should calculate the pseudo inverse with $A^{+} = (A^{T}.A)^{-1}.A^{T}$. However, if the rows of the matrix are linearly independent, the pseudo inverse should be calculated with $A^{+} = A^{T}.(A.A^{T})^{-1}$.

- Λ: it is a binary matrix. It concatenates two sub-matrices of sizes $L \times m_{t-1}$ (# columns is equal to the number of *Dictionary* members in time t - 1) and $L \times G$ (#columns is equal to the number of unsolved orange alarms). It keeps track of whether kernel values of x_t with each *Dictionary* member and kernel values of x_t with each of unsolved orange alarm, exceed the value of d for the previous L time steps or not.

4.2.4. *Projection Error*

For each arriving input vector (*x*) at time (*t*), the projection error δ_t should be evaluated like $\delta_t = k_{tt} - \widetilde{K}_{t-1}(x_t)^T \tilde{a}_t$, where $k_{tt} = k(x_t, x_t)$.

4.2.5. *Kernel Matrix Calculation*

The first step to evaluating δ_t is the computation of the kernel matrix.
The vector $\tilde{k}_{t-1}(x_t)$ includes the kernel value of the current input vector with each *Dictionary* element.

✓ Example: assume that $t = 5$ and $D = \{x_1, x_3, x_5\}$ then $\tilde{k}_4(x_5) = \begin{bmatrix} k(dic_1, x_5) \\ k(dic_2, x_5) \\ k(dic3, x_5) \end{bmatrix}$.

4.2.6. Compute Sparsification Vector a_t

$a_t = \widetilde{K}_{t-1}^{-1}\tilde{k}_{t-1}(x_t)$

4.2.7. Update Λ Matrix

- If $t > L$: Remove first row of matrix Λ and append Λ with 1 or 0
- If $t < L$: Append Λ with one or zero.
    - ✓ 1: when kernel values of x_t with each of *Dictionary* members exceed value of *d*.
    - ✓ 0: when kernel values of x_t with each of *Dictionary* members does not exceed value of *d*.

4.2.8. Raise "Red1" Alarm

- When $\square_t > \nu_2$
- Only matrix Λ changes between time steps and $\widetilde{K}_t$ remains unchanged ($\widetilde{K}_t = \widetilde{K}_{t-1}$).
- Update $q_t = \frac{P_{t-1}\alpha_t}{\gamma + a_t^T P_{t-1}\alpha_t}$
- Update $P_t = \frac{1}{\gamma}(P_{t-1} - q_t a_t^T P_{t-1})$
- Update $\tilde{\alpha}_t = \tilde{\alpha}_{t-1} + \widetilde{K}_{t-1}^{-1} q_t + \left(y_t - \tilde{k}_{t-1}(x_t)^T \tilde{\alpha}_{t-1}\right)$

4.2.9. Raise "Orange" Alarm

- When $\nu_1 < \delta_t < \nu_2$
- Set $\Theta = [\Theta \cup x_t], D = [D \cup x_t]$, where Θ is the set of unsolved Orange alarms.
- $\widetilde{A}_t = a_t$
- Compute $\widetilde{K}_1^{-1}$ and $\widetilde{K}_t$
- Compute $\widetilde{K}_t^{-1} = \begin{bmatrix} \square_t\widetilde{K}_{t-1}^{-1} + \tilde{a}_t\tilde{a}_t^T & -\tilde{a}_t \\ -\tilde{a}_t^T & 1 \end{bmatrix}$
- Compute $\widetilde{K}_t = \begin{bmatrix} \widetilde{K}_{t-1} & \tilde{k}_{t-1}(x_t) \\ \tilde{k}_{t-1}(x_t)^T & k_{tt} \end{bmatrix}$
- Updated $a_t = (0,\cdots,1)^T$

- ✓ Example: assume that $t = 5$, $D = \{x_1, x_3\}$, x_1=dic_1, x_3=dic_2 , and $\square_5 < \nu_2$. Then x_5 causes the Orange alarm. x_5 is not linearly dependent on the *Dictionary* elements. Therefore, x_5 cannot be expressed in the form of: $a_1 \times dic_1 + a_2 \times dic_2$. Observation x_5 can then be stated as: $0 \times dic_1 + 0 \times dic_2 + 1 \times x_5$. The corresponding coefficient vector a_5 when x_5 is added to the *Dictionary* is thus:

  $$\boldsymbol{a_5} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

- Compute $\mathrm{P_t} = \frac{1}{\gamma}\begin{pmatrix} \boldsymbol{P}_{t\text{-}1} & 0 \\ 0^T & \gamma \end{pmatrix}$
  - ✓ Example: assume $t = 5$, $m = 2$, $D = \{x_1, x_3\}$, and $\square_5 < \nu_2$. Then $\boldsymbol{P}_5$ will be equal to $\begin{bmatrix} \boldsymbol{P}_{t-1} & & 0 \\ & & 0 \\ 0 & 0 & 1 \end{bmatrix}$.
- Append Λ with $(0, \cdots, 1)^T$.
  - ✓ Example 1: assume $t = 5$, $D = \{x_1, x_5\}$ and $\Theta = \{x_5\}$ Then $\boldsymbol{\Lambda} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$.

    The second column is the result of $(0\ 1)^T$ at time $t = 5$, when the "Orange" alarm is raised.
  - ✓ Example 2: assume $t = 10$, $D = \{x_1, x_5, x_{10}\}$, $t = 20$ and $\Theta = \{x_5, x_{10}\}$.

Then $\mathbf{\Lambda} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. Both x_5, x_{10} are unsolved orange alarms.

- Update $\alpha_t = \begin{bmatrix} \gamma^{\frac{-1}{2}} \times \tilde{\alpha}_{t-1} - \frac{\tilde{\alpha}_t}{\delta_t}(y_t - \gamma^{\frac{-1}{2}} \times \tilde{k}_{t-1}(x_t)^T \times \tilde{\alpha}_{t-1}) \\ \frac{1}{\delta_t}(y_t - \gamma^{\frac{-1}{2}} \times \tilde{k}_{t-1}(x_t)^T \times \tilde{\alpha}_{t-1}) \end{bmatrix}$.
  - ✓ Example: assume $t = 4$, $m = 1$, $D = \{x_1\}$, $\square_4 < \nu_2$, $a_4 = 1$, $a_3 = [1.0199]$, and x_4 causes orange alarm. Then $P_4 = \begin{bmatrix} 0.333 & 0 \\ 0 & 1 \end{bmatrix}$ which results in $a_4 = \begin{bmatrix} -1.149 \\ 2.48 \end{bmatrix}$.

    As the current input vector is added to the *Dictionary*, the size of *Dictionary* is increased by 1.

4.2.10. Lower Orange Alarm to Green

- If there is an unsolved "Orange" alarm at time step $t - l$, the secondary usefulness test should be applied to resolve the "Orange" alarm. A *Dictionary* element x_{t-1} is regarded as useful if it was used to explain a significant number of input vectors between time steps $t - l$ to t. In other words, if a noticeable amount of kernel values between x_{t-l} and $(x_{t-l+1}, x_{t-l+2}, \cdots, x_t)$ is high, then x_{t-1} should be added to the *Dictionary*, and subsequently x_{t-l} should not be considered as anomaly. It demonstrates the migration or expansion of normal traffic in the feature space.

- Λ matrix also should be updated. The $(m_{t-1})^{th}$ column of matrix Λ keeps track of kernel values of the x_{t-1} ("Orange" alarm) with $(x_{t-l+1}, x_{t-l+2}, \cdots, x_t)$.
- KOAD evaluates the sum of the all kernel values between x_{t-1} and $(x_{t-l+1}, x_{t-l+2}, \cdots, x_t)$ and compares whether it is less than a specific value or not. If it is less than $(\varepsilon \times l)$ it will be considered anomalous and the "Orange" alarm will be elevated to a "Red2" alarm.
    - ✓ Example 1: assume $t = 11$, $D = \{x_1, x_4, x_{10}\}$, $l = 7$ and $\Theta = \{x_4, x_7\}$, $\varepsilon = 0.2$ and

$$\mathbf{\Lambda} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

For resolving "Orange" alarm, S= SUM (Λ (5:11,2)) is calculated.
S is equal to zero ($0 < 0.2 \times 7$). Therefore, the "Orange" alarm should be elevated to "Red2" alarm.

4.2.11. Remove Absolute Elements

When the kernel value of x_{t-L} and all incoming input vectors up to x_t become zero, it causes the relevant column of Λ to contain all zeros. As a result, the $(x_{t-L})^{th}$ member of the *Dictionary* will be marked obsolete and should be removed.

4.2.12. Drop Element (pth) from Dictionary

- This needs to be done either when a previous “Orange” alarm is upgraded to “Red2” alarm, or when a *Dictionary* element becomes obsolete.
- Initially, the pth row and columns of $\widetilde{\mathrm{K}}_{\mathrm{t}}$ and $\widetilde{\mathrm{K}}_{\mathrm{t}}^{-1}$ are moved to the end of the matrix. As a result, the kernel values of every other element with pth element will be transferred to the last row and column of $\widetilde{\mathrm{K}}_{\mathrm{t}}$ and $\widetilde{\mathrm{K}}_{\mathrm{t}}^{-1}$.
- Update $\square_p = \frac{1}{[\widetilde{\mathrm{K}}_{\mathrm{t}}^{-1}]_{\mathrm{m_t,m_t}}}$.
- Update $\tilde{a}_p = -\square_p \times [\widetilde{\mathrm{K}}_{\mathrm{t}}^{-1}]_{1:\mathrm{m_t}\text{-}1,\mathrm{m_t}}$.
- Update $\widetilde{\mathrm{K}}_{\mathrm{t}}^{-1} = [\widetilde{\mathrm{K}}_{\mathrm{t}}^{-1}]_{1:\mathrm{m_t}\text{-}1,\,1:\mathrm{m_t}\text{-}1} - \frac{\tilde{a}_p\tilde{a}_p^T}{\square_p}$.
- Update $\tilde{\alpha}_{\mathrm{t}} = \tilde{\alpha}_{\mathrm{t}} - \frac{1}{\square_p}\begin{pmatrix} \tilde{a}_p\tilde{a}_p^T & -\tilde{a}_p \\ -\tilde{a}_p^T & 1 \end{pmatrix}\widetilde{\mathrm{K}}_{\mathrm{t}}\tilde{\alpha}_{\mathrm{t}}$.
- Update $a_t = a_t(1{:}\,m-1)$.
- Update $\widetilde{K}_t = [\widetilde{K}_t]_{1:m_t-1,1:m_t-1}$.
- Remove pth element from *D*.
- Remove pth column from Λ,
- Update $m_t = m_{t-1} - 1$.
- Update $P = C \times I_{mt}$.
  - ✓ The recalculation of the covariance matrix *P* requires full access to historical data. In order to simplify the calculation, the matrix *P* is reset to a large constant (*C*) times the identity matrix with size equal to m_t (*Dictionary* size at time **t**).
  - ✓ Example, if *C* = 10000 and $m_t = 3$, then $p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

- Update $\widetilde{K}_{t-1}$.
    - ✓ This matrix contains the kernel values of all x_{t-1} *Dictionary* members between themselves.
    - ✓ Example: Assume m = 6 and $D = \{dic_1, dic_2, \cdots, dic_6\}$, then

$$\widetilde{K}_{t\text{-}1} = \begin{bmatrix} k(\text{dic}_1 \text{ and } x_{t\text{-}1}) \\ k(\text{dic}_2 \text{ and } x_{t\text{-}1}) \\ k(\text{dic}_3 \text{ and } x_{t\text{-}1}) \\ k(\text{dic}_4 \text{ and } x_{t\text{-}1}) \\ k(\text{dic}_5 \text{ and } x_{t\text{-}1}) \\ k(\text{dic}_6 \text{ and } x_{t\text{-}1}) \end{bmatrix}.$$

4.2. *K*-means Algorithm

k-means clustering is one of the iterative benchmark unsupervised algorithms that has been used in many clustering applications. Assume that the X is the dataset of *N* samples with d-dimension, where $D = \{x_1, x_2, \cdots, x_N\}, x_N \in R^d$. The k-means algorithm tries to divide the dataset into k disjoint clusters C_i, where $C_i \in \{c_1, c_2, \cdots, C_k\}$. Each cluster is represented with its centroid m_i , where $i = \{1,2,,\cdots,k\}$. Euclidean, Mahalanobis, Manhattan and Chebyshev are examples of distance metrics which can be used by k-means algorithm to measure the similarity between each datapoints x_N and cluster centroids. Figure 6 presents a separation of 2-dimentional data using k-means algorithm with cluster number equal to two.

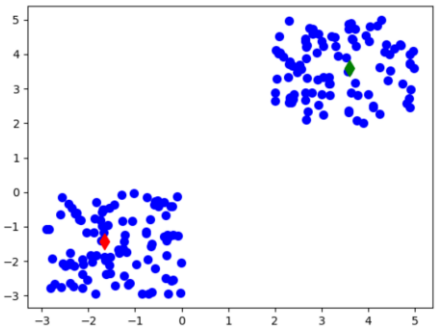


Figure 6 example of *k*-means clustering (k=2)

4.3.1 How K-means Algorithm Works

- Initially, the algorithm selects k points randomly as the centroids of clusters.
- The algorithm measures the distance between every x_N data point and the centroids m_i.
- The algorithm assigns each point to the nearest cluster.
- The algorithm calculates the mean of the points in each cluster and the centroid is replaced by the mean value.
- The algorithm repeats from step 2 until the centroid locations remain unchanged.

The k-means algorithm aims to minimize the squared error objective function in the equation (4.11) to find the local minimum.

$$distortion = error(centroids) = \sum_{i=1}^{N}\sum_{k=1}^{C}(\|x_i - m_k\|)^2 \quad (4.11)$$

K-means algorithm performs ideally when datapoints are distinct and linearly- separated from each other.

K-means algorithm suffers 2 major problems:

- It finds local minimum.
- It requires to define the number of clusters in advance.

4.3.1 How Elbow Method Works

- The number of clusters K can be selected based on the elbow visual-method.
- It starts with K=2 and increases by 1 in each step (Kodinariya and Makwana, 2013)
- It calculates the distortion value in each step and plots the distortion value against the number of clusters (K).

- The location of the knee in the plot (as seen in Figure 7Figure 7) is considered as the most appropriate number of clusters (K), where the distortion value stops decreasing dramatically.

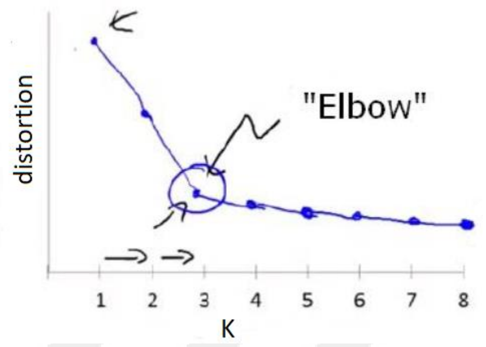


Figure 7 Elbow curve

4.3. Mahalanobis Distance

The Mahalanobis distance is a metric for measuring the distance between a multidimensional point P and a distribution D. It is computed by equation (4.12)

$$D_M = \sqrt{(x_i - \mu)^T S^{-1} (x_i - \mu)}, \tag{4.12}$$

where x_i is a row vector representing the multivariate measurement for an observation, S is the covariance matrix of the sample, μ is the mean of the sample and T is the transpose of matrix. The mean is calculated by equation(4.13).

$$\mu = \frac{\sum_{i=0}^{N} x_i}{N}, \tag{4.13}$$

where N is the number of samples.

The covariance matrix of 2-dimentional point P is calculated by equation (4.14).

$$S = \begin{bmatrix} \delta_1^2 & \delta_2\delta_1 \\ \delta_1\delta_2 & \delta_2^2 \end{bmatrix}, \tag{4.14}$$

where δ_1^2, δ_2^2 are the variance of first and second variables respectively and $\delta_2\delta_1$ is the variance between first and second variables (De Maesschalck et al., 2000).

According to Prykhodko et al. (2018), the squared Mahalanobis distance of the samples from multivariate normal distribution flows a Chi-square distribution $X_{d,\alpha}^2$, where d is the degree of freedom and α is significance level. Additionally, empirical results reveal that the squared Mahalanobis distance is Chi-square distributed (Thill, 2017). The Mahalanobis larger than the critical Chi-square value is related to abnormal/suspicious/outlier datapoints.

Outlier detection is the most common use of the Mahalanobis distance. Additionally, it is used to evaluate the similarity of a set of conditions to a known (predefined) set. For example, in anonymous network traffic detection, the Mahalanobis distance can be used to measure the similarity between unknown traffic and normal behavior (Bayarjargal and Cho, 2014, Santiago-Paz et al., 2012).

4.4. Chi-square Test

The Chi-square test is a non-parametric (distribution-free) statistic test and assumes that data is derived from the random samples. The Chi-square test is used to determine whether a sample data matches a population or not. The Chi-square statistic is used to measure the difference between observed and expected values of the distribution. It also demonstrates the goodness-of-fit between observed and expected values. The Chi-square value is calculated using equation (4.15). To investigate the similarity of the observed and expected values, the value of the Chi-square should be compared against the critical value from a Chi-square table. A Chi-square value higher than the critical value indicates that there is a significant difference between expected/calculated (E) and observed/actual (O) values [Norušis, 2006].

$$X_c^2 = \frac{\sum(O_i - E_i)^2}{E_i} \tag{4.15}$$

Figure 8 shows the Chi-square distribution with the degree of freedom equal to 3 at significance level 0.05.

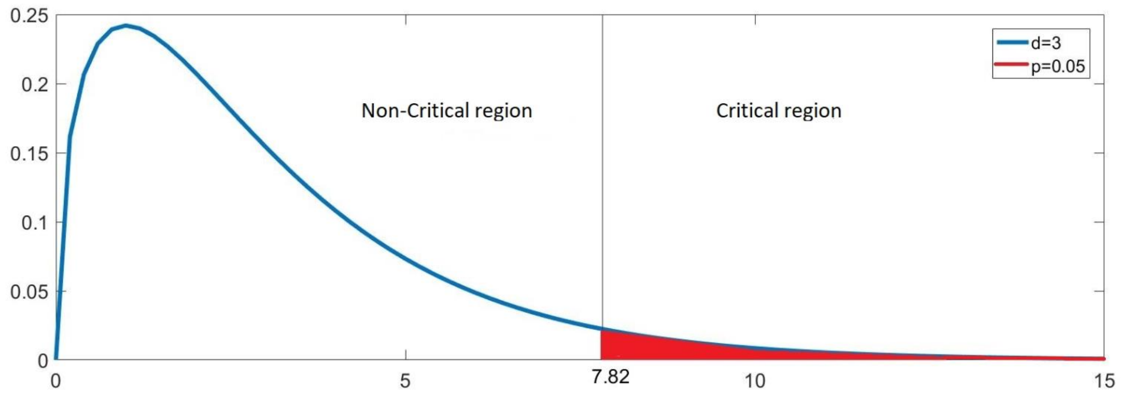


Figure 8 Chi-square distribution with critical and non-critical areas (d=3, α=0.05).

4.5. Chi-square Test

The updated KOAD algorithm is called Enhanced KOAD (E-KOAD), which runs based on the automatic setting of threshold values and the sigma parameter. Additionally, it performs the "Utility Test" before the final decision about inclusion and exclusion of the suspicious input vector in the *Dictionary*.

The "Utility Test" is the combination of "Usefulness Test" of KOAD algorithm, Mahalanobis distance metric and Chi-square test. Therefore, if the significant amount of the subsequent input vectors is dependent on a suspicious data point in the *Dictionary* ("Orange" alarm at x_{t-l}), but the Chi-square test for the corresponding Mahalanobis distance value of that point is true, it should be removed from the *Dictionary* as well. In other words, if the Mahalanobis distance value of a suspicious input vector passed the Chi-square test, it signals the existence of a DDoS attack. As a result, the E-KOAD increases the "Orange" Alarm to the "Red2" alarm and removes the corresponding input vector (x_{t-1}) from the *Dictionary*. Algorithm 1 provides a high-level overview of the E-KOAD algorithm.

Algorithm 1: Pseudocode of the E-KOAD algorithm.

```
1. Run the initialization phase
2. Compute the $\nu_1$, nu$_2$ and $\sigma$
3. Run the E-KOAD
4. For t = 1, 2, . . . do
    a. Compute projection error $\delta_t$ for $x_t$ using $D_t$
    b. if $\delta_t > \nu_2$ then
        i. Raise Red1 Alarm
    c. Endif
    d. if $\delta_t > \nu_1$ then
        i. Raise Orange Alarm
        ii. Store $x_t$ in $\Theta$
    e. Endif

    /* Process previous orange alarm */
    f. if Orange Alarm ($x_{t-l}$) then
        i. Re-evaluate projection error $\delta$ for $x_{t-l}$ using $D_t$
        ii. if $\delta > \nu_1$ then
            1. Perform the Utility Test for $x_{t-l}$
            2. if NOT relevant then

               Raise Red2 Alarm ($x_{t-l}$)
            3. Else

               Add $x_{t-l}$ dictionary D

               Lower Orange Alarm ($x_{t-l}$)
            4. Endif
        iii. Else
            1. Lower Orange Alarm ($x_{t-l}$)
        iv. Endif
            1. Remove $\Theta\{1\}$
    g. Endif

/* Remove obsolete elements */
5. Evaluate usefulness of each dictionary element over previous L
   measurements
6. Remove any useless element from dictionary D
7. EndFor
```

CHAPTER 5

EXPERIMENTAL ANALYSES AND RESULTS

Features were extracted from the CICIDS2017 dataset in the manner described in Chapter 3. Then, the dataset was represented by a row feature vector, with respect to each timestep. The detection algorithm, which was presented in Chapter 4, was then utilized to analyze the constructed row vectors using MATLAB^{TM}.

This chapter presents the main findings of this research. It proceeds as follows. The detection algorithm parameters were set. The algorithm sensitivity is also analyzed regarding different parameter settings. This pursues two fundamental purposes: first, to demonstrate that the results are not immensely sensitive to precise parameter settings by means of Receiver Operating Characteristics (ROC) curves; second, to provide a systematic way to set the parameters to achieve high detection accuracy while reducing false alarms. The performance of the algorithm is then measured using different performance metrics. Subsequently, the performances of the proposed algorithms are quantitatively compared with the performances of other works that utilized the CICIDS2017 dataset to validate their algorithms. Finally, the computational complexity of the proposed algorithm was calculated to verify the claim that the proposed algorithm is suitable for online DDoS detection with complexity values that are independent of time.

5.1. Threshold setting

The original KOAD algorithm incorporates two thresholds. These thresholds were set by trial and error in all previous systems (Ahmed et al., 2007a, Ahmed, 2009, Ahmed et al., 2010, Ahmed et al., 2016b, Ahmed et al., 2017, Sahi et al., 2017, Anika et al., 2017, Islam an Ahmed, 2018).

When the KOAD algorithm commences, there is no definition of normality because the *Dictionary* is empty. Accordingly, ν_2 should be set to the maximum possible value to prevent a "Red1" alarm and the inclusion of all input vectors in the *Dictionary*. During this initial calibration period (60 samples in the experiments), ν_2 is set to 1 (Ahmed et al., 2017). Additionally, ν_1 was set to 0.1 (it could be any value less than 1). The algorithm measures and records δ for 1 hour.

Subsequently, the *k*-means algorithm is used to cluster the δ values. The number of clusters (*K*) is selected based on the elbow method using the distortion metric, which computes the sum of squared distances from each point to its assigned center. As can be seen in Figure 9, the distortion score reduces slightly when the number of clusters is higher than five. Consequently, five can be considered as the number of clusters.

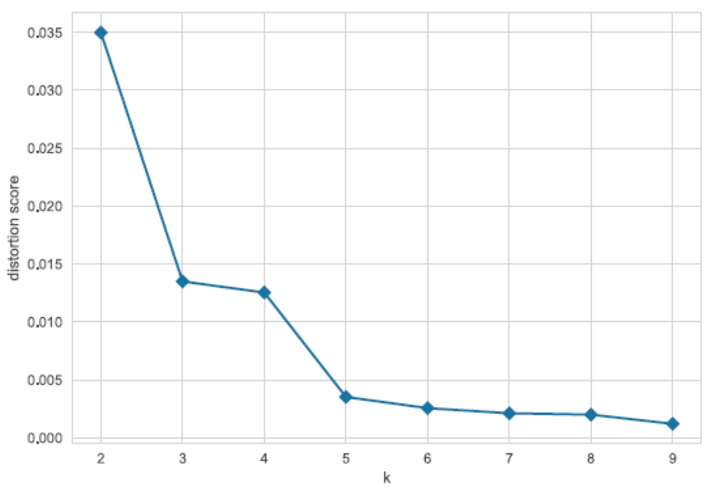


Figure 9 k-elbow visualizer for selecting the optimal number of clusters for k-means algorithm.

Table III shows the number of δ values that belong to each cluster. Cluster3 has the largest size; this demonstrates that the δ values are more likely to belong to Cluster3. Finally, the algorithm updates the threshold values as follows: ν_1 is the minimum value of δ in cluster3 (0.027), and ν_2 is the maximum value of δ in Cluster3 (0.059) at the end of the calibration period (1 hour).

5.2. Sigma (σ) setting

The proposed algorithm can use various types of kernels, such as linear, polynomial, and radial basis function (RBF). As there was not any prior knowledge about the relationships among data points, the general-purpose RBF kernel was employed.

Standard deviation (σ) plays a substantial role in the performance of the RBF kernel. An overly large δ results in losing the discrimination power of the kernel function, as a nearly flat hypersurface is obtained. Accordingly, two points may be considered similar even if

Table III Cluster assumption of δ values for calibration.

Cluster	Size of cluster	Size percentage
Clster1	12	20%
Clster2	13	22%
Clster3	20	33%
Clster4	7	11.66%
Clster5	8	13.33%

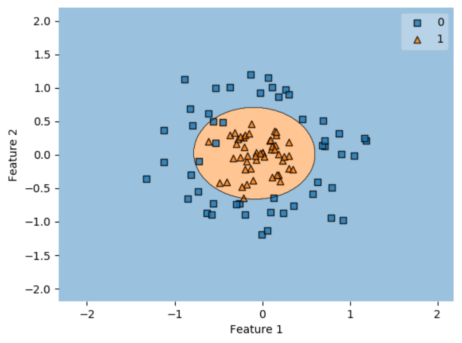
they are far from each other, whereas an overly small σ may cause overfitting. Unfortunately, there is no standard method for defining σ for the RBF kernel. In this work, the algorithm by Liu et al. (2015) was used to detect the optimum σ for the RBF kernel of the E-KOAD algorithm. The algorithm is based on the principle of maximizing between-class separability and minimizing within-class separability. Their algorithm does not require any optimization search process. Therefore, the sigma selection algorithm by Liu et al. (2015) is computationally effective and is less complicated.

The built-in MATLAB sigma selection function for RBF kernel (MATLAB, 2018) was used to select optimal sigma and it resulted in $\sigma = 2.63$

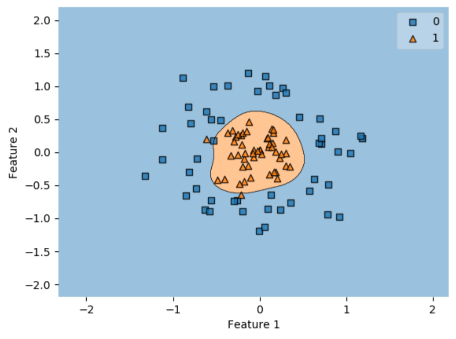
Figure 10 demonstrates the effect of different sigma values on decision boundaries of the SVM algorithm with RBF kernel for the 2-dimensional dataset.

5.3. Tracing of the proposed algorithm

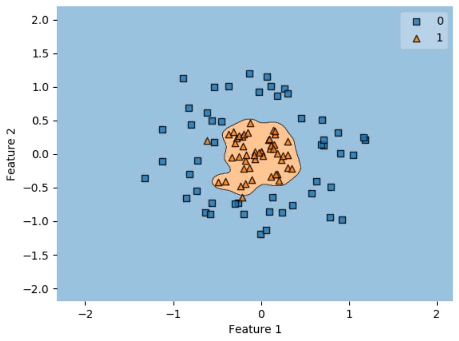
The proposed algorithm is an online algorithm, as was described in Chapter 4. It takes the normalized input feature vector of corresponding network traffic at time step t and investigates the presence of any DDoS attack. The algorithm automatically finds the optimal standard deviation of RBF kernel as $\sigma = 2.63$ and the optimal thresholds as $\nu_1 = 0.027$ and $\nu_2 = 0.059$. The other parameters of the algorithm were derived from the recommended default settings for KOAD (Ahmed et al., 2007a): d = 0.9 and l=20 for resolving orange alarm, ε=0.2 and L = 100 for usefulness testing, and no-forgetting parameter. The algorithm was calibrated in 1 hour using 60 input vectors. This time period is important for the construction of the *Dictionary*. Therefore, all abnormal input data should be treated as "Orange" alarm. Accordingly, the ν_2 threshold was set to 1 for an hour and then declined to 0.059.



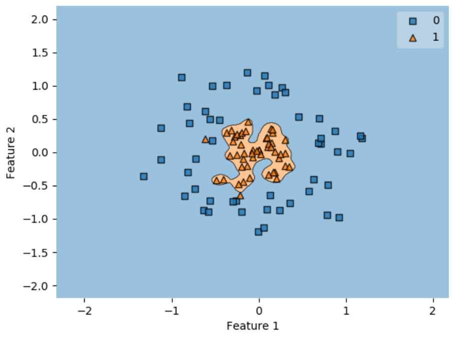
(a)



(b)



(c)



(d)

Figure 10 Decision boundries of the SVM classifier on 2-dimensional data when gamma ($\frac{1}{\delta}$) is equal to (a) one, (b) ten, (c) fifty and (d) hundred.

Figure 11 shows the number of current *Dictionary* members every 20 time steps. It can be seen that the algorithm monitored normal system behavior for approximately 7.5 hours by using a maximum number of 18 *Dictionary* members. The size of the *Dictionary* (m) will be significantly smaller than that of the input vectors (T), thus reducing computational and storage costs. The algorithm was required to store only these *Dictionary* members to measure the Mahalanobis distance of each abnormal data point from normal system activity and detect a DDoS attack.

Figure 12 shows a plot of the detection statistic δ between 8:55 and 17:02 ($535 \leq \text{Time\_Interval} \leq 1022$). As δ is small for normal network traffic, green stems are so small that they cannot be discerned in the figure 11.

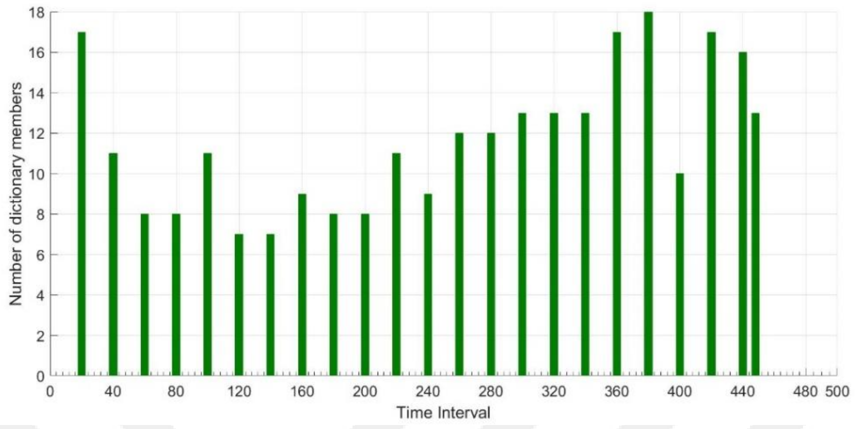


Figure 11 *Dictionary* size (m) corresponding to time intervals in which abnormal data points were evaluated.

n the next step, the algorithm measures the distance between suspicious and normal feature vectors ($\overrightarrow{F_2}$) using the Mahalanobis distance. Figure 13 shows the Mahalanobis distance for each abnormal data point. the Mahalanobis distance was measured from the distribution of the current *Dictionary* every 20 minutes. The proposed algorithm can detect the DDoS attack traffic from other abnormal data points with a maximum delay of 20 minutes. This delay can be adjusted based on the severity level of the system so that DDoS attacks may be detected as soon as possible. The algorithm can be forced to decide about all incoming input vectors instantly by setting (l=0). Of course, this may result in increasing the false positive alarms (Please refer to Table VI for more information).

In the final step, a Chi-square test with 3 degrees of freedom at different significance level of α (0.05, 0.01 and 0.001) was performed using equation (4.15) to evaluate the Mahalanobis distance against the critical value, where O_i was the calculated Mahalanobis distance and E_i was the estimated Mahalanobis distance from the Chi-square table. If the traffic is normal, then with probability less than (α), the score of the moving average of the distance exceeds the corresponding critical value in the Chi-square table. Thus, the score of the moving average of the distance the exceeds that critical value can be regarded as DDoS with probability (1-α).

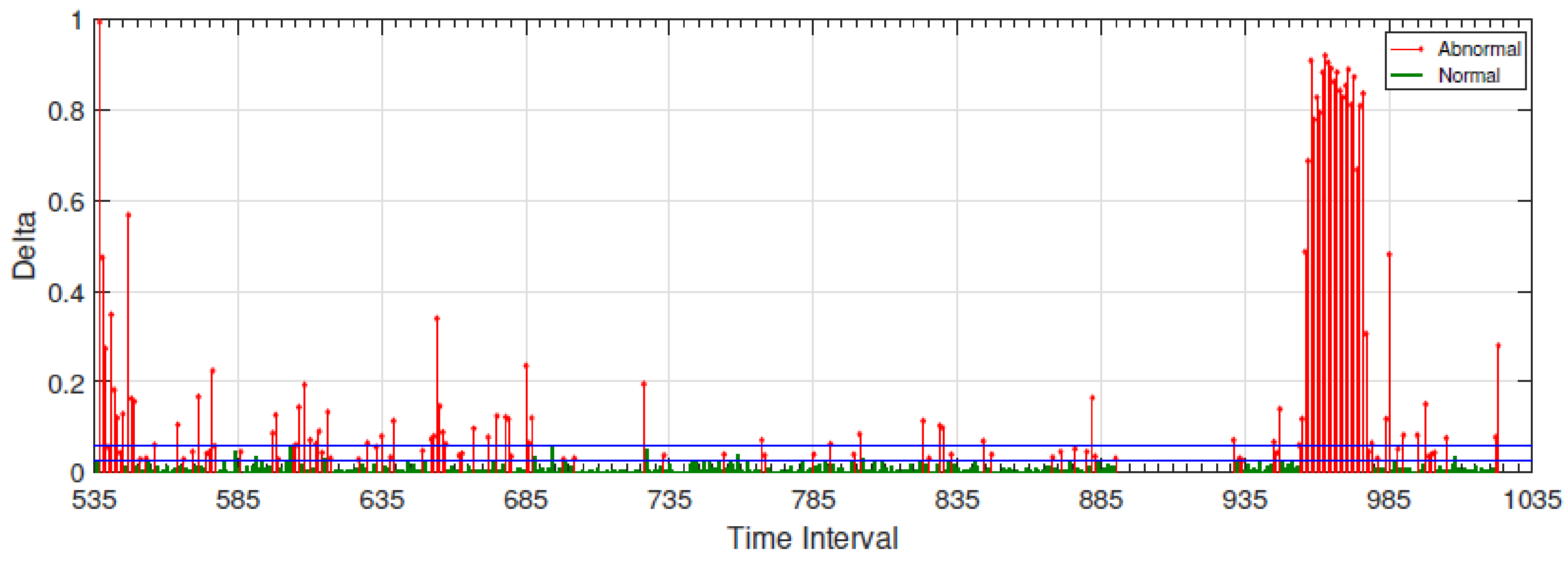


Figure 12 Detection statistic over the dataset. Red stems represent abnormal input data, and green stems represent normal input data. Two vertical blue lines represent v_1 and v_2 thresholds. There are no data between Time_Intervals 989 and 929.

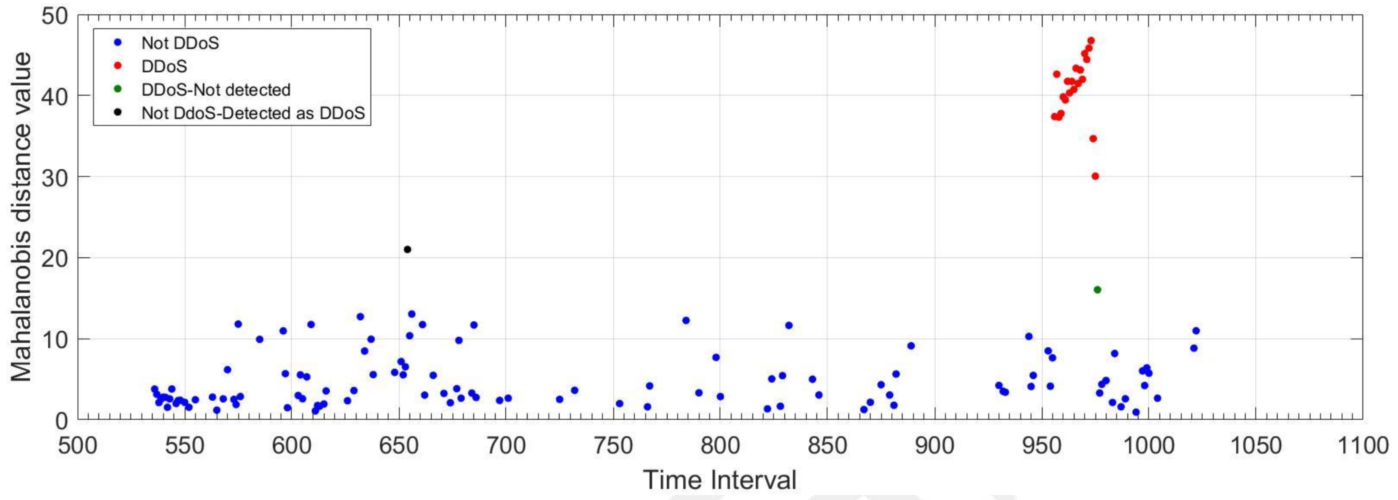


Figure 13 Mahalanobis distance for abnormal data points evaluated using Chi-square test at the significance level of α=0.001. Red data points are related to DDoS attacks and blue data points are related to other types of anomalies in the system. The green data point is the only DDoS attack vector that remained undetected by the algorithm.

The algorithm failed to detect only one DDoS attack at Time_Interval=976. As seen in Figure 13, the DDoS attack started to diminish at Time_Interval = 974 and stopped at 976. It is suspected that this vector was labeled as DDoS in the CICIDS2017 dataset because the DDoS attack simulation tool was actually active at the corresponding time. However, the severity of the simulated DDoS attack was reduced to approximately that of normal network traffic level.

5.4. Complexity analysis

Memory and complexity issues are prominent factors in online detection algorithms. The algorithm's memory requirements are as follows:

- $(m \times m)$ matrix for storing the kernel matrix of *Dictionary* elements (Ahmed et al., 2007a).
- $(l \times d_1)$ matrix for storing input vectors that result in orange alarm, where d_1 is the dimension of the feature vector $(\overrightarrow{F_1})$ (Ahmed et al., 2007a).
- Binary $(L \times m)$ matrix for performing usefulness test (Ahmed et al., 2007a).
- $(S_s \times d_2)$ matrix for storing the suspicious dataset, where S_s is the size of the suspicious dataset at time t, and d_2 is the dimension of the feature vector $(\overrightarrow{F_2})$.
- $(m \times d_2)$ matrix for storing the *Dictionary* dataset, where d_2 is the dimension of the feature vector $(\overrightarrow{F_2})$.

Table IV shows the maximum memory requirements for the proposed algorithm.

The time complexity of the KOAD algorithm is $O(m^2)$ for every usual time interval, and $O(m^3)$ for time intervals when an element is removed from the dictionary. According to Zhang and Zhong (2009), the time complexity of the Mahalanobis metric is $O(d_2^2 + m^2 \times d_2)$. Additionally, the algorithm has a cost of $O(S_s)$ for performing the Chi-square test. The complexity of the proposed algorithm is thus independent of time; accordingly, the algorithm is suitable for online use (Ahmed et al., 2007a).

Table IV Memory complexity of proposed algorithm

Memory complexity	Maximum memory usage
$(m\times m)$	(18×18)
$(el\times d_1)$	(20×4)
$(L\times m)$	(100×18)
$(S_s\times d_2)$	(20×4)
$(m\times d_2)$	(18×4)

5.5. Performance evaluation

As no label information is provided in unsupervised algorithms, there is no specific technique for evaluating the performance of most unsupervised learning methods. According to Jain and Dubes (1998), validating the cluster structure of cluster analyses is a very frustrating task, and it requires deep experience and knowledge in the field. There are internal and external measures of cluster validity as following.

- Cluster Cohesion: Measures the closeness of the objects within a cluster based on the Sum of the Squared Errors (SSE) within clusters.

- Cluster Separation: Measure how well a cluster is far from other clusters based on SSE between clusters.

- Silhouette Coefficient: measures average distance of a point in a cluster from other points in the same cluster and points in different clusters. It combines the ideas behind both cluster cohesion and cluster separation.

- Entropy: measures the amount of disorder in a point using Shannon entropy.

- Purity: measures the cleanness of a cluster based on the definition of entropy. The purity of a cluster is the maximum probability that a member of a cluster belongs to a specific class.

However, it is difficult to find an appropriate metric for the validity of clustering algorithms (Kovács et al., 2005).

On the other hand, if label information is provided the performance of the algorithm can be measured precisely.

Table V depicts the confusion matrix to describe the performance of a classification algorithm.

Table V Confusion matrix in machine learning

-		Predictive	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

Four possible outcomes of the confusion matrix for the classification of a dataset in this thesis are given as:

- True Positive (TP) is the number of DDoS attack traffic vectors that are classified correctly.
- True Negative (TN) is the number of normal network traffic vectors that are classified correctly.
- False Positive (FP) is the number of normal network traffic vectors that are incorrectly classified as DDoS attack.
- False Negative (FN) is the number of DDoS attack traffic vectors that are incorrectly classified as normal traffic.

The class label information of the CICIDS2017 dataset were only used as a reference to assess the proposed unsupervised approach. To evaluate the algorithm, the following metrics are utilized:

- Accuracy: is the number of all correct predictions divided by the total number of the dataset and is calculated using equation (5.1) :

$$\text{Accuracy} = \frac{\text{TN} + \text{TP}}{\text{TN} + \text{TP} + \text{FN} + \text{FP}} \tag{5.1}$$

- Recall: also known as sensitivity or True Positive Rate (TPR), is a measure that tells us what proportion of network traffic that was actually DDoS attacks was detected by the algorithm as DDoS using equation (5.2):

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.2}$$

- Precision: is a measure that tells us what proportion of network traffic was detected as DDoS, actually was DDoS attack using the equation (5.3):

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.3}$$

- False Positive Rate (FPR): is a measure that tells us what proportion of network traffic detected as normal (non-DDoS) was detected by the algorithm as DDoS using equation (5.4).

$$\text{FPR} = \frac{\text{FP}}{\text{TN} + \text{FP}} \tag{5.4}$$

- Receiver Operating Characteristic (ROC) curve. It plots FPR against TPR for different threshold settings.Figure 14 shows the trade-off between FPR and TPR using four threshold settings.

Table VI presents the performance metrics for the proposed algorithm based on various values of the time that is required to resolve the orange alarm (*l*). It can be seen that reducing *l* from 20 to 2 minutes did not affect the detection power of the algorithm, and the algorithm is not sensitive to this parameter. However, the number of false alarms issued by the algorithm increased sharply when *l* dropped below 5 minutes.

Table VI Performance metrics of the proposed algorithm for different values of *l*.

Dataset size	# DDoS attacks	*l*	FN	FP	ACC	TPR	FPR
448	21	20	1	1	99.55%	95.23%	0.23%
448	21	10	1	6	99.55%	95.23%	1.4%
448	21	5	1	4	99.55%	95.23%	0.9%
448	21	2	1	71	99.55%	95.23%	17%

The area under the ROC curve is a more robust performance metric than accuracy. In the experiments, the ROC curve analysis was utilized to demonstrate the effect of selecting different thresholds, *l* (lag-time for resolving orange alarm) and alpha (significance level in Chi-square test) by trial and error on the performance of the proposed algorithm. The point closest to the upper-left corner of the ROC curve indicates better performance, as it has lower FPR and higher TPR.

The red point in Figure 14 indicates the best performance for the algorithm when the thresholds are selected automatically. Ahmet et al. (2007a) pointed out that optimal threshold setting is quite challenging and depends on various traffic metrics. The proposed algorithm is able to adjust both the ν_1 and ν_2 thresholds automatically using an unsupervised method, so that performance is optimized, as seen in Figure 14.

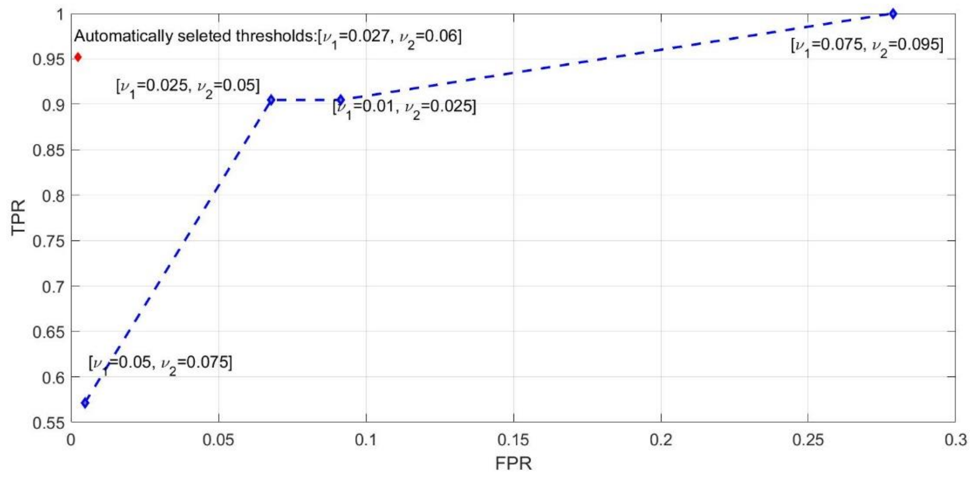


Figure 14 ROC curve shows the trade-off between FPR and TPR with different threshold settings.

The results in Table VI shows that the algorithm requires at least 5 minutes to detect 95% of DDoS attacks with false positive alarm rate below 1%. The algorithm requires some time to establish a normal traffic baseline. Experiments were repeated when lag-time for "Orange" alarm was altered between 20 and 2 minutes. As can be seen in Figure 15, the detection rate remains identical while the FPR is increasing for the small values of *l*.

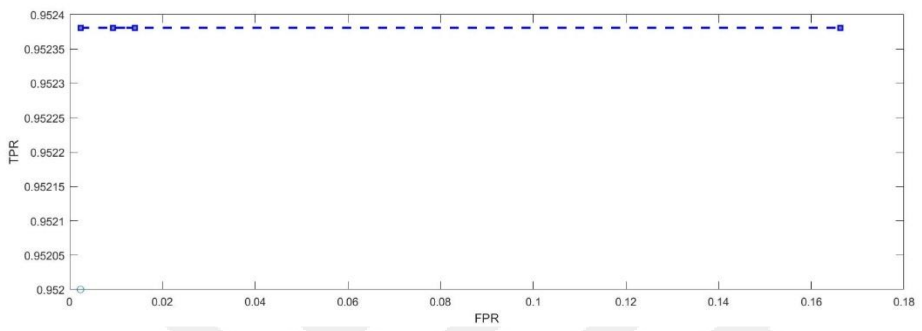


Figure 15 ROC curve shows the trade-off between FPR and TPR with different lag-time settings.

Experiments were also repeated at the different significance levels (α) of the Chi-square test. The results of the Chi-squre analyses are presented in Appendix A. As can be seen in figure 14, the detection rate is 100% when α=0.01 but 95% when α=0.001. On the other hand, the increase in the detection rate causes an increase of 2% in FPR as well. Therefore, there is a trade-off between detection rate and FPR regarding the selecting of the significance level (α).

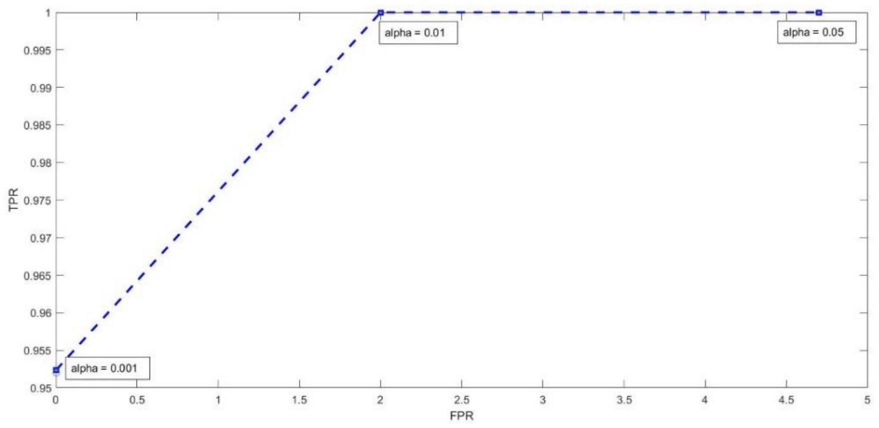


Figure 16 ROC curve shows the trade-off between FPR and TPR with the different significance level (α) settings.

5.6. Comparison with literature

Table VII compares the proposed algorithm with those by other studies that used the CICIDS2017 dataset. Of course, as this is a recent dataset, the number of these studies is still limited. It can be seen that the proposed algorithm is comparable with the others. Its performance is slightly lower than that of the Autoencode algorithm (Attak et al., 2018). However, the proposed algorithm has several advantages: it does not require labeled data, it is completely unsupervised, its memory and time complexity is independent of time and it can operate online as well as adapt to changing network traffic.

Table VII Comparison of the proposed hybrid algorithm with others.

Author	Algorithms	Online	Best Accuracy	Recall	Precision
Aamir and Zaidi (2019)	*k*-NN, RF, SVM	No	96.66% (RF)	91.7%	88%
Aksu et al. (2018)	*k*-NN, DT, SVM	No	99.8% (*k*-NN)	95.8%	95.6%
Attak et al. (2018)	One-calss SVM, Auto-encoder	No	98% (Auto-encoder)	97%	99%
Boukhamla and Gaviro	*k*-NN, DT (c4.5), Naïve-bayes	No	96.65% (*k*-NN)	90.6%	91.27%
Ahmim et al. (2018)	JRip algorithm, Random Forest	No	NA	NA	NA
Azwar et al. (2018)	Gradient Boosted DT (XGBoost)	No	NA	81.8%	84.4%
Proposed algorithm	Hybrid	Yes	99.55%	95.24%	95.24%

Moreover, the performance of the proposed algorithm was compared with the work by Gu et al. (2019). This comparison was not placed on Table VII because Gu et al. (2019) used a different performance measurement metric named TOPSIS. They set TOPSIS to $(1 - \text{Recall} + \text{FPR})$ to represent the fitness function of their proposed algorithm. The TOPSIS metric was also computed for the proposed algorithm. The values equal to 0.05 and 0.0218 were obtained for the best and worst performance of the proposed algorithm based on the different lag-times (l) required to resolve the orange alarm. However, Gu et al. (2019) reported the TOPSIS value of 0.298 for their semi-supervised algorithm.

5.7. Comparison with benchmark unsupervised algorithms

The performance of the proposed algorithm also was compared with the performance of the benchmark offline unsupervised algorithms. The built-in unsupervised machine learning algorithms in the Sklearn library of Python^{TM} were utilized.

It is clearly shown in Table VIII that TPR of the proposed algorithm exceeds the Density-Based Clustering (DBSCAN) and Agglomerative clustering algorithms. Additionally, the proposed online algorithm reaches the same TPR of the offline k-means algorithm. However, *k*-means achieved a slightly lower FPR. The results confirm that the novel proposed algorithm has comparable performances with unsupervised benchmark algorithms. The advantageous of the proposed detection algorithm is its ability to operate in an online manner. Offline algorithms require iterative re-clustering to adapt to the changing trends in network traffic. Therefore, their time and memory complexity are dependent of time.

Table VIII Comparison of the proposed hybrid algorithm with benchmark unsupervised algorithms.

Algorithms	Online	Parameter settings	FPR	TPR
k-means	No	Number of clusters	0	95.23%
Agglomerative	No	Number of clusters	0	85.71%
DBSCAN	No	Minimum samples in each cluster	0	90.4%
Proposed algorithm	Yes	L, *l*, ε	0.23%	95.23%

Figure 17 demonstrates the results of the *k*-means clustering analysis. The 8-dimensional data was converted to 2-dimensional data using PCA and then multi-dimension clusters were depicted in 2-dimensional plot. As it is seen in the Figure 17, clusters are separated perfectly.

5.8. Indirect comparison with state-of-the-art

This section includes the indirect performance comparison of the proposed algorithm with 6 recent unsupervised approaches for DDoS attack detection. The proper comparison of machine learning-based attack detection studies is possible only when they use the same validation dataset or the same algorithm. As replication of all detection algorithms is a frustrating task, here, only an indirect comparison of the algorithms was made. It can be inferred from Table IX that the proposed algorithm is superior to other existing methods except for the algorithm by Idhammad et al. (2018) from both perspectives of TPR and

FPR. Furthermore, the proposed algorithm is the only DDoS detection algorithm that meets all requirements of an online learning algorithm.

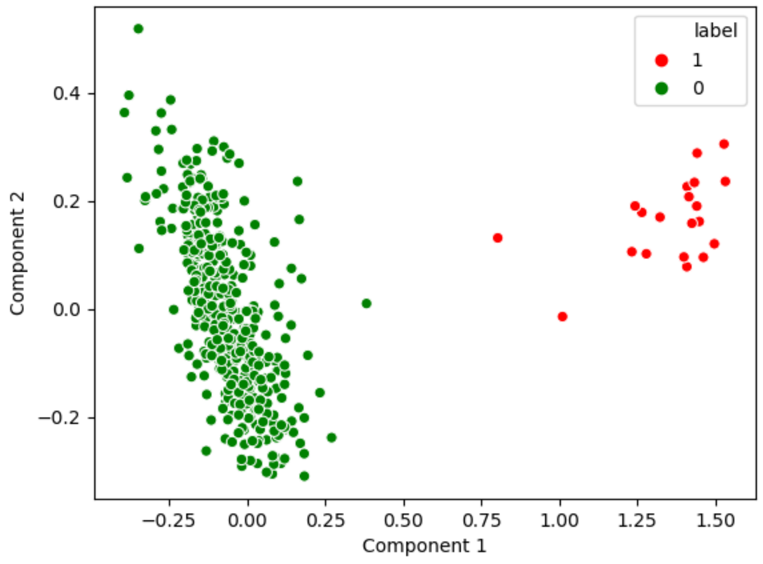


Figure 17 Detection of DDoS attacks using k-means algorithm

Table IX Indirect comparison of the proposed algorithm with recent unsupervised DDoS Detection algorithms.

Author	Algorithm	Online	TPR	FPR
Casas et al. (2012 a,b)	DBSCAN, SSC, EAC	No	90%	1-3.5%
Dromard et al. (2016)	Incremental Clustering	Yes	94%	0
Roudiere and Owezarski (2017)	*k*-NN, Histogram	Semi	83%	0.01%
Fernandes Jr et al. (2016)	PCA, Anti Colony Optimization	Semi	92%	21%
Idhammad et al. (2018)	Co-clustering, Entropy, Extra Tree	No	98%	0.33%
Proposed algorithm	E-KOAD, Entropy, Mahalanobis Distance, Chi-square	Yes	95%	0.23%

CHAPTER 6

CONCOLUSION AND FUTURE DIRECTIONS

This thesis has addressed the detection of DDoS attacks in the Local Area Network (LAN) with an emphasis on automating the adaption of the detection algorithm on frequently changing network traffic. Researchers have advocated a variety of different DDoS detection schemes since the 2000s. However, a literature survey reveals that the techniques available today leave some significant gaps. First, most existing algorithms require batch-processing, which causes time-delay. Second, most online-processing algorithms require pre-defined settings of algorithm parameters and thresholds. Third, most online-processing algorithms require high memory and storage resources and have significant time complexities. Forth, outdated DDoS datasets or limited simulation approaches were used to validate the proposed detection algorithms by most researchers.

6.1. Concolusion

In this thesis, a novel algorithm has been proposed for the detection of DDoS attacks by utilizing a kernel-based anomaly detection method, the Mahalanobis distance metric and Chi-square test where there is no need to train detection algorithm using labeled data. Discriminating DDoS attacks from normal traffic using an unsupervised algorithm that can adapt itself to a continually changing environment without requiring a predefined normal behavior of the network traffic and re-training the detection model is a considerably new attempt for DDoS detection, and the proposed algorithm has shown promising results. The KOAD algorithm was used as a backbone of the proposed detection scheme while it was improved by defining an automatic procedure for setting the thresholds

The proposed algorithm was validated on the CICIDS2017 dataset. The performance of the proposed algorithm was also evaluated based on different time settings for resolving suspicious network traffic and significant-level (α) for the Chi-square test through Receiver Operating Characteristics (ROC) curves. Additionally, the analysis of the sensitivities of the threshold settings of the proposed algorithm has shown that the algorithm produces the highest performances when thresholds are set automatically.

The proposed algorithm has been compared with the three most well-known benchmark unsupervised algorithms which are currently being used in DDoS detection literature: *k*-means, Agglomerative and DBSCAN. Moreover, the performance of the proposed algorithm was directly compared with the works of six researchers who utilized the CICIDS2017 dataset as their validation instrument. It has been shown that the proposed algorithm achieved a higher detection accuracy rate compared to the other six algorithms, in addition to meeting all constraints of an online detection algorithm. The performance of the proposed algorithm also was indirectly compared with five new unsupervised DDoS detection algorithms in the literature. The results revealed that the proposed algorithm outperformed the other four algorithms while only achieved 3% less TPR than offline unsupervised DDoS detection algorithm by Idhammad et al. (2018).

The proposed algorithm is based on the kernel online anomaly detection algorithm and intrinsically suitable for an online application as its computational and memory complexities are independent of time. As might be expected, the proposed algorithm is superior to the original KOAD algorithm due to the automatic selection of thresholds.

6.2. Future directions

The scope of this thesis has been limited to DDoS detection in a LAN environment. The future researches might convey the proposed algorithm into cloud networks, ISP-level networks or Software Defined Networks (SDN). A weakness of the proposed algorithm at this point is to classify different kinds of DDoS attacks and ideally distinguish them from similar-looking FE traffic. The algorithm also suffers to find DDoS attacks, which are very similar to normal underlying network traffic from the perspective of volume, traffic rate, traffic duration and randomness of source/destination IP addresses/ports. It is hypothesized that geolocation analysis of IP addresses, historical analysis of IP addresses, and black-listing/white-listing techniques could be added as extra components to the proposed detection scheme. These analyses can be applied to all suspicious traffic vectors in addition to the Mahalanobis distance measurement. Besides, rule-based correlation algorithms could be added to the machine learning-based DDoS detection scheme to establish patterns to control events, which might be indicators of a DDoS attack.

REFERENCES

Aamir, M., & Zaidi, S. M. A. (2019). Clustering based semi-supervised machine learning for DDoS attack classification. *Journal of King Saud University-Computer and Information Sciences*.

Agarwal, B. and Mittal, N. (2012). Hybrid approach for detection of anomaly network traffic using data mining techniques. *Procedia Technology*, 6:996–1003.

Ahmed, M. and Mahmood, A. N. (2014). Network traffic analysis based on collective anomaly detection. In *2014 9th IEEE Conference on Industrial Electronics and Applications*, pages 1141–1146. IEEE.

Ahmed, M. and Mahmood, A. N. (2015). Novel approach for network traffic pattern analysis using clustering-based collective anomaly detection. *Annals of Data Science*, 2(1):111–130.

Ahmed, M., Mahmood, A. N., and Hu, J. (2016a). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31.

Ahmed, T. (2009). Online anomaly detection using KDE. In *GLOBECOM*, pages 1–8.

Ahmed, T., Ahmed, S., Ahmed, S., and Motiwala, M. (2010). Real-time intruder detection in surveillance networks using adaptive kernel methods. In *Proc. IEEE International Conference on Communications*, pages 1–5, Cape Town, South Africa.

Ahmed, T., Ahmed, S., and Chowdhury, F. E. (2016b). Taking meredith out of grey's anatomy: Automating hospital ICU emergency signaling. In *Proc. Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 1886–1890, Shanghai, China.

Ahmed, T., Coates, M., and Lakhina, A. (2007a). Multivariate online anomaly detection using kernel recursive least squares. In *Proc. 26th IEEE International Conference on Computer Communications (INFOCOM)*, pages 625–633, Anchorage, AK, USA.

Ahmed, T., Oreshkin, B., and Coates, M. (2007b). Machine learning approaches to network anomaly detection. In *Proceedings of the 2nd USENIX workshop on Tackling computer systems problems with machine learning techniques*, pages 1–6, Cambridge, MA, USA. USENIX Association.

Ahmed, T., Pathan, A.-S. K., and Ahmed, S. (2014). Adaptive algorithms for automated intruder detection in surveillance networks. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 2775–2780. IEEE.

Ahmed, T., Pathan, A.-S. K., and Ahmed, S. S. (2017). Learning algorithms for anomaly detection from images. In *Biometrics: Concepts, Methodologies, Tools, and Applications*, pages 281–308. IGI Global.

Ahmed, T., Wei, X., Ahmed, S., and Pathan, A.-S. K. (2013). Efficient and effective automated surveillance agents using kernel tricks. *Simulation*, 89(5):562–577.

Ahmim, A., Maglaras, L., Ferrag, M. A., Derdour, M., & Janicke, H. (2019, May). A novel hierarchical intrusion detection system based on decision tree and rules-based models. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)* (pp. 228-233). IEEE.

Akamai (2019). Financial services attack economy. https://www.akamai.com/us/en/campaign/assets/soti/security-financial-services-attack-economy.jsp. [Online; accessed 30-July-2018].

Akamai Technologies (2019). State of the internet security. Technical report. [Online]. Available: https://content.akamai.com/PG11811-soti-2018-a-year-in-review-report.html?lang=us-en.Aksu, D., Üstebay, S., Aydin, M. A., & Atmaca, T. (2018, September). Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature selection algorithm. In *International Symposium on Computer and Information Sciences* (pp. 141-149). Springer, Cham.

Anika, A., Karim, K. L., Muntaha, R., Shahrear, F., Ahmed, S., and Ahmed, T. (2017). Multi image retrieval for kernel-based automated intruder detection. In *IEEE Region 10 Symposium (TENSYMP), 2017*, pages 1–5. IEEE.

Arbor (2016). Arbor Networks Spectrum: The Network-Based Advanced Threat Solution. https://www.netscout.com/report/. [Online; accessed 15-July-2018].

Attak, H., Combalia, M., Gardikis, G., Gastón, B., Jacquin, L., Katsianis, D., Litke, A., Papadakis, N., Papadopoulos, D., Pastor, A., et al. (2018). Application of distributed computing and machine learning technologies to cybersecurity. In *Computer & Electronics Security Applications Rendez-vous (C&ESAR)*.

Azwar, H., Murtaz, M., Siddique, M., & Rehman, S. (2018, November). Intrusion Detection in secure network for Cybersecurity systems using Machine Learning and Data Mining. In *2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS)* (pp. 1-9). IEEE.

Bayarjargal, D. and Cho, G. (2014). Detecting an anomalous traffic attack area based on entropy distribution and Mahalanobis distance. *International Journal of Security and Its Applications*, 8(2):87–94.

Behal, S. and Kumar, K. (2017). Detection of DDoS attacks and flash events using information theory metrics–an empirical investigation. *Computer Communications*, 103:18–28.

Behal, S., Kumar, K., and Sachdeva, M. (2017). Characterizing DDoS attacks and flash events: Review, research gaps and future directions. *Computer Science Review*, 25:101–114.

Behal, S., Kumar, K., and Sachdeva, M. (2018). D-face: An anomaly based distributed approach for early detection of DDoS attacks and flash events. *Journal of Network and Computer Applications*, 111:49–63.

Bhuyan, M. H., Bhattacharyya, D., and Kalita, J. K. (2015a). An empirical evaluation of information metrics for low-rate and high-rate DDoS attack detection. *Pattern Recognition Letters*, 51:1–7.

Bhuyan, M. H., Kalwar, A., Goswami, A., Bhattacharyya, D., and Kalita, J. (2015b). Low-rate and high-rate distributed dos attack detection using partial rank correlation. In *Communication Systems and Network Technologies (CSNT), 2015 Fifth International Conference on*, pages 706–710, Gwalior, India. IEEE.

Boukhamla, A., & Gaviro, J. C (2018). CICIDS2017 dataset: performance improvements and validation as a robust intrusion detection system testbed. *International Journal of Information and Computer Security*.

CAIDA Dataset (2007). The CAIDA DDoS attack dataset. Massachusetts Institute of Technology. [Online]. Available: http://www.caida.org/data/passive/ddos-20070804dataset.xml.

Casas, P., Mazel, J., and Owezarski, P. (2012a). Knowledge-independent traffic monitoring: Unsupervised detection of network attacks. *IEEE Network*, 26(1):13–21.

Casas, P., Mazel, J., and Owezarski, P. (2012b). Unsupervised network intrusion detection systems: Detecting the unknown without knowledge. *Computer Communications*, 35(7):772–783.

Chen, L., Zhang, Y., Zhao, Q., Geng, G., and Yan, Z. (2018). Detection of DNS DDoS attacks with random forest algorithm on spark. *Procedia computer science*, 134:310–315.

Chen, R.-C., Cheng, K.-F., Chen, Y.-H., and Hsieh, C.-F. (2009). Using rough set and support vector machine for network intrusion detection system. In *Proc. Intelligent Information and Database Systems, 2009. ACIIDS 2009. First Asian Conference on*, pages 465–470, Dong Hoi, Vietnam.

Cheng, J., Yin, J., Liu, Y., Cai, Z., and Li, M. (2009). DDoS attack detection algorithm using IP address features. In *International Workshop on Frontiers in Algorithmics*, pages 207–215. Springer.

Chitrakar, R. and Huang, C. (2014). Selection of candidate support vectors in incremental SVM for network intrusion detection. *computers & security*, 45:231–241.

CICFLOWMETER. A network traffic biflow generator and analyzer. Available at http://netflowmeter.ca/ (2019/01/18).

Daneshgadeh, S., Ahmed, T., Kemmerich, T., and Baykal, N. (2019a). Detection of DDoS attacks and flash events using Shannon entropy, KOAD and Mahalanobis distance. In *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 222–229. IEEE.

Daneshgadeh, S., Baykal, N., et al. (2017). DDoS attack modeling and detection using SMO. In *Machine Learning and Applications (ICMLA), 2017 16th IEEE International Conference on*, pages 432–436, Cancun, Mexico. IEEE.

Daneshgadeh, S., Kemmerich, T., Ahmed, T., and Baykal, N. (2018). A hybrid approach to detect DDoS attacks using KOAD and the Mahalanobis distance. In *2018 IEEE 17th*

International Symposium on Network Computing and Applications (NCA), Cambridge, MA USA. IEEE.

Daneshgadeh, S., Kemmerich, T., Ahmed, T., and Baykal, N. (2019b). An empirical investigation of DDoS and flash event detection using Shannon entropy, KOAD and SVM combined. In *2019 International Conference on Computing, Networking and Communications (ICNC)*, pages 658–662. IEEE.

DataFlair Team (2018). Kernel functions-introduction to SVM kernel & examples. [Online]. Available: https://data-flair.training/blogs/svm-kernel-functions/.

David, J. and Thomas, C. (2015). DDoS attack detection using fast entropy approach on flow-based network traffic. *Procedia Computer Science*, 50:30–36.

David, J. and Thomas, C. (2019). Efficient DDoS flood attack detection using dynamic thresholding on flow-based network traffic. *Computers & Security*.

De Maesschalck, R., Jouan-Rimbaud, D., and Massart, D. L. (2000). The Mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 50(1):1–18.

Dromard, J., Roudière, G., and Owezarski, P. (2016). Online and scalable unsupervised network anomaly detection method. *IEEE Transactions on Network and Service Management*, 14(1):34–47.

Engel, Y., Mannor, S., and Meir, R. (2004). The kernel recursive least-squares algorithm. *IEEE Transactions on signal processing*, 52(8):2275–2285.

Fernandes Jr, G., Carvalho, L. F., Rodrigues, J. J., and Proença Jr, M. L. (2016). Network anomaly detection using IP flows with principal component analysis and ant colony optimization. *Journal of Network and Computer Applications*, 64:1–11.

Gan, X.-s., Duanmu, J.-s., Wang, J.-f., and Cong, W. (2013). Anomaly intrusion detection based on pls feature extraction and core vector machine. *Knowledge-Based Systems*, 40:1–6.

Gogoi, P., Bhattacharyya, D., Borah, B., and Kalita, J. K. (2013). Mlh-IDS: a multi-level hybrid intrusion detection method. *The Computer Journal*, 57(4):602–623.

Gu, Y., Li, K., Guo, Z., and Wang, Y. (2019). Semi-supervised k-means DDoS detection method using hybrid feature selection algorithm. *IEEE Access*, 7:64351–64365.

Haasdonk, B. and Burkhardt, H. (2007). Invariant kernel functions for pattern analysis and machine learning. *Machine learning*, 68(1):35–61.

Hoque, N., Bhattacharyya, D. K., and Kalita, J. K. (2016). Ffsc: a novel measure for low-rate and high-rate DDoS attack detection using multivariate data analysis. *Security and Communication Networks*, 9(13):2032–2041.

Hoque, N., Kashyap, H., and Bhattacharyya, D. (2017). Real-time DDoS attack detection using FPGA. *Computer Communications*, 110:48–58.

Horng, S.-J., Su, M.-Y., Chen, Y.-H., Kao, T.-W., Chen, R.-J., Lai, J.-L., and Perkasa, C. D. (2011). A novel intrusion detection system based on hierarchical clustering and support vector machines. *Expert systems with Applications*, 38(1):306–313.

Idhammad, M., Afdel, K., and Belouch, M. (2018). Semi-supervised machine learning approach for DDoS detection. *Applied Intelligence*, 48(10):3193–3208.

Islam, H. and Ahmed, T. (2018). Anomaly clustering based on correspondence analysis. In *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, pages 1019–1025. IEEE.

ISO/IEC 27002:2013 (2013). Information technology - Security techniques - Code of practice for information security management. Standard, International Organization for Standardization, Geneva, CH.

Jain, A. K. and Dubes, R. C. (1988). Algorithms for clustering data. *Englewood Cliffs: Prentice Hall, 1988*.

Jain, A. K. and Dubes, R. C. (1988). Algorithms for clustering data. *Englewood Cliffs: Prentice Hall, 1988*.

Jun, J.-H., Ahn, C.-W., and Kim, S.-H. (2014). DDoS attack detection by using packet sampling and flow features. In *proceedings of the 29th annual ACM symposium on applied computing*, pages 711–712, Gyeongju, Korea. ACM.

Jyothsna, V., Prasad, V. R., and Prasad, K. M. (2011). A review of anomaly-based intrusion detection systems. *International Journal of Computer Applications*, 28(7):26–35.

Khan, L., Awad, M., and Thuraisingham, B. (2007). A new intrusion detection system using support vector machines and hierarchical clustering. *The VLDB journal*, 16(4):507–521.

Kodinariya, T. M. and Makwana, P. R. (2013). Review on determining number of cluster in k-means clustering. *International Journal*, 1(6):90–95.

Kovács, F., Legány, C., and Babos, A. (2005). Cluster validity measurement techniques. In *6th International symposium of hungarian researchers on computational intelligence*, page 35. Citeseer.

Kuang, F., Xu, W., and Zhang, S. (2014). A novel hybrid KPCA and SVM with GA model for intrusion detection. *Applied Soft Computing*, 18:178–184.

Lee, K., Kim, J., Kwon, K. H., Han, Y., and Kim, S. (2008). DDoS attack detection method using cluster analysis. *Expert systems with applications*, 34(3):1659–1665.

Li, K., Zhou, W., Li, P., Hai, J., and Liu, J. (2009). Distinguishing DDoS attacks from flash crowds using probability metrics. In *Network and System Security, 2009. NSS'09. Third International Conference on*, pages 9–17. IEEE.

Liao, H.-J., Lin, C.-H. R., Lin, Y.-C., and Tung, K.-Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24.

Liu, Z., Zuo, M. J., Zhao, X., and Xu, H. (2015). An analytical approach to fast parameter selection of gaussian RBF kernel for support vector machine. *J. Inf. Sci. Eng.*, 31(2):691–710.

MATLAB (2018). Sigma selection of gaussian RBF kernel for classification.

Mirkovic, J. and Reiher, P. (2004). A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53.

MIT Lincoln Laboratory (2000). MIT LLS_DDOS Dataset. Massachusetts Institute of Technology. [Online]. Available: http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/idel/data/2000data.html.

Montgomery, D. C., Jennings, C. L., and Kulahci, M. (2015). *Introduction to time series analysis and forecasting*. John Wiley & Sons, Hoboken, NJ, USA.

Moore, D., Shannon, C., Brown, D. J., Voelker, G. M., and Savage, S. (2006). Inferring internet denial-of-service activity. *ACM Transactions on Computer Systems (TOCS)*, 24(2):115–139.

Nesselroade, J. R. and Cattell, R. B. (2013). *Handbook of multivariate experimental psychology*. Springer Science & Business Media.

Nezhad, S. M. T., Nazari, M., and Gharavol, E. A. (2016). A novel DoS and DDoS attacks detection algorithm using ARIMA time series model and chaotic system in computer networks. *IEEE Communications Letters*, 20(4):700–703.

Norušis, M. J. (2006). *SPSS 14.0 guide to data analysis*. Prentice Hall Upper Saddle River, NJ.

Nychis, G., Sekar, V., Andersen, D. G., Kim, H., and Zhang, H. (2008). An empirical evaluation of entropy-based traffic anomaly detection. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 151–156. ACM.

Papalexakis, E. E., Beutel, A., and Steenkiste, P. (2014). Network anomaly detection using co-clustering. *Encyclopedia of Social Network Analysis and Mining*, pages 1054–1068.

Prykhodko, S., Prykhodko, N., Makarova, L., and Pukhalevych, A. (2018). Application of the squared Mahalanobis distance for detecting outliers in multivariate non-gaussian data. In *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)*, pages 962–965. IEEE.

Qin, X., Xu, T., and Wang, C. (2015). DDoS attack detection using flow entropy and clustering technique. In *Computational Intelligence and Security (CIS), 2015 11th International Conference on*, pages 412–415, Shenzhen, China. IEEE.

Rao, M., Chen, Y., Vemuri, B. C., and Wang, F. (2004). Cumulative residual entropy: a new measure of information. *IEEE transactions on Information Theory*, 50(6):1220–1228.

Reid, R. and Van Niekerk, J. (2014). From information security to cyber security cultures. In *2014 Information Security for South Africa*, pages 1–7. IEEE.

Rényi, A. (1965). On the foundations of information theory. *Revue de l'Institut International de Statistique*, pages 1–14.

Roudiere, G. and Owezarski, P. (2017). A lightweight snapshot-based DDoS detector. In *2017 13th International Conference on Network and Service Management (CNSM)*, pages 1–7. IEEE.

Sahi, A., Lai, D., Li, Y., and Diykh, M. (2017). An efficient DDoS TCP flood attack detection and prevention system in a cloud environment. *IEEE Access*, 5:6036–6048.

Santiago-Paz, J., Torres-Roman, D., and Velarde-Alvarado, P. (2012). Detecting anomalies in network traffic using entropy and Mahalanobis distance. In *CONIELECOMP 2012, 22nd International Conference on Electrical Communications and Computers*, pages 86–91. IEEE.

Scholkopf, B. and Smola, A. J. (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Semerci, M., Cemgil, A. T., and Sankur, B. (2018). An intelligent cyber security system against DDoS attacks in sip networks. *Computer Networks*, 136:137–154.

Seo, J., Lee, C., Shon, T., Cho, K.-H., and Moon, J. (2005). A new DDoS detection model using multiple SVMs and TRA. In *International Conference on Embedded and Ubiquitous Computing*, pages 976–985. Springer.

Shameli-Sendi, A., Pourzandi, M., Fekih-Ahmed, M., and Cheriet, M. (2015). Taxonomy of distributed denial of service mitigation approaches for cloud computing. *Journal of Network and Computer Applications*, 58:165–179.

Sharafaldin, I., Gharib, A., Lashkari, A. H., and Ghorbani, A. A. (2018a). Towards a reliable intrusion detection benchmark dataset. *Software Networking*, 2018(1):177–200.

Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018b). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Prog. of the 4th Int. Conf. on Information Systems Security and Privacy (ICISSP)*, Funchal, Portugal.

Thill, M. (2017). The relationship between the Mahalanobis distance and the Chi-squared distribution.

Verizon (2019). Data Breach Investigations Report. https://enterprise.verizon.com/resources/reports/dbir/. [Online; accessed 30-July-2018].

Wagner, C., François, J., Engel, T., et al. (2011). Machine learning approach for IP-flow record anomaly detection. In *International Conference on Research in Networking*, pages 28–39. Springer.

Xiang, Y., Li, K., and Zhou, W. (2011). Low-rate DDoS attacks detection and traceback by using new information metrics. *IEEE transactions on information forensics and security*, 6(2):426–437.

Xu, T., He, D., and Luo, Y. (2007). DDoS attack detection based on RLT features. In *cis*, pages 697–701. IEEE.

YANG, M.-h. and WANG, R.-c. (2008). DDoS detection based on wavelet kernel support vector machine. *The Journal of China Universities of Posts and Telecommunications*, 15(3):59–94.

Yu, J., Lee, H., Kim, M.-S., and Park, D. (2008). Traffic flooding attack detection with snmp mib using svm. *Computer Communications*, 31(17):4212–4219.

Yu, S., Zhou, W., Jia, W., Guo, S., Xiang, Y., and Tang, F. (2012). Discriminating DDoS attacks from flash crowds using flow correlation coefficient. *IEEE Transactions on Parallel and Distributed Systems*, 23(6):1073–1080.

Zhang, X. and Zhong, S. (2009). An improved path-based transductive support vector machines algorithm for blind steganalysis classification. In *International Conference on Artificial Intelligence and Computational Intelligence*, pages 453–462. Springer.

Zhou, L., Liao, M., Yuan, C., and Zhang, H. (2017). Low-rate DDoS attack detection using expectation of packet size. *Security and Communication Networks*, 2017.

APPENDICES

APPENDIX A

Mahalanobis distance values and Chi-square test results

Mahalanobis distance	Time	alpha=0.05	alpha=0.01	alpha=0.001
3.788507	536	0	0	0
3.144352	537	0	0	0
2.119748	538	0	0	0
2.573952	539	0	0	0
2.787513	540	0	0	0
2.770793	541	0	0	0
1.545647	542	0	0	0
2.582686	543	0	0	0
3.798436	544	0	0	0
2.006835	546	0	0	0
2.386099	547	0	0	0
2.40586	548	0	0	0
2.151711	550	0	0	0
1.545746	552	0	0	0
2.46843	555	0	0	0
2.797316	563	0	0	0
1.183986	565	0	0	0
2.584387	568	0	0	0
6.179406	570	0	0	0
2.521087	573	0	0	0
1.885014	574	0	0	0
11.80373	**575**	**1**	**1**	**0**
2.867908	576	0	0	0
9.920155	**585**	**1**	**0**	**0**
10.9671	**596**	**1**	**0**	**0**
5.688897	597	0	0	0
1.490073	598	0	0	0
2.994716	603	0	0	0
5.532619	604	0	0	0
2.599124	605	0	0	0

Mahalanobis distance	Time	alpha=0.05	alpha=0.01	alpha=0.001
5.287621	607	0	0	0
11.75426	**609**	**1**	**1**	**0**
1.09476	611	0	0	0
1.769663	612	0	0	0
1.675633	613	0	0	0
1.95016	615	0	0	0
3.559468	616	0	0	0
2.348398	626	0	0	0
3.600126	629	0	0	0
12.72213	**632**	**1**	**1**	**0**
8.478768	**634**	**1**	**0**	**0**
9.938449	**637**	**1**	**0**	**0**
5.568832	638	0	0	0
5.846324	648	0	0	0
7.182955	651	0	0	0
5.54912	652	0	0	0
6.534555	653	0	0	0
21.00665	**654**	**1**	**1**	**1**
10.37103	**655**	**1**	**0**	**0**
13.0389	**656**	**1**	**1**	**0**
11.74201	**661**	**1**	**1**	**0**
3.047871	662	0	0	0
5.481711	666	0	0	0
3.249266	671	0	0	0
2.096866	674	0	0	0
3.840073	677	0	0	0
9.794947	**678**	**1**	**0**	**0**
2.666256	679	0	0	0
3.296364	684	0	0	0
11.6906	**685**	**1**	**1**	**0**
2.754891	686	0	0	0
2.391558	697	0	0	0
2.665245	701	0	0	0
2.522156	725	0	0	0
3.631692	732	0	0	0
1.999664	753	0	0	0

Mahalanobis distance	Time	alpha=0.05	alpha=0.01	alpha=0.001
1.595494	766	0	0	0
4.176344	767	0	0	0
12.2557	**784**	**1**	**1**	**0**
3.325645	790	0	0	0
7.69481	798	0	0	0
2.860762	800	0	0	0
1.343874	822	0	0	0
5.045717	824	0	0	0
1.664093	828	0	0	0
5.447875	829	0	0	0
11.64197	**832**	**1**	**1**	**0**
5.001577	843	0	0	0
3.058539	846	0	0	0
1.259557	867	0	0	0
2.15972	870	0	0	0
4.324535	875	0	0	0
3.055996	879	0	0	0
1.789773	881	0	0	0
5.645057	882	0	0	0
9.119706	**889**	**1**	**0**	**0**
4.239411	930	0	0	0
3.515205	932	0	0	0
3.397901	933	0	0	0
10.28204	**944**	**1**	**0**	**0**
4.108383	945	0	0	0
5.470661	946	0	0	0
8.495946	**953**	**1**	**0**	**0**
4.140344	954	0	0	0
7.642688	955	0	0	0
37.39175	**956**	**1**	**1**	**1**
42.6255	**957**	**1**	**1**	**1**
37.33347	**958**	**1**	**1**	**1**
37.78027	**959**	**1**	**1**	**1**
39.82963	**960**	**1**	**1**	**1**
39.45025	**961**	**1**	**1**	**1**
41.7446	**962**	**1**	**1**	**1**

Mahalanobis distance	Time	alpha=0.05	alpha=0.01	alpha=0.001
40.32104	**963**	**1**	**1**	**1**
41.72865	**964**	**1**	**1**	**1**
40.75256	**965**	**1**	**1**	**1**
43.35497	**966**	**1**	**1**	**1**
41.47939	**967**	**1**	**1**	**1**
43.14325	**968**	**1**	**1**	**1**
41.9959	**969**	**1**	**1**	**1**
45.1576	**970**	**1**	**1**	**1**
44.44451	**971**	**1**	**1**	**1**
45.84193	**972**	**1**	**1**	**1**
46.76683	**973**	**1**	**1**	**1**
34.67277	**974**	**1**	**1**	**1**
30.04073	**975**	**1**	**1**	**1**
16.03983	**976**	**1**	**1**	**0**
3.300738	977	0	0	0
4.378893	978	0	0	0
4.847064	980	0	0	0
2.132391	983	0	0	0
8.171737	**984**	**1**	**0**	**0**
1.601019	987	0	0	0
2.588164	989	0	0	0
0.941538	994	0	0	0
6.029054	997	0	0	0
4.229866	998	0	0	0
6.402328	999	0	0	0
5.743155	1000	0	0	0
2.662727	1004	0	0	0
8.836709	**1021**	**1**	**0**	**0**
10.98077	**1022**	**1**	**0**	**0**

APPENDIX B

Clustering analysis results for detection statistic δ

Instance No.	Delta	Cluster No.	Instance No.	Delta	Cluster No.
0	0.1	0	30	0.043222	3
1	0.995462	2	31	0.013961	4
2	0.473523	2	32	0.007063	3
3	0.274606	1	33	0.090156	3
4	0.055556	4	34	0.029335	0
5	0.663468	2	35	0.646831	4
6	0.214114	1	36	0.048058	2
7	0.121457	0	37	0.016298	4
8	0.077021	0	38	0.075897	3
9	0.15028	1	39	0.088888	0
10	0.01547	3	40	0.433326	0
11	0.669492	2	41	0.120443	2
12	0.19781	1	42	0.031193	0
13	0.209048	1	43	0.03301	4
14	0.0298	4	44	0.49221	4
15	0.034377	4	45	0.064641	2
16	0.008998	3	46	0.057751	0
17	0.055273	4	47	0.032283	4
18	0.02471	3	48	0.086644	4
19	0.033168	4	49	0.145402	0
20	0.068633	0	50	0.15085	1
21	0.030485	4	51	0.059733	1
22	0.008019	3	52	0.027684	4
23	0.102214	0	53	0.039611	3
24	0.035379	4	54	0.026134	4
25	0.001991	3	55	0.029957	3
26	0.020307	3	56	0.035059	4
27	0.018169	3	57	0.065238	4
28	0.195285	1	58	0.02384	0
29	0.019824	0	59	0.04667	3

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Daneshgadeh Çakmakçı, Salva

Nationality: Turk

Date and Place of Birth: 19.09.1984, Tabriz

Marital Status: Married

Phone: +90 554 911 1120

E-mail: salva.daneshgadeh@gmail.com

Linkedin: https://www.linkedin.com/in/salva-daneshgadeh-415840169/

Google Scholar: Salva Daneshgadeh Çakmakçı
https://scholar.google.com/citations?user=hNvX5LQAAAAJ&hl=en

Research Interest	Network Security, Information security, Intrusion Detection, Machine Learning, Deep learning	
Academic Background	Middle East Technical University, Turkey *Information Systems, Ph.D.* *(Qualification Exam: MAY 2015)*	*2013 - 2019*
	Middle East Technical University, Turkey *Information Systems, MSc*	*2009 – 2012*
	Azad University of Tabriz, Iran *Computer Engineering, BSc*	*2003 – 2007*

Professional Experience

MAY Cyber Technologies, Turkey

Security Data Scientist

- *Network Security*
- *Statistic and Machine learning (MATLAB)*
- *Computer Programming (Python)*
- *Database (SQL, Elasticsearch)*

2016 – Current

Publications

Daneshgadeh, S., Baykal, N., and Karabey, B. (2016). Security issues of smartphones regarding m-commerce. *In Encyclopedia of E-Commerce Development, Implementation, and Management*, pages 1461–1472. IGI Global.

Daneshgadeh, S., Baykal, N., et al. (2017). DDoS attack modeling and detection using SMO. In Machine Learning and Applications (ICMLA), *2017 16th IEEE International Conference on*, pages 432–436, Cancun, Mexico. IEEE.

Daneshgadeh, S., Kemmerich, T., Ahmed, T., and Baykal, N. (2018). A hybrid approach to detect DDoS attacks using KOAD and the Mahalanobis distance. *In 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, Cambridge, MA USA. IEEE.

Daneshgadeh, S., Ahmed, T., Kemmerich, T., and Baykal, N. (2019). Detection of DDoS attacks and flash events using shannon entropy, KOAD and Mahalanobis distance. *In 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 222–229. IEEE.

Daneshgadeh, S., Kemmerich, T., Ahmed, T., and Baykal, N. (2019). An empirical investigation of DDoS and flash event detection using Shannon entropy, KOAD and SVM combined. *In 2019 International Conference on Computing, Networking and Communications (ICNC)*, pages 658–662. IEEE.

Daneshgadeh, S., Oney, M., Kemmerich, T., and Baykal, N. (2019). A simulation environment for cyber-security attack analysis based on network traffic logs. *Modeling and Simulation of Complex Networks*.

Projects	Network Anomaly Detection 1501 – TÜBİTAK Industrial R&D Projects Grant Programme, No.3150972 *Researcher*	*2016-2019*