UNIVERSITY OF JOHANNESBURG

DOCTORAL THESIS

Network Intrusion Detection with Sensor Fusion: Performance Bounds and Benchmarks

---

Author:
Nenekazi Nokuthala
Penelope MKUZANGWE

Supervisor:
Prof. Fulufhelo
NELWAMONDO

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy

in the

Faculty of Engineering and the Built Environment
Department of Electrical and Electronic Engineering Sciences

January 2020

Declaration of Authorship

I, Nenekazi Nokuthala Penelope MKUZANGWE, declare that this thesis titled, "Network Intrusion Detection with Sensor Fusion: Performance Bounds and Benchmarks " and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at the University of Johannesburg.
- No part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all of the main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

_____

Date:

_____

"Strive for continuous improvement instead of perfection."

Kim Collins

Abstract

The achievable performances of intrusion detection systems are unknown beforehand. Currently, intrusion detection researchers implement these systems before they can determine what the performances of their systems will be or compare the performance of their systems to existing systems in order to evaluate the performances of their systems . Another challenge of network researchers is the unavailability of real world traffic traces of network activities due to privacy and legal restrictions.

This Thesis contributes to the literature by

1. presenting the achievable performances of the existing anomaly and learning based network intrusion detection systems (NIDSs) in detecting the Transmission Control Protocol (TCP) synchronised (SYN) flooding attacks. Two anomaly based algorithms, adaptive threshold and cumulative sum based algorithms were considered in building the anomaly based NIDSs. The logic OR operator was used to combine the outcomes of the two anomaly based algorithms to enhance their performance. The three algorithms were used to detect TCP SYN flooding attacks that were synthetically generated according to a Poisson process and constant interarrival times. The logic OR operator performed better than the two algorithms. The three algorithms detected the Poisson process attacks better than the constant interarrival times attacks. For the learning based NIDSs, the decision tree and a novel fuzzy logic based NIDSs were used to detect Neptune, which is a type of a TCP SYN flooding attack. The decision tree outperformed the fuzzy logic system.

2. providing the achievable upper bounds on the accuracies of two ensembles of classifiers based NIDSs. The first NIDS is an AdaBoost based ensemble that uses decision stamp as a base learner. The second NIDS is a Bagging based ensemble that uses a decision tree as a base learner. The obtained bounds will enable researchers to estimate the performance of their ensemble based NIDSs before they implement them and determine how well their ensemble based NIDSs are performing relative to these bounds. From the empirical studies, it was deduced that if the dataset entropy with respect to the features falls between 0.9578 to 0.9586 and the average information gain amongst the features used in the ensemble falls between 0.045615 and 0.25615 then the accuracy of the first NIDS will be at most 0.9065 and the accuracy of the second NIDS will be at best 0.9193. These obtained ensemble accuracy upper bounds hold irrespective of the attack or dataset provided that the features used in the ensemble (AdaBoosted decision stump ensemble or Bagged decision tree ensemble) have the same characteristics as the features used in this Thesis and the features are discretised in the same way as in this work.

3. providing a novel differentially private number of Transmission Control Protocol Synchronise (TCP SYN) packets associated with the Hypertext Transfer Protocol (HTTP) requests to address the issue of unavailability of real world traffic traces of network activities. The utility analysis of the privatised number of TCP SYN packets associated with HTTP requests indicates that the released counts are research useful with the added advantage of preserving privacy and will work well for some algorithms.

Acknowledgements

To God be the Glory for yet another good and perfect gift!

My sincere gratitude goes to Prof Fulufhelo Nelwamondo for his patience, support, guidance, motivation, advice and financial support throughout the duration of this research.

I would like to thank my two mentors, namely, Andre McDonald for introducing me to intrusion detection field and his help in implementing the anomaly based intrusion detection systems and Nyalleng Moroosi for helping me use the information theoretic measures with machine learning and with the application of some of the machine learning techniques. To Dr Graham Barbour, Dr Gugulethu Mabuza-Hocquet, Mbulelo Ntlangu and Sisanda Makinana, thank you for being my sounding boards.

I would like to thank Dr Stephen Moepya, Dr Vukosi Marivate, Tshepiso Mokoena and Abiodun Modupe for helping with the understanding of some of the machine learning techniques and their applications.

Thank you, Prof Sonali Das, for your advice on data handling.

To the Data Mining team in the CSIR Modelling and Digital Science (MDS) unit, MDS students and staff members, thank you for the good laughs that we had that kept me going.

Thank you, Morne Pretorius, for your help with Latex.

I would like to thank the Council for Scientific and Industrial Research (CSIR) and the Department of Science and Technology (DST) of South Africa for funding this research.

To my children Migcobo, Ngakuyo and Nkazimulo, thank you guys for your patience and love. To my friends Vuyokazi Potso and Lulama Kephe , thank you for walking with me on this journey. Your support and motivation encouraged me to persevere.

I am truly grateful to have had the love and support from my parents, siblings, nephews, and niece. You have made this journey easier.

I am sad that this day has come when my grandparents are no more, I know they would have been proud.

List of Publications

From this thesis, the following papers were published:

- Nenekazi N P Mkuzangwe, Andre McDonald and Fulufhelo V Nelwamondo. Implementation of anomaly detection algorithms for detecting Transmission Control Protocol Synchronized flooding attacks. In *Proceedings of the 12th International Conference on Fuzzy Systems and Knowledge Discovery(FSKD)*, pages 2137-2141, Zhangjiajie, China, 15-17 August 2015.
- Nenekazi N P Mkuzangwe and Fulufhelo V. Nelwamondo. A fuzzy logic based network intrusion detection system for predicting the TCP SYN flooding attack. In *Proceedings of the 9th Asian Conference on Intelligent Information and Database Systems (ACIIDS)*, pages 14-22, Kanazawa, Japan, 3-5 April 2017.
- Nenekazi N P Mkuzangwe and Fulufhelo Nelwamondo. Ensemble of classifiers based network intrusion detection system performance bound. In *Proceedings of the 4th International Conference on Systems and Informatics(ICSAI)*, pages 970-974, Hangzhou, China, 11-13 November 2017.
- Nenekazi N P Mkuzangwe and Fulufhelo Nelwamondo. Differentially private Transmission Control Protocol Synchronize packet counts. *International Journal of Network Security*, 21(5):835-842, 2019.

Contents

List of Figures

List of Tables

Abbreviations & Acronyms

AI	Artificial Intelligence
ALAD	Application Layer Anomaly Detector
ANFIS	Adaptive Neural Fuzzy Inference System
ANN	Artificial Neural Network
BPA	Basic Probability Assignment
BPSO	Binary Particle Swarm Optimisation
CART	Classification and Regression Trees
CFS-GA	Correlation based feature selection-genetic algorithm
CUSUM	Cumulative Sum
DARPA	Defence Advanced Research Project Agency
DF	Data Fusion
DNS	Domain Name Server
DoS	Denial of Service
DP	Differential Privacy
DT	Decision Tree
EWMA	Exponential Weighted Moving Average
FAST	Filtering and Adaptive Sampling for Differential Private Time Series Monitoring
FOD	Framework of Discernment
FRM-SFM	Feature Removal Method-Sole Feature Method
FTP	File Transfer Protocol
GFR	Gradual Feature Removal
GMM	Gaussian Mixture Models
HIDSs	Host based Intrusion Detection Systems
HTTP	Hypertext Transfer Protocol
IBK	Instance Based Learning with Parameter K
IDS	Intrusion Detection System
IDSDFM	Intrusion Detection System Data Fusion Model
IDSs	Intrusion Detection Systems
IG	Information Gain
IoT	Internet of Things
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
KDD	Knowledge Discovery in Databases

kNN	k-Nearest Neighbour
LGP	Linear Genetic Programming
MAC	Media Acess Control
MCS	Multiple Classifier System
MLP	Multi-Layer Perceptron
MQTT	MQ Telemetry Transport
NB	Naïve Bayes
NIDS	Network Intrusion Detection System
NIDSs	Network Intrusion Detection Systems
NN	Neural Network
NSL	Network Socket Layer
PHAD	Packet Header Anomaly Detector
PLSR	Partial least square regression
R2L	Root to Local
RBF	Radial Basis Function Neural Network
RDPSO	Rough-Discrete Particle Swarm Optimisation
RF	Random Forest
RMSE	Root Mean Square Error
SOM	Self Organising Map
SVM	Support Vector Machines
TCP	Transmission Control Protocol
U2R	User to Root
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network

I dedicate this research to my children Migcobo, Ngakuyo and Nkazimulo, my mother Nomfundo and my grandfather Ngubonde…

Chapter 1

Introduction

1.1 Introduction

Businesses, governments [1] and society increasingly rely on computers and the internet for communication, information storage, information exchange and other online services. These services are vulnerable to cyberattacks that attempt to compromise the integrity, availability or confidentiality of an information resource. From a user's perspective cyberattacks result in the unavailability of services, theft of personal information, identity, and fraud. In 2011 Sony 's PlayStation network was shut down by Lulzsec [1]. The hack affected 77 million accounts, the attackers stole valuable personal client information like logins, passwords, names, emails, purchase history, home addresses and credit cards [1]. Sony reportedly lost almost $171 million and was not insured against this risk [1]. The frequency and severity of cyberattack incidents have increased due to increased access to computers and attack tools (available online) and highly charged geopolitical events [2], to mention a few. Advanced network security measures are therefore needed to protect information systems against these threats or attacks.

A secure networked computer system ensures that the confidentiality, integrity, as well as the availability of data and resources, are preserved [3]. The traditional security mechanisms that are used include firewalls and cryptography [4], however, in modern networks, these mechanisms are no longer able to completely secure a network of computers from external threats [4]. Several reasons like network scale, software complexity, maintenance complexity, traffic rate and complexity, more sophisticated attacks and interconnections are responsible for this. These mechanisms also do not protect the network against internal threats as they do not traverse a firewall. To address the shortcomings of tradition mechanisms, intrusion detection systems (IDSs) have been proposed for safeguarding modern computer networks against both external and internal threats.

Intrusion detection systems are hardware systems or software applications that attempt to identify attacks that are conducted against information systems, and may

be classified according to the source of those events that they monitor, for example, network events or host events. According to Hu and Hu [5], network events consist of IP package information collected by the network hardware such as routers and switches and host events are made up of audit data that is collected from the target host machine. The effectiveness of an IDS is evaluated based on its ability to correctly classify events to be an attack or normal network behaviour [6].

The reliability and the accuracy of IDSs have been reduced by the increasing network traffic complexity and the growing sophistication of mechanisms used to attack the network. Standalone IDSs are famous for generating a lot of alerts with many of them being false positives or of low importance [7]. According to Bhatti et el. [8], the major causes of false positives are noise, incomplete data, spurious and duplicate data, lack of knowledge about the network topology and multiple log formats. Default IDS configuration is also a cause of false positives. Standalone IDS with high false positive rates are of little value and need to be improved in terms of accuracy and speed.

Different techniques for improving the performance of intrusion detection systems have been proposed in the literature [7], [9]–[13]. Research shows that IDSs tend to show preference in detecting an attack with certain attributes with improved accuracy while performing poorly or moderately on the other attack categories. With the continuing improvements in computing power, it has become possible to implement different types of IDSs that have different detection capabilities on the same network. It has been shown that combining the decisions of multiple IDSs enables a more reliable and accurate decision for a wider class of attacks [14]. An ensemble of classifiers and sensor fusion techniques have been developed to answer the question of how to effectively combine multiple sensor outputs to perform classification. The key concept of sensor fusion is to aggregate information from IDSs and to infer the state of the network (intrusion or normal). In ensemble methods, base classifiers are combined in a certain way to yield a strong classifier that outperforms all the base classifier in the ensemble.

Work that has been done on sensor fusion has been conducted mainly using one of the methods like probability theory (Bayesian theory), evidence theory (Dempster Shafer belief theory), voting fusion theory, fuzzy logic theory or machine learning (neural network) to aggregate information [15] and [16]. Dempster-Shafer belief theory is commonly used to perform inference in intrusion detection systems that make use of sensor fusion [17]. Most research and discussions on IDS that make use of sensor fusion [11], [12], [18]–[20] focus on improving intrusion detection using advances in sensor fusion or on achieving the best fusion performance. A common trend is that researchers estimate detection performance through simulations of the system.

The idea behind constructing an ensemble of classifiers is to build a stronger

classification capability that has a better performance than the individual classifiers that make the ensemble. It has been shown that combining classifiers can lead to a classifier that performs better than a single best classifier. It has been proven that a set of weak learners can be boosted to a strong learner. Although strong learners are desirable they are difficult to get while weak learners are easy to obtain. This resulted to the generation of strong learners by ensemble methods.

However, the achievable performances of these systems are not known before they are implemented. Researchers have to implement the systems before they can determine what the performance of their systems will be.

Besides the various techniques for improving the performance of intrusion detection systems that have been proposed by researchers, advancement in network research (including intrusion detection) depends crucially on the availability of real world traffic traces of network activities. Unfortunately, real world network traces release is highly restricted by privacy and legal issues. Organisations are not willing to share their traces since raw network traces may consist of sensitive information that should not be publicly shared, for example, information that identifies individuals, patterns of the traffic that can be analysed to determine strategies of organisations, hints to the weaknesses of a system, etc. [21]. Unavailability of raw network traces to researchers poses a risk of developing models that compromise accuracy.

To continue with their activities, researchers end up simulating data or signing non-disclosure agreements and these two ways of obtaining data may compromise accuracy and repeatability of the research [22]. The traditional solution to this trace problem is the use of anonymised data. However, anonymisation is vulnerable to attacks that infer sensitive information [23]. Mogul and Arlitt [21] proposed an alternative approach to trace anonymisation where data owners perform the analyses in the place of the researchers to preserve privacy, privacy is preserved in this approach based on human verification which is likely to make error [24]. This indicates that the existing approaches provide no guarantee in protecting sensitive information and therefore a formal privacy guaranteeing approach, that will make data owners comfortable to adopt before releasing their data, is needed [24].

This research aims at empirically determining the achievable performance bounds of IDSs that uses sensor fusion or ensemble of classifiers for event aggregation. The knowledge of these bounds would help researchers with designing the system without relying only on the simulation of the system. Moreover, these bounds would allow researchers to determine how far the current sensor fusion or ensemble of classifiers based IDS performance falls short of what is achievable. This work also aims at the use of differential privacy as a means of providing privacy to network trace in order to encourage trace owners in different organisations to release their traces for use in intrusion detection.

1.2 Problem Description

With the continuing improvements in computing power, it has become possible to implement different types of IDSs that have different detection capabilities on the same network. Methods like an ensemble of classifiers and sensor fusion have been successfully implemented in intrusion detection to enhance the performance of IDSs. However, the achievable performances of these systems are not known before they are implemented. Researchers have to implement the systems before they can determine what the performances of their systems will be. This work aims at providing the achievable performance bounds of network intrusion detection systems that are based on an ensemble of classifiers that uses dependent and independent base classifiers. The knowledge of these bounds would help researchers estimate the performance of their ensemble of classifiers based network intrusion detection systems (NIDSs) before they even implement them. These bounds will help developers to design their IDS to reach a particular performance.

Apart from improving the performance of IDSs, the availability of real network traffic data is important for network research. However, privacy and legal issues restrict the release of such data. To solve the trace problem, researchers have proposed the anonymisation of data. However, anonymisation is prone to attacks that infer sensitive information [23]. Therefore, a formal privacy guaranteeing approach, that will make data owners comfortable to adopt before releasing their data, is needed. This research aims at the use of differential privacy as a means of providing privacy to network trace.

1.3 Thesis Statements

- Achievable upper bounds on the performance of Network Intrusion Detection Systems (NIDS) that make use of multiple sensor combining techniques can be determined via an information theoretic approach with given sensor specification. Two information theoretic measures, information gain of the features used in building the NIDS and dataset entropy with respect to the features (referred to as dataset entropy in this study), can be used to define these bounds under two sensor specifications, namely, a NIDS that combines multiple independent sensors and a NIDS that combines multiple dependent sensors.
- Privacy-preserving publishing of the number of Transmission Control Protocol Synchronise (TCP SYN) packets associated with Hypertext Transfer Protocol (HTTP) requests can be achieved via differential privacy. Differential privacy is a formal privacy-preserving technique for publishing data that guarantees

that the absence or presence of an individual is hard to infer from the output of the analysis.

1.4 Research Motivation

The motivation for investigating the first Thesis statement is that currently researchers rely on comparing the performance of their newly developed NIDSs to the performance of existing NIDSs in order to determine how good they are. This way of gauging performance does not establish if the new NIDS performance is optimal or not, all the researcher learns from it is that their new NIDS performs better or worse than the existing NIDSs. Determining the NIDS's achievable performance bound is important because it will provide researchers with a performance reference point that is optimal and researchers can gauge the performance of their new NIDS against this bound. From this bound researchers can determine if their NIDSs need improvement or not. The second Thesis statement is investigated since privacy preserving network trace that retains the research usefulness of the network trace is needed since the unavailability of raw network trace (due to privacy and legal issues that restrict the release of such data) poses a risk of developing models that compromise accuracy.

1.5 Research Aim

This Thesis aims at addressing the following gaps

- Find the achievable accuracy upper bound and its corresponding detection and false positive rates of an ensemble based NIDS that has dependent base classifiers.
- Find the achievable accuracy upper bound and its corresponding detection and false positive rates of an ensemble based NIDS that has independent base classifiers.
- Implement differentially privacy on, TCP SYN packet counts, a different network trace than in literature.

1.6 Rerseach Objectives

The Thesis statements are investigated using the following objectives:

1. To evaluate the performance of anomaly and learning based NIDSs in detecting the TCP SYN flooding attack.

2. To empirically determine the achievable performance bound on how well a NIDS that uses an ensemble of classifiers with dependent base classifiers can detect whether a network is under attack or not.

3. To provide the true positive and false positive rate associated with the bound in objective 1.

4. To empirically determine the achievable performance bound on how well a NIDS that uses an ensemble of classifiers with independent base classifiers can detect whether a network is under attack or not.

5. To provide the true positive and false positive rate associated with the bound in 2.

6. To provide research useful differentially private number of TCP SYN packets associated with HTTP requests.

1.7 Contribution, Assumptions, Delimitations and Roadmap

The contribution of this Thesis is to

- present the achievable performance of existing individual NIDSs and combined multiple NIDSs in detecting the TCP SYN flooding attack.
- provide achievable performance bounds on how well a NIDS that uses an ensemble of classifiers can resolve uncertainty about whether a network is under attack or not. The utility of this bound is to provide insight into the fundamental limits of the performance of NIDS as a function of its design, i.e. ensemble based NIDS built using independent or dependent classifiers. These bounds can be used to quantify how far the performance of ensemble based systems fall short of what is achievable. These bounds are the first to be defined in terms information gain and dataset entropy.
- improve the literature by enabling the NIDS designers to determine what is required of a NIDS, in terms of its true positive and false positive rates, to give a certain level of performance.
- provide a novel differentially private Transmission Control Protocol Synchronise packet counts.

The assumptions of this study are that

- the sensors that are used in the AdaBoosted ensemble are dependent.
- the sensors that are used in the Bagging based ensemble are independent.

Sensors are independent if the joint probability density/mass function of their outcomes on any class i is equal to the product of the marginal density/mass functions of their outcomes on any class i, otherwise, they are dependent. Only the

- ensemble based NIDS that uses dependent decision stumps as base classifiers.
- ensemble based NIDS that uses independent decision trees as base classifiers .

are investigated in this study.

The rest of this work is organised as follows:

Chapter 2 provides an intrusion detection and network trace privatisation background and the review of the studies that used multiple sensors combining techniques to enhance the performance of IDSs and network trace privatisation techniques. Specifically, the literature on sensor fusion and an ensemble of classifiers based techniques that have been successfully applied in intrusion detection and network trace privatisation techniques is reviewed.

Chapter 3 describes the methodology and data that will be used in the Thesis.

Chapter 4 presents the implementation of some of the intrusion detection algorithms for detecting intrusions in a network to illustrate the performance of the current intrusion detection systems. The outcomes of the two of the existing intrusion detection algorithms are fused to illustrate how combining multiple sensors improve the detection rate.

Chapter 5 provides the achievable performance bounds of two network intrusion detection systems (NIDSs) that use an ensemble of classifiers. The performance bounds are defined in terms of the dataset entropy and the average information gain associated with the features used in building the ensembles. The performance bounds are obtained by using ensemble method algorithms, AdaBoost and Bagging.

Chapter 6 presents the use of differential privacy as a means of providing privacy to network trace, improves the accuracy of the released aggregates by adopting the filtering component of [25] and test the utility of the released data by using two utility metrics and comparing the performances of two anomaly based intrusion detection algorithms on the original aggregates and the released aggregates.

Chapter 7 concludes the Thesis with the summary of findings and suggestions on the direction in which the work can be extended.

Chapter 2

Background and Literature Review

This Chapter provides the background of intrusion detection and network trace privatisation. The literature review on sensor fusion techniques and ensemble of classifiers methods used in intrusion detection and network traffic trace privatisation techniques is also presented in this chapter.

2.1 Intrusion Detection

An intrusion is defined as an unauthorised access on or unauthorised attempt to access a computer or an information system [9] and [14]. Intrusion detection is the process of identifying whether an intrusion has been attempted, is occurring or has occurred [26]. Intrusion detection systems are systems that attempt to identify attacks that are carried out against information systems. IDSs are a crucial element of network security infrastructure and play a vital role in detecting intrusions by monitoring the network traffic or host activities looking for evidence of intrusive behaviour [7]. IDSs may be classified according to the source of those events that they monitor. Specifically, IDSs can be classified as host-based IDSs (HIDSs) and network-based IDSs (NIDSs). HIDSs detect attacks against a specific host by analysing audit data produced by operating systems, whereas NIDSs detect attacks by analysing network traffic transmitted, received or forwarded on a network link. According to Hu and Hu [5] HIDSs can achieve high detection rate and low false alarm rates since the information provided by the audit data can be highly comprehensive and elaborate. However, the host-based approaches have the following drawbacks [5]:

- They cannot easily prevent attacks when an intrusion is detected or the attack has partially started.
- Attackers may alter the audit data, hence, influencing the reliability of audit data.

Network-based approaches detect intrusions using the information from the Internet Protocol (IP) packages collected by the network hardware such as switches and routers [5]. Therefore, network-based approaches have the following advantages [5]:

- They can detect distributed intrusions over the whole network, making intrusion detection easier for the individual host machines.
- NIDSs can also protect the host machines since detection is done before the data reaches the host machine.

Due to these advantages of NIDSs over the HIDSs, this research focuses on studying NIDSs.

2.1.1 Network Intrusion Detection Systems (NIDSs)

Network intrusion detection systems are software applications or hardware systems dedicated to detecting intrusions in a target network [4] and [18]. NIDS sensors gather information from the network, processes the data and send information regarding the state of the network to a central node. The node then analyses the information to identify possible security breaches against the system or the network.

A standard NIDS monitors network traffic on a target network while attempting to detect malicious activities [17]. In general, NIDSs attempt to identify attacks carried out over the network that targets some end-node as well as attacks that target the network itself. The NIDS's purpose is to reliably detect (in terms of the detection probability) and/or classify attacks in as short a time span as possible (i.e. the detection delay), while maintaining a low false positive rate.

Network traffic is typically nonstationary and bursty, with daily and monthly trends [27]. A large number of connections between nodes (i.e. the connectivity factor) and the use of packet oriented protocols (such as IP) introduce statistical correlation between packets transported over a network. Furthermore, commonly used connection-oriented protocols (such as Transmission Control Protocol (TCP)) introduce correlation between network packets transmitted at different time instances. Network traffic is also feature rich, with a large number of protocols being used in modern networks, in addition to a multiplicity of fields in packet headers. The complexities and variability of network traffic are major challenges to realising effective NIDSs [18].

Network intrusion detection techniques are typically classified as being misuse-based or anomaly-based [4]. In misuse-based detection techniques, a signature of an attack is compiled or specified, after which network traffic is monitored for an occurrence of the attack. Figure 2.1 depicts how a misuse-based NIDS works. Anomaly-based detection techniques define a baseline of normal network

behaviour, and monitor the network for any behaviour that deviates from the baseline. Figure 2.2 illustrates how an anomaly-based NIDS works. These techniques are complementary in their strengths and weaknesses, misuse based IDSs have a very low false positive rate but they cannot detect previously unknown attacks (i.e. zero-day attacks, or even simple variations of existing attacks). Anomaly-based IDSs are able to detect certain previously unobserved attacks but suffer from a higher false positive rate than misuse-based IDSs.

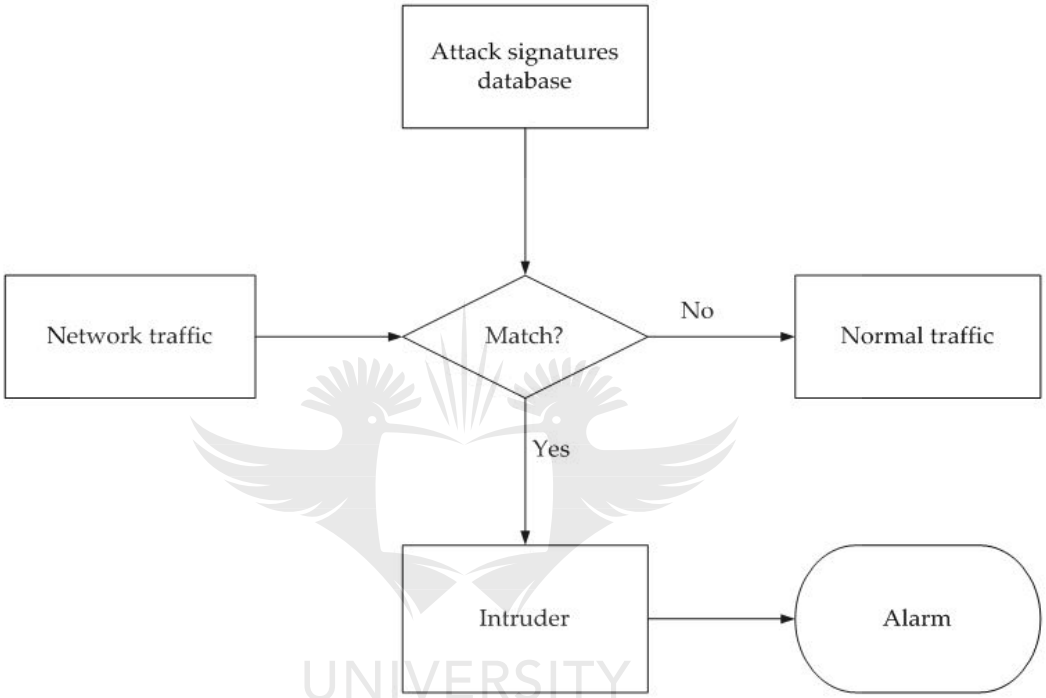


FIGURE 2.1: A signature based network intrusion detection.

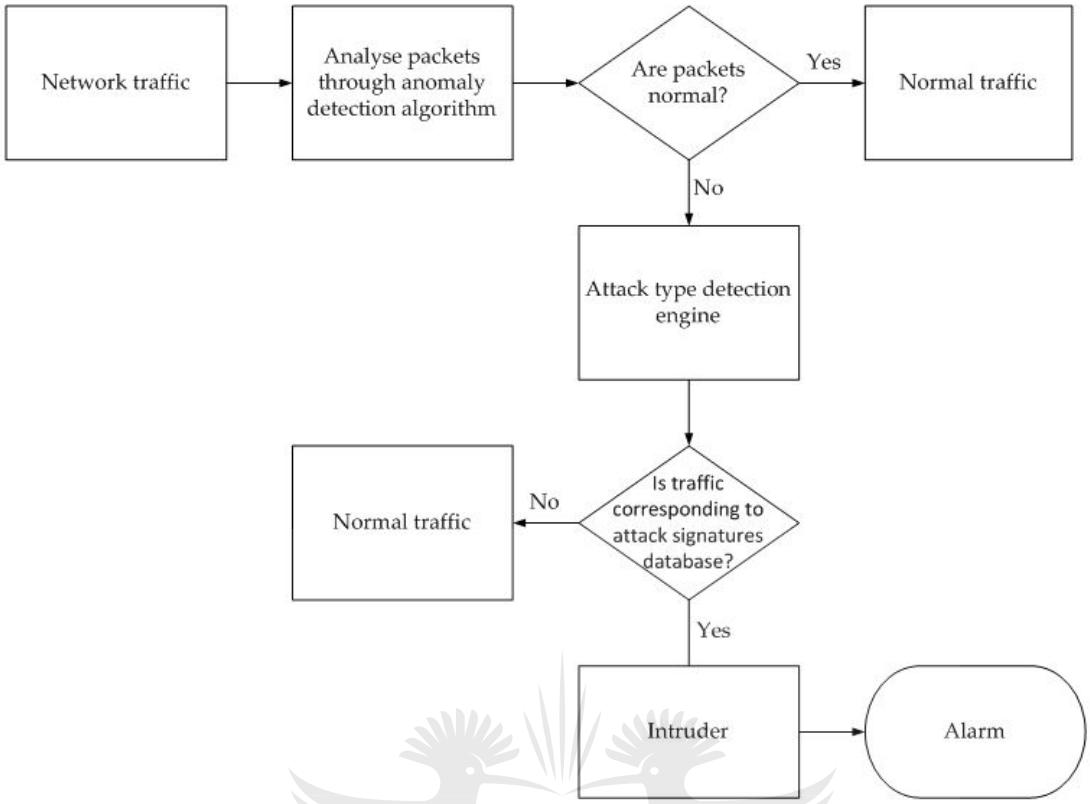


FIGURE 2.2: An anomaly based network intrusion detection.

Due to the complexity of the network traffic, NIDSs that use a single sensor cannot give a complete runtime situational awareness of a complex system [7], [26]. Furthermore, the literature reveals that IDSs detect one class of attack with improved accuracy while performing moderately or poorly on the other classes. These have led designers to use multiple distinct sensors in modern NIDSs in order to increase the range of attack classes that are reliably and accurately detected. Sensor fusion and ensemble of classifiers techniques have been developed to combine multiple sensor outputs to perform classification. The next two sections review the literature on sensor fusion techniques and ensemble methods in intrusion detection.

2.1.2 Literature on Sensor Fusion Techniques in Intrusion Detection

Sensor fusion combines inputs from many sources of limited accuracy and reliability to give information with greater accuracy and reliability. In intrusion detection, sensor fusion techniques are used to combine outputs of different IDSs to yield one output that is more accurate than the outputs of the single IDSs. The reviewed literature developed their sensor fusion based NIDSs using the following sensor fusion techniques: Dempster Shafer evidence theory, majority voting, Naive Bayes, averaging, Neural Network and weighted averaging, logic OR operator.

The developed sensor fusion based NIDSs performed equally or better than the corresponding individual NIDSs.

Thomas et al. [11] presented an architecture using data dependent decision fusion. The method gathers an in-depth understanding about the input and also the behaviour of the individual IDS by means of a feed forward back propagation neural network supervised learner unit. The output of the neural network are weights assigned to each IDS based on each IDS output and input data. The fusion unit performed the weighted aggregation of the IDS outputs for the purpose of classifying the input data. According to the authors the proposed architecture does not depend on the data and the structure used guarantees improved performance in terms of detection rate and false alarm rate, works well on large data set, can identify novel attacks since the rule is dynamically updated and has improved scalability. They evaluated the proposed architecture using DARPA dataset and three intrusion detection systems, packet header anomaly detector (PHAD), application layer anomaly detector (ALAD) and Snort. The following performance measures were used precision, recall, accuracy, detection performance, F-Score and receiver operating characteristic (ROC) curve. The data dependent decision fusion IDS achieved a detection rate of 0.68 and F-score of 0.50 as compared to PHAD, ALAD and Snort with detection rates of 0.35, 0.38 and 0.09 respectively and F-scores of 0.31, 0.35 and 0.15 respectively. The results indicate the proposed approached shows improved performance as compared to individual IDSs.

Thomas et al. [11] architecture incorporates the reliability (weights) of an IDS to the fusion rule which improved the performance of their fusion based IDS, however, the limitation to their architecture is that the assignment of weights to the IDSs depends on supervised learning. Thus, if the target outputs are unknown the weight assignment strategy will have to be modified. Therefore, a weight assigning strategy that can handle both supervised and unsupervised learning is needed.

Tian et al. [12] presented a fusion model called Intrusion Detection System Data Fusion Model (IDSDFM). The model merges alerts from the different IDSs and performs inferences by applying the Dempster-Shafer (DS) evidence theory in which the alerts are used as evidence. After the current network security status has been estimated, some of the IDSs are adjusted dynamically and new rules are added to the IDS feature database. According to Tian et. al. [12], adjusting the IDS and adding of new rules, improves its detection efficiency and reduces false positive and false negatives rates. Their [12] alert correlation strategy assigns similar alerts i.e. alerts with dissimilarity(degree of difference between alerts) close to zero, to the same alert track(a set of alerts which relate to an attack event). This means their model did not capture cases where similar alerts describe different attack events or different alerts describe the same attack event.

Thomas et al. [13] used Chebyshev inequality principle to set threshold bounds

for sensor fusion and simple rule based fusion in order to improve sensor fusion performance. They also attempted to prove the advantage of sensor fusion over individual IDS. Thomas et al. [28] criticised the rule based fusion system of [13] arguing that it only works with small input data and it is dependent on the IDSs that are used in that particular sensor fusion. In their [13] attempt to prove the advantage of sensor fusion over individual IDS, the result indicated that sensor fusion incorporating both Chebychev inequality threshold bounds and rule based fusion perform better than individual IDSs in terms of detection rate. Even though their [13] sensor fusion based IDS detection rates for the R2L and U2R attacks, of 49% and 27% respectively, are better or equal to those of the best performing individual IDS they are still low. Thomas et al. [13] claims, based on investigating feature relevance in the 1999 DARPA IDS evaluation dataset, that this poor detection of the R2L and U2R attacks is due to the fact that the features that are used to characterise these attacks do not discriminate them much. This means, apart from using sensor fusion advances to improving IDS performance, features that discriminate these attacks well are still needed in order to improve the detection rate of these attacks.

Gong et al. [20] presented a neural network based data fusion model. In the model a back propagation neural network was designed to perform the fusion and intrusion detection. In their work a pruning algorithm is used to prune redundant connections and neurones provided that the pruning process will not decrease the processing ability of the neural network. The pruning process is supposed to improve the network in terms of generalisation and reduction of computational cost [29] and hence make the IDS more efficient. Gong et al. [20] used the KDD 99 dataset in this study. They considered four types of attacks from the data, namely, Neptune, Smurf, Satan and Portsweep. The results show that the model yields high overall detection accuracy of 83.4% before pruning and 82.0% after pruning. The drawback in their work [20] is that they have adopted the misuse detection approach at the output layer of the neural network to decide on whether a certain attack is occurring or not. The use of misuse detection method is a drawback due to the fact that the method cannot detect new attacks since signatures of these new attacks would not have been defined. This means that their IDS may not be able to detect any new or slight variations of the four types of attacks they have considered.

Chan et al. [30] did a comparative study on adopting different fusion strategies for a multiple classifier system (MCS) in a denial of service (DoS) problem. The comparison was done on these fusion methods: Dempster–Shafer combination, majority vote, weighted majority vote, naïve Bayes combination, average and a neural network (radial basis function neural network). KDD 99 dataset was used in this study to evaluate the performance of the different fusion methods. They considered normal and DoS attacks from the dataset. Six DoS attacks were chosen from the dataset and used in the study. These were Surmf, Neptune, Back, Teardrop, Land and Ping of Death attacks. Accuracy and false alarm rate are used as a

performance measures and 10 fold cross-validation is adopted. The results of the different fusion methods are averaged. The neural network and Dempster-Shafer combination provided the best performance with high average accuracy and low average false positive rate. The majority vote, weighted majority vote and the average gave comparable performance with their average accuracy being approximately 20% lower than the Dempster-Shafer and the neural network and an average false positive rate of approximately 2% higher than the Dempster-Shafer and the neural network. The naïve Bayes combination was the worst performing method of them all. The architecture of their MCS is such that each base classifier is assigned to detect one of the six DoS attack types considered. The authors also pointed out that the design of their architectures is such that they can add a classifier for detecting novel attacks in the future which means currently their architecture does not cater for novel attacks. This means the classifiers and fusion methods may be underperforming.

Kaliappan et al. [31] proposed fusion of heterogeneous IDSs for detecting network attacks. Their sensor fusion based NIDS consists of five detection algorithms with three of them following the anomaly detection approach and two of them being signature based. The anomaly based algorithms are the support vector machines (SVM), instance-based learning with parameter k (IBK) and random forest (RF). The signature based algorithms are the J48 and BayesNets. They used the NSL KDD dataset to evaluate their proposed system. The proposed system architecture is made up of three phases. In the first phase, features are selected using information gain and the genetic algorithm. In the second phase, the selected features serve as inputs to the five IDSs and the IDSs classify the input as one of the attack categories found in the NSL KDD dataset (probe, DoS, U2R or R2L) or normal. In the third phase, the input data is categorised as either attack or non-attack. The traffic attacks (probe, DoS, U2R or R2L) are labelled as attack group and the normal traffic as non-attack group such that if an IDS outcome is DoS then that outcome is labelled as an attack. The outputs of the IDSs are combined by majority voting to obtain the final decision. The results indicate that the proposed system performed better than individual IDS. They [31] also compared their work to the work of Thomas and Balakrishnan [28] and their [31] work outperformed the work of Thomas and Balakrishnan [28]. Kaliappan et al. [31] rely on comparing the performance of their system to that of other works to gauge the performance of their [31] system. If NIDS performance bounds were known then they [31] would be able to determine how far their NIDS is from these bounds in order to decide whether their NIDS needs improvement or not.

Shah et al. [32] discussed the limitations of Dempster-Shafer (DS) rule in combining decisions/evidence from multiple intrusion detection systems and proposed a modified framework for fusing these decisions. According to Shah et al. [32], most fusion rules proposed in literature including DS rule assume all evidence provided

by the fused IDSs to be equally reliable and weigh them equally during the fusion process. However, some IDSs are more reliable than others [32], where reliability of an IDS is defined as a level of trust about the evidence provided by the IDS for the presence of an intrusion that lies between 0 and 1 [33]. Therefore, their [32] framework incorporates the reliability value for each fused intrusion detection systems in the fusion rule. The results show that fusing with the proposed rules leads to improved performance as compared to fusing with DS rule. The main problem with the proposed fusion rule is assigning the reliability value of each IDS to be fused. They [32] proposed two ways of assigning the IDS reliability value, namely,

- using the true positive rate (TPR). The drawback of using the TPR is that the ground truth needs to be known.
- using the conflicting evidence amongst the IDSs since it can work without knowing the ground truth [32]. However, assigning reliability using IDSs conflicting evidence might be subjective (what reliability score gets assigned to all agreeing or disagreeing IDSs?). In their [32] extended work [34] they proposed that a highly conflicting IDS should be assigned least reliability value and a least conflicting IDS be assigned the highest reliability score. This is still subjective since assigning reliability values for IDSs that are not so highly conflicting and those that are least conflicting will depend on a individual.

The proposed framework that incorporates IDS reliability improves the performance of the fused IDS. However, objective ways that do not depend on the knowledge of the ground truth are needed to assign the IDSs reliability values.

Mkuzangwe et al. [35] fused the outcomes of two anomaly based intrusion detection algorithms for detecting the Transmission Control Protocol Synchronised flooding attacks using the logic OR operator. The two algorithms are the adaptive threshold algorithm and the cumulative sum algorithm of Siris and Papagalou [36] and their outcomes were fused in order to enhance their detection probability. The attacks were synthetically generated in accordance to constant rate arrivals. The DARPA dataset was used to implement the two algorithms. The results indicate that the logic OR operator outperformed the two algorithms in terms of detection probability and detection delay while it had the worst false alarm rate.

Li et al. [37] presented a review of data fusion (DF) techniques used in building network intrusion detection systems. They reviewed the DF techniques in the feature fusion and decision fusion layers of the data fusion levels since they [37] have observed that most research in NIDSs uses open dataset, therefore tend to omit the data fusion layer. In the review, the authors proposed a general data fusion framework for NIDSs and a criteria for evaluating the performance of these DF techniques based NIDS. The proposed framework consists of six parts: input or data source, data pre-processing, feature fusion, classification, decision fusion and output

or decision. The proposed performance evaluation criteria evaluated the reviewed NIDS based on dataset utilised, validity, efficiency, data security and scalability. To test the efficiency of the NIDS, they first compared the feature fusion techniques of the reviewed NIDS in terms of training time on different datasets. According to the authors the following feature fusion techniques had a minimum training time:

- gradual feature removal (GFR) method, feature removal method-sole feature method (FRM-SFM) and classification and regression trees(CART) for the KDD dataset variants (DARPA99, KDD99 and KDD99_10%).
- correlation based feature selection-genetic algorithm (CFS-GA) for the NSL-KDD dataset.
- partial least square regression (PLSR) for the Kyoto 2006^{+} dataset.

To test the validity of the NIDSs, the authors compared the feature fusion techniques in terms of accuracy, precision, recall, F measure, false positive rate and false negative rate. The review provides the techniques that led to the highest accuracy and low FPR for the different dataset. The authors notice that accuracy of classifiers in the UNSW-NB15 dataset was not so good. According to the authors all the reviewed studies did not consider data security since they were using publicly available data and scalability was not mentioned in the studies. According to [37] the evaluation of the decision fusion indicated that the Dempster-Schafer evidence theory (DS), neural networks (NN), random forest (RF), data dependent fusion (DD) and adaptive boosting (AdaBoost) performed well in fusing the multiple decisions in terms of accuracy and false positive rate in the KDD dataset variants.

Some of the open issues that were highlighted from this review were:

- Data security is not considered in the existing DF techniques.
- There are few studies that have considered data fusion visualisation.

The authors have proposed the following future works:

- To find advanced DF techniques that will improve the detection of attacks on complex data sets like UNSW_NB15 dataset.
- Investigation of DF techniques that are effective and adaptive to big network data.
- Studies on a universal, flexible and extensible fusion framework are needed.
- Data security needs to be ensured in data fusion.
- Data layer fusion is an important part in conducting real time network intrusion detection.

2.1.3 Literature on Ensemble Methods in Intrusion Detection

In ensemble methods, learning algorithms are used to construct a set of classifiers whose predictions are combined in a certain way (commonly by weighted or unweighted) to classify a new instance. It has been discovered that ensembles are more accurate than the individual classifiers that make up the ensemble. Ensemble methods have been successfully adopted in intrusion detection to build a classifier that is more accurate than the classifiers in the ensemble and below is the review of the works that have employed the ensemble methods in intrusion detection.

Hu and Hu [5] proposed an AdaBoost based algorithm that uses decision stumps as weak classifiers for network intrusion detection. They adopted simple strategies for removing noise and avoid overfitting, since the AdaBoost algorithm is sensitive to noise and sometimes it overfits, to improve the performance of the algorithm. Hu and Hu [5] modified the initial weights assigned to samples from a uniform distribution to adjustable initial weights since in AdaBoost theory the uniform distributed initial weights focus on reducing the mean classification error which is not suitable for intrusion detection according to [5]. Hu and Hu [5] say in intrusion detection it is necessary to reduce the false alarm rate instead of the mean error. They provided decision rules for both categorical and continuous features of the KDD 99 data set. They considered all attacks as a single category. They reported their results in terms of detection and false positive rates for the standard AdaBoost algorithm and the proposed AdaBoost based algorithm. The detection and false alarm rates for the standard AdaBoost algorithm were 90.738% and 3.428%. They reported a detection rate ranging from 90.04% to 90.88% with a false alarm rate lying between 0.31% and 1.79% for the proposed AdaBoost based algorithm. They also compared the performance of their algorithm with published intrusion detection algorithms in terms of detection rate, false alarm rate and computational times. Their algorithm outperformed all the published algorithms in terms of false positive rate and computational time.

Borji [38] proposed a combining classification approach for intrusion detection. He implemented four base classifiers, namely, artificial neural network (ANN), support vector machines (SVM), k-nearest neighbour (kNN) and decision trees (DT). The outputs of the four base classifiers were fused using three combining strategies: majority voting, Bayesian averaging and a belief measure. Different combining strategies to illustrate the effect of combining approaches were used. Detection and false positive rates were used to measure the performance of the proposed system. The results showed that the proposed approach outperformed the individual classifiers and that the belief measure was a better combining approach followed by the Bayesian averaging then the majority voting. Borji's work illustrates how ensemble methods enhance the performance of individual classifiers and the effect of combining approaches on the performance of the ensemble. However, the

work does not give insight on the extent to which the ensemble methods are able to improve the performance of the individual classifier irrespective of the combining approach.

Govindarajan and Chandrasekaran [39] presented a new arcing based hybrid classifier that is made up of heterogeneous classifiers. They use radial basis function neural network (RBF) and support vector machines (SVM) as base learners. The proposed hybrid intelligent intrusion detection system consisted of four main phases, namely, data pre-processing, feature reduction, classification and combination phases. They used the NSL KDD dataset to train and test their base learners and the hybrid classifier. Classification accuracy was used as the performance metric and was evaluated using 10-fold cross validation. The results indicate that the hybrid classifier outperforms the individual base learners.

Sornsuwit and Jaiyen [40] focused on building an ensemble for detecting network intrusion data that are difficult to detect. Specifically, they proposed AdaBoost based ensembles of a single weak learner for the prediction of U2R and R2L attacks. They implemented AdaBoost ensembles from each of the following weak classifiers, naïve Bayes (NB), decision tree (DT), multilayer perceptron (MLP), support vector machine (SVM) and k-nearest neighbor (kNN). They used the KDD 99 dataset to train and test the ensemble. They build AdaBoost ensembles of these weak learners using all the features in the dataset. To improve the performance of their ensembles they used correlation based feature selection and built the ensembles using the selected features only. Sensitivity and specificity were used as performance measures.

For the AdaBoost based ensembles with all features, the MLP ensemble produced the highest sensitivity of 0.1753 and the NB achieved the highest specificity of 0.9856. For the AdaBoost based ensembles with feature selection, the NB and the MLP ensembles achieved the highest sensitivity of 0.76 and the NB ensemble produced the highest specificity of 0.9905. The AdaBoost based ensembles with feature selection had improved sensitivity and specificity as compared to the AdaBoost based ensemble with all the features. Both these ensembles had better performance than their corresponding individual classifiers except for NB that outperformed its ensemble when all the features were used.

Prusti and Jena [41] proposed an ensemble based predictive model to classify normal and attack classes. They used support vector machine (SVM), decision tree (DT) and neural network (NN) as weak learners and combined their decisions by majority voting to get the final predicted class. In the training phase, the data was pre-processed and normalised and then used to train the learning algorithms (SVM, DT and NN). The AdaBoosting concept was used to reduce the training error during the training process. The model was evaluated on two datasets, namely, NSL KDD and KDD corrected. Different performance metrics including accuracy, true positive

rate (TPR) and false positive rate (FPR) were used. The results indicated that the proposed system performed very well in both the NSL KDD and the KDD corrected datasets.

Zainal et al. [42] presented an ensemble of classifiers based IDS with the intention of maximizing detection accuracy and minimizing false alarm rates. The ensemble was made up of linear genetic programming (LGP), adaptive neural fuzzy inference system (ANFIS) and random forest (RF) as base learners. The KDD Cup 99 dataset was used to train and test the individual classifiers and the ensemble. Five classes (normal, probe, DoS, U2R and R2L as they exist in the dataset) were considered in their classification. They performed feature selection using rough-discrete particle swarm optimisation (RDPSO) that selected 15 features for all the classes out of the 41 features contained in the KDD Cup 99 dataset. The features differed from class to class. They further refined the obtained features using binary particle swarm optimisation (BPSO) and obtained 6 to 8 features for the different classes.

The performances of the individual classifiers and the ensemble were evaluated in terms of accuracy, true positive rate and false positive rates. The decisions of the individual classifiers were combined using weighted voting to make the final decision. The accuracy of the ensemble of 99.27 for normal class, 99.88% for probe, 98.26% for DoS, 99.96% for U2R and 99.79% for R2L slightly improved from that of LGP which was the best individual classifier with 98.83% for normal class, 99.68% for Probe, 97.45% for DoS, 99.91% for U2R and 99.63% for R2L.

The works of [39]–[42] focused on building an ensemble of classifiers based NIDS with different base learners with the intentions of producing a NIDS that has improved performance measured by one or more of these performance metrics accuracy, detection rate/sensitivity, false positive and specificity. These works indeed improved these performance metrics, however, the bounds on the performance of an ensemble of classifiers based are unknown. These researchers do not know how far their systems are from what is achievable by such systems, they just know that their systems are performing better than the individual classifiers and cannot estimate the performance of their systems before they implement them.

Natesan et al. [43] presented an AdaBoost algorithm for a NIDS that uses a single weak classifier for the detection of DoS, probe, U2R and R2L attacks. The main focus of their work was to improve the detection rate and reduce the false alarm rate to a minimum level. They used AdaBoost to improve the performance of each of the following weak classifiers: Bayes net, naïve Bayes and decision tree. The proposed NIDS consisted of four phases, namely, feature extraction, instance labeling, selection of base learners and building of a strong learner. They compared the performance of the three AdaBoost based NIDSs based on the detection rate of the four attacks, false alarm rate, training time and testing time. The results show that the AdaBoosted naïve Bayes had the smallest false alarm rate. The AdaBoosted

decision tree was the fastest in both training and testing. They also compared the performances of their AdaBoosted naïve Bayes and decision tree classifiers with published intrusion detection algorithms, namely, KDD 99 winner, multi-classifier and associate rule. The AdaBoosted decision tree classifiers outperformed the existing algorithms in terms of detection rate.

Both works of [5] and [43] focused on improving the detection and false positive rates of their proposed NIDSs and resulted in algorithms that yield lowest false positive rate [5] and highest overall detection rate [43] as compared to other existing intrusion detection algorithms. However, the lowest achievable bound on the false positive rate and highest achievable bound on the detection rate of a NIDS are unknown. This means

- they rely on comparing the performances of their systems to those of other systems to gauge their systems' performances.
- even though their algorithms outperform the other algorithms, they cannot determine how far their NIDSs are from these bounds in order to decide if their NIDSs need improvement or not.

Kumar and Kumar [44] presented a review on the use of artificial intelligence (AI) based ensembles for intrusion detection. Their work helps with a better understanding of the different directions in which research on ensembles has been done in the field of intrusion detection. They have observed that successful application of ensemble methods in intrusion detection depends on factors like size of the training dataset, modification of dataset for training different base classifiers, choice of accurate and diverse base classifiers, etc.

Most of the studies they have evaluated implemented their ensemble based IDSs using the KDD 99 dataset which fails to realistically simulate a real network [44]. They pointed out that there is a need for high quality benchmark datasets for intrusion detection since an IDS trained and tested on the KDD 99 dataset may not perform well in the real environment. An important feature of an IDS is its ability to adapt to the changes of the behaviour of intrusive and normal traffic without having to be retrained. If an IDS is not able to adjust according to the changes of the traffic behaviour then its detection performance may decline. According to Kumar and Kumar [44], the AI based ensembles are able to address this issue but little work has been done to address it. This means the current IDS may be underperforming due to a lack of this desired feature and need to be improved accordingly in order to perform better in the presence of such traffic changes.

Aburimman and Reaz [45] presented a survey of intrusion detection systems based on ensemble (both homogeneous and heterogeneous) and hybrid techniques along with heterogeneous ensemble applied to other domain. The survey also presented an overview of the popular ensemble algorithms namely, bagging, boosting,

stacking and a mixture of experts as well as an overview of the variants of voting methods for combining class labels. Aburimman and Reaz [45] also conducted a performance comparison of different methods used to classify the full NSL KDD dataset. Accuracy was used to compare these methods and the results suggest that an ensemble that uses majority voting to combine the decisions of the base classifiers is a better classifier. Aburimman and Reaz [45] discovered the following from the survey:

- homogeneous ensembles have been successfully implemented in IDS
- heterogeneous ensembles implementation in IDS is less complete
- weighted majority voting is rarely implemented in heterogeneous ensembles

The literature for ensemble and hybrid techniques examined in [45] survey indicate that the researchers either report the performance of their newly developed IDSs in terms of what they can achieve or compare the performance of their newly developed IDSs to the performance of existing IDS to gauge their performances. If upper bounds on the performances of homogenous and heterogeneous ensemble based IDSs were known, researches would compare the performances of their newly developed IDSs to these bounds and determine how far they are from the achievable bounds.

Jabbar et al. [46] presented a cluster based ensemble of classifiers IDS that uses the alternative decision tree and k-nearest neighbour as base learners. They implemented their IDS on Gure dataset. In their [46] approach the data was first clustered into two clusters using k-means clustering. This was followed by training their ensemble. The obtained ensemble performance was evaluated using detection rate, false alarm rate, Huberts index, Rand index and accuracy. The performance of their ensemble was also compared to that of existing IDSs of [47] and [48] in terms of detection rate and false alarm rate. The results indicate that the proposed IDS performed well and outperformed the existing IDSs. It is unclear why the authors would use a clustering technique k- means on a labeled dataset.

The recent work of Mkuzangwe and Nelwamondo [49] presented the performance bound of an AdaBoost based NIDS that uses the decision stump as a base learner. The performance bound of their NIDS is defined in terms of the average information gain amongst the features that were used in building the ensemble. The NSL KDD dataset was utilised in this study.

Moustafa et al. [50] presented an AdaBoost based ensemble of classifiers IDS for detecting attack on network traffic of the Internet of Things (IoT). The decision tree, naïve Bayes and artificial neural networks were used as weak learners. Statistical features associated with DNS, HTTP, MQTT protocols were generated since these protocols are mainly used in the IoT services. The features associated with the HTTP and DNS protocols were generated from the UNSW-NB15 and NIMS datasets.

The coefficient of correlation was used to select features to be included in the ensemble. For each dataset (UNSW-NB15 or NMS), two ensembles were built using the HTTP features for one ensemble and the DNS features for the other ensemble. The performance of each of the developed ensembles was evaluated using accuracy, detection rate (DR), false positive rate (FPR), time and receiver operating characteristic (ROC) curve. For both datasets, the results indicate the built ensembles outperformed the weak learners in terms of accuracy, detection rate, false positive rate and ROC curve. The ensembles were also evaluated on detecting specific attacks and the results show that the ensembles performed well. The performances of the ensembles were compared to the existing support vector machines of [51], Bayesian networks of [51] and Markov chain for botnets of [52]. The ensembles outperformed the existing works in terms of DR and FPR. According to the authors, this work can be extended by collecting more relevant features from other IoT protocols in order to build an effective profile of normal and attack patterns.

Mirsky et al. [53] presented Kitsune, an unsupervised online ensemble of autoencoders based NIDS to perform an anomaly detection of attacks on a local network. Kitsune's framework consists of packet capturer, packet parser, feature mapper and anomaly detector. The main input parameter for Kitsune is the maximum number of inputs to the autoencoders (m). The parameter determines the complexity of the ensemble and involves the trade-off between detection and runtime performance. Real IP camera vieo surveillance network and wi-Fi network consisting of nine IoT devices and three personal computers were used to evaluate the performance of Kitsune in terms of detection and runtime. The performance of Kitsune at $m = 1$ and $m = 10$ was compared to offline algorithms such as isolation forest of [54] and Gaussian mixture models (GMM) of [55] and online algorithms such as incremental GMM of [56], pcStream2 of [57] and Suricata of [58]. The true positive rate (TPR) and false negative rate (FNR) of these systems were compared at FPR of 0 and 0.001. The results indicate that the performance of Kitsune is indeed comparable to the existing algorithms. However, the number of hidden layers of Kitsune's autoencoders was restricted to three and how this value was obtained is not justified in the work. This means that one cannot tell if the obtained results are optimal or not.

2.2 Network Trace Privatisation

Network trace privatisation techniques can be typically categorised into two paradigms, namely, sanitisation based systems and query based systems [59]. The sanitisation based systems remove or anonymise privacy sensitive packet fields such as payloads and IP addresses. The anonymisation process intends to protect the privacy of the monitored users, hide the topology of the network and

provide realistic and useful anonymised network trace [60]. Literature reveals that anonymisation is vulnerable to attacks that infer sensitive information. This means trace anonymisation provides no guarantee in protecting sensitive information.

The query based systems are commonly used in the problem of protecting sensitive information in a database while allowing statistical queries. Adam and Wortmann [61] classify the approaches taken by the query based systems into four categories: query restriction, data perturbation, output perturbation and conceptual approach. Where [61] defines these categories as follows:

- In query restriction, the queries are required to follow a particular structure in order to prevent attackers from gaining too much information about records in a specific database.
- In data perturbation, the database is perturbed and the query is answered according to the perturbed database.
- In the output perturbation, the database computes the answer of the query first and returns the perturbed version of the answer.
- In the conceptual approach two models are defined namely, conceptual and lattice models. The conceptual model provides a structure for investigating the security problem at the conceptual data level. The lattice model provides a structure for representing the data in a tabular form.

The use of query restriction and data perturbation approaches in protecting network trace have been proposed in the literature. Subsection 2.2.1 presents a review of studies done on network trace privatisation.

2.2.1 Literature on Network Trace Privatisation Techniques

Many challenges are associated with network trace privatisation such as information loss after privatising the data, utility loss of the privatised data, attacks that want to infer sensitive information from the privatised data, etc. Therefore, the proposed systems must be mindful of these challenges in their design requirements [59] so that they produce research useful data while preserving privacy. A review of works that conducted network traffic trace privatisation is presented below.

Mogul and Arlitt [21] proposed an alternative approach to trace anonymisation where data owners run analyses on behalf of the researchers, i.e. researchers send their code to data owners, to preserve privacy. The proposed approach does not send traces from the trace owners to the researchers, only the results and this solves the problem of shipping large traces. However, the results may still reveal some information about the behaviour of the information contributors in the trace that might single out individuals or machines [21]. To handle this the authors have

proposed that an independent expert or the trace owner must review the properties of the framework and analysis module to detect leakage. This means that the privacy preserving approach depends on human verification which is error prone [24]. The approach also addresses the issue of trace storage, since some organisations prohibit internal trace storage, by performing online data analysis as the data arrives and discarding it after the analysis. The authors have pointed out several drawbacks in their approach that include the following:

- Debugging the analysis software will be difficult since the code would have been trained on different software.
- The reasons for trace owners to participate are unclear.
- It might be hard to use this approach if the analyses need to be done across multiple sites since there may be in conflict with data owner privacy policies.

McSherry and Mahajan [24] investigated the potential for network trace analysis while providing the guarantees of differential privacy (DP). The network trace analyses performed by [24] include multiple examples of packet level, flow level and graph level computation from network literature. They used Privacy Integrated Queries (PINQ) [62] as an analysis platform that provides differential privacy. They performed two kinds of packet level analyses. One computed the cumulative distribution functions (CDFs) of packet size and ports and the other was a computation that involved payloads, ports and IP addresses for automated worm fingerprinting. For the CDFs of the packet size and ports they observed the behaviour of the CDFs at three privacy budgets, $\varepsilon = 0.1, 1$ and 10 and compared these CDFs to the CDF of the noise free data. They also computed the root mean square error to measure the overall accuracy. The results indicate that the privately computed CDFs for both packet size and ports faithfully followed that of a noise free data. The RMSE for packet size and ports were 0.01% and 0.07% respectively at $\varepsilon = 0.1$. In automated worm fingerprinting the results indicate that with dispersion threshold of 50 for sources and destinations, the noise free computation yielded 29 payloads. The differential private search with $\varepsilon = 0.1, 1$ and 10 reveals 7, 24 and 29 payloads respectively. This means the accuracy of worm fingerprinting is low at high privacy levels and high at low privacy levels.

In the flow level analysis, McSherry and Mahajan [24] considered two kinds of analyses where one computes statistics within the flow and the other analysis operates across packets of different flows. For statistics within the flow they computed the flow round trip time (RTT) and loss rate without noise and with differential privacy at $\varepsilon = 0.1, 1$ and 10 and then studied their distributions in terms of CDFs. The differentially private CDFs faithfully followed that of a noise free RTT and loss rate. The RMSE for RTT and loss rate were 2.8% and 0.2% respectively at $\varepsilon = 0.1$. For the analysis across the flow, they detected stepping stone relationships between flows with differential privacy at $\varepsilon = 0.1, 1$ and 10. The results indicated

that $\varepsilon = 0.1$ led to the highest false positive rate and $\varepsilon = 1$ with the lowest false positive rate.

In the graph level analysis [24] considered two analyses, namely, anomaly detection and passive network discovery. In anomaly detection they observed the link level traffic volumes across time. They followed the analysis proposed by Lakhina et al. [63] to detect anomalies. The results show that the behaviour of anomalous traffic computed with differential privacy is indistinguishable from the one computed without noise. In the passive network discovery they wanted to reproduce the clustering analysis conducted by Eriksson et al. [64] in a differentially private manner. The results indicate that the analysis is only accurate for low privacy levels. McSherry and Mahajan's [24] results indicate that differential privacy has the potential to be the basis for mediated network trace analysis. However, the work [24] was implemented at a granularity privacy principal of records in the dataset, developing support for coarser granularity privacy principals like hosts or flows (even if the underlying data is at a finer granularity like packets) is still a challenge as pointed out by [24]. They [24] also pointed out that guidelines for data owners on what privacy level to set for their datasets need to be developed.

Pang and Paxson [65] developed a new method to allow anonymisation of packet payloads as well as headers. The tool provides high level language support for packet transformation allowing the user to write short policy scripts to express sophisticated trace transformation. Traces are processed in three steps, namely,

- Reassembling and parsing of payloads to generate application protocol level protocol and semantically meaningful data elements.
- A policy script transforms data elements to remove sensitive information and send the resulting elements to the composer.
- The trace composer converts application protocol data elements back to bytes sequences and frames the bytes into packets, matching the new packets to the original packets in order to preserve the transport protocol dynamics.

They developed an anonymisation scheme for FTP trace and implemented it on the LBNL's FTP traces. Their objectives were as follows:

- To ensure that anonymisation hides the identification of clients, non-public FTP servers, non-public files and confidential authentication
- To ensure that anonymisation keeps the original request/reply sequence and other non-sensitive information intact.

Their scheme [65] does not address all of the attacks that infer sensitive information and is dependent on the specific policy approved by a site [65] which means a scheme that is site policy independent and prevents sensitive information from being inferred by attackers is needed.

Xu and Moon [66] focused on IP address anonymisation. They evaluated the prefix preserving anonymisation techniques with the intention of establishing the effect of some types of attacks on the security of the prefix preserving anonymisation process and to determined a bound on the effectiveness of the attacks in general. They also developed a cryptography based prefix preserving anonymisation technique. They developed the cryptography based technique in order to address the drawbacks of the TCP dpriv, an existing prefix anonymisation tool, while maintaining the same level of anonymity as TCPdpriv.

Instead of preserving network trace privacy by using anonymisation or removal of sensitive data, Mirkovic [67] proposed a new paradigm of secure queries. In this paradigm the data owner publishes a query language and online portal allowing researchers to submit sets of queries to be run on the data. In the work of [66], the privacy of results released by the data owners was verified by an independent expert. To remove human verification Mirkovic [67] proposes query restriction based on the provider's privacy policy and was enforced by the secure query language. The query restriction forbids some operations to be done on certain trace fields and in some context in order to avoid private leakage in the presence of passive and active attacks and maximise trace's research utility. The authors pointed out that they could not prove that there was no information leakage by secure queries, however, they showed that some attacks could be handled by their proposed privacy methods.

Mugal and Arlitt [21] presented a method that uses human verification to preserve privacy and Mirkovic [67] removed human verification and introduced rules to preserve privacy, however, the privacy properties attained by these works are unclear [24].

Dijkhuizen and Ham [68] conducted a literature survey over the period of 1998-2017 on network traffic anonymisation techniques and their implementations. In the survey,

- a brief description of currently available anonymisation and pseudonymisation techniques and a rough indication of their effectiveness is provided. These techniques include prefix preservation [69], replacement of a field [65] and data removal [70]
- fields containing privacy sensitive information in the link, internet and transport layers are discussed,
- existing anonymisation tools and frameworks like Bro Anonymiser Plug-in [65], PktAnon [71] and Anonym [72] are described. The tools are compared against each other based on the following features, whether the source compilation is without problems, ability to anonymise IPv4 and IPv6 addresses, whether the anonymisation supports MAC addresses, VLAN tags,

UDP/TCP port numbers, header checksum correction, application layer, tunnels and IP/TCP options, real time or online anonymisation and license type for the tool (open source or proprietary),

- future research directions to enable easier sharing of network traffic are provided.

From the comparison of the tool, PktAnon met most of the evaluation features except for anonymisation support for the application layer and partial anonymisation support for IP/TCP options. Some of the future research directions suggested by Dijkhuizen and Ham [68] include:

- the need for updating the rigorous mathematical review of the anonymisation techniques since the last one was conducted by Coull et al. [73] in 2008 to show new developments.
- IPv6 packet sensitive fields have not been looked into much. Furthermore, sensitive fields may change due to the changing nature of network traffic and the context of the captured network.
- their survey indicated that packet trace anonymisation has not matured, therefore there is a need for undertaking more anonymisation of packet traces since packet traces are needed for conducting different network research.

Fan et al. [74] adopted differential privacy to protect the presence or absence of each individual web browsing session. Fan et al. [74] considered the problem of releasing the number of visits to each web page at every time stamp instead of releasing a collection of browsing sessions. A state space approach to monitor the number of visits to each webpage at every time stamp using differential privacy with the true aggregates being the hidden states and the noisy aggregates (noise perturbed true aggregates by a differential privacy mechanism) as noisy observations was proposed by Fan et al. [74]. Fan et al. [74] used the Laplace mechanism to perturb true aggregates. Based on the differential privacy perturbation mechanism, Fan et al. [74] proposed to release the posterior estimates of the hidden states than the noisy observations for web monitoring. Fan et al. [74] established a univariate time series space model to monitor the number of visits to a webpage per time stamp and a multivariate time series space model for simultaneously monitoring the visits to all webpages. For both models, they used the Kalman filter to obtain the posterior estimates of the true aggregates. To learn the parameters of the model they used a subset of the training data. Fan et al. [74] used the following utility measures: average relative error, precision for top-K mining and KL-divergence to test if the usefulness of the released posterior estimates is close to that of the true aggregates. They conducted their experiments using real data from msnbc.com. The results indicate that their proposed methods release in real time useful posterior estimates that preserve privacy. The work of Fan et al. [74] is limited to monitoring browsing

sessions to a single server, however, in real world users browse in different servers, across platforms, etc. Fan et al. [74] pointed out this shortfall as something that can be explored in the future.

Blocki et al. [75] presented a novel mechanism for releasing the perturbed password frequency list. They based their differential privacy mechanism on the exponential mechanism of McSherry and Talwar [76]. They represented all possible password frequency lists of a passwords database consisting of N users as all possible partitions of integer N with a password frequency list being a single partition of the integer N. In exponential mechanism, given a password frequency list as an input, each possible randomly sampled integer partition from the exponential mechanism is returned with a probability associated with it. The exponential mechanism preserves ε- differential privacy and leads to minimum cumulative distortion (cumulative distortion measures the distance between the original password list and the released password list). However, according to [75] direct implementation of the exponential mechanism will require exponential time and space and there is evidence that sampling with the exponential mechanism is computationally demanding [77]. To address this drawback of an exponential mechanism, they proposed a novel dynamic programming algorithm to approximate samples from an exponential mechanism for integer partitions when N is large. As a first step in developing this novel sampling algorithm they proposed the relaxation of the exponential mechanism by completely ignoring partitions from the exponential mechanism that are far from the true distribution (large values of cumulative distortion). This relaxed exponential mechanism provides (ε, δ)-differential privacy and minimal cumulative distortion. They developed a novel algorithm (of $(N^{1.5}/\varepsilon)$ time according to [75]) to sample from this relaxed exponential mechanism. They conducted their experimental work using the data from RockYou passwords breach with ε varying between 0.002 and 8 and $\delta = 2^{-100}$. The results indicate that the normalised distortion coefficient (which is the distance between the released password frequency list and the original password frequency list divided by the number of users) lies between $8.83E-07$ and $1.90E-03$. Based on the obtained values [75] concluded that their sanitized password frequency list would be useful for password research without testing the utility of these released password frequency list. They also had a challenge of fitting the dynamic programming tables into memory as ε got smaller. To overcome the memory challenge they only stored pieces of the dynamic programme table in memory and recomputed the rest as they progressed. They pointed out the memory challenge as a place of improvement in their work.

Deng and Mirkovic [78] proposed a mechanism that achieves commoner privacy-interactive k-anonymity. Commoner privacy fuzzes only those output points where an individual's contribution is an outlier by omitting or aggregating or adding noise. They also discussed query composition and showed how they can guarantee

privacy via a pre-sampling step or query introspection. They implemented their privacy mechanism and query introspection on network traces, namely, packet counts sent per source port, packet counts received per destination service port, connection count in the trace and traffic volume in the trace using a system called Patrol. They compared the performance of their privacy preserving mechanism against differential privacy and crowd blending privacy. The results indicate that their proposed mechanism released outputs that have a higher research utility as compared to the two privacy preserving techniques. However, differential privacy guarantees high privacy than the other two techniques [78] and can protect against both all-but-one and interactive adversaries. The other two techniques can protect an individual from interactive adversary only. Several approaches to improve the utility of release aggregates using differential privacy exists. Therefore, releasing aggregates using differential privacy is still of benefit.

2.3 Summary

Table 2.1 provides a summary of the papers that have been presented in intrusion detection using sensor combining techniques and network trace privatisation techniques since 2002. The author and the year of publication are presented in the first column. The sensor combining technique or network trace privatisation technique used is provided in the second and the third column gives the aim of each paper.

TABLE 2.1: Summary of some of the literature in intrusion detection using sensor combining techniques and network trace privatisation techniques since 2002.

Author	Sensor Combining Technique or Network Trace Privatisation Technique	Main Objective
Thomas et al. [11] 2008	Weighted Aggregation	presented an architecture using data dependent decision fusion.
Tian et al. [12] 2005	DS	Merged alerts from different IDSs to estimate the current network security status
Thomas et al. [13] 2007	Rule based fusion	Set threshold bounds for sensor fusion using the principle of Chebyshev inequality and used rule based fusion to improve sensor fusion performance.

Gong et al. [20] 2010	NN	presented a neural network based data fusion model to perform fusion and intrusion detection
Chan et al. [30] 2005	DS, MV, WMV, NB, A and NN.	Compared the different fusion strategies for a multiple classifier system for a denial of service problem.
Kaliappan et al. [31] 2015	MV	fused heterogeneous IDSs to detect network attacks.
Shah et al. [32] 2017	Modified DS rule	discussed the limitations of DS rule in combining IDSs decisions and proposed a new fusion rule which modifies the DS framework.
Mkuzangwe et al. [35] 2015	OR operator	fused the outcomes of two anomaly based IDS in order to improve the detection probability of the two IDSs.
Li et al. [37] 2015	DS, RF, NN and DD	presented a review of data fusion techniques used in building network intrusion detection systems.
Hu and Hu [5] 2005	Weighted Majority voting	proposed an AdaBoost based algorithm that uses decision stump as a base learn for network intrusion detection systems.
Borji [38] 2007	Majority voting,Bayesian averaging and a belief measure	presented a combining approach for intrusion detection where different combining strategies we used to illustrate their effect.
Govindarajan and Chandrasekaran [39] 2012	Arcing	presented a new arcing based hybrid classifier that is made up of heterogeneous classifiers.
Sornsuwit and Jaiyen [40] 2015	Weighted Majority Voting	built an AdaBoost based ensemble for detecting network intrusion data that are difficult to detect (U2R and R2L).
Prusti and Jena [41] 2015	Majority voting	presented an AdaBoost based predictive model to classify normal class and attack class.
Zainal et al. [42] 2009	Weighted voting	presented an ensemble of classifiers based IDS in order to maximise detection accuracy and minimise false alarm rate.

Natesan et al. [43] 2012	Weighted Majority Voting	to improve the performance of each of the weak classifiers (Bayes Net, Naïve Bayes and decision tree), used the AdaBoost algorithm.
Kumar and Kumar [44] 2012	Majority Voting, Threshold plurality vote, Fuzzy Theory, etc	presented a review on the use of artificial intelligence (AI) based ensembles for intrusion detection.
Aburimman and Reaz [45] 2017	Voting	presented a survey of intrusion detection systems based on ensemble (both homogeneous and heterogeneous) and hybrid techniques and heterogeneous ensemble applied to other domain.
Jabbar et al.[46] 2017	Voting	presented a cluster based ensemble of classifiers that used the alternative decision tree and k-Nearest Neighbour as base learners.
Moustafa et al. [50] 2018	weighted Majority Voting	presented an AdaBoost based ensemble of classifiers IDS for detecting attacks on network traffic of IoT.
Mirsky et al. [53] 2018	Voting	presented Kitsune, an unsupervised online ensemble of autoencoders based NIDS to perform anomaly detection of attacks on a local network.
Mogul and Arlitt [21] 2006	Human Verification (Trace owners/ Independent Expert)	proposed an alternative approach to trace anonymisation where researchers send their code to data owners and the data owners do the analysis for the researchers, to preserve privacy. After the analysis, trace owners send the results to the researchers. To prevent any information leakages an independent experts checks the data
McSherry and Mahajan [24] 2011	Differential Privacy (DP)	investigated the potential for network trace analysis while providing the guarantees of differential privacy.
Pang and Paxson [65] 2003	Policy Script	in order to anonymise packet headers and payloads, proposed a tool that provides high level language support for packet transformation allowing the user to write short policy scripts to express sophisticated trace transformation.

Xu and Moon [66] 2002	cryptography based prefix preserving anonymisation	developed a cryptography based prefix preserving anonymisation technique in order to address the drawbacks of the TCP dpriv, an existing prefix anonymisation tool.
Mirkovic [67] 2008	Secure Queries	proposed a new paradigm of secure queries where the data owner publishes a query language and online portal allowing researchers to submit sets of queries to be run on the data.
Dijkhuizen and Ham [68] 2018	Bro Anonymiser Plug-in, PktAnon, Anonym, etc.	presented a literature survey over the period of 1998-2017 on network traffic anonymisation techniques and their implementation.
Fan et al. [74] 2014	DP	protected the presence or absence of each individual web browsing session using differential privacy.
Blocki et al. [75] 2016	DP	protected a password frequency list using differential privacy.
Deng and Mirkovic [78] 2017	Commoner Privacy-Interactive k-Anonymity	proposed a mechanism that achieves commoner privacy-interactive k-anonymity.
Mkuzangwe and Nelwamondo [79] 2019	Differential Privacy	proposed differentially private number of TCP SYN packets, which is a different network trace than in literature.

The above review of works in the field of intrusion detection based on sensor fusion and ensemble methods reveals that researchers

- focus on improving the performance of their NIDSs even though they do not know to what extent the performance can be improved.
- implement their systems and either report the performance obtained by their NIDSs or compare the performance of their new IDSs to existing NIDSs in order to gauge how good their new NIDSs are.
- cannot estimate the performance of their NIDSs before they implement them.

This review indicates that if the achievable performance bounds of the NIDSs that are based on sensor fusion and ensemble methods were known, researchers would

compare their new NIDS to these bounds and be able to determine if their NIDSs need improvement or not. Furthermore, researchers would be able to estimate what their NIDSs are capable of achieving before they implement them. Therefore, this Thesis aims at determining these achievable performance bounds of sensor fusion and ensemble methods based NIDSs so that researchers can have a reference point to gauge the performance of their NIDSs.

The review of studies in network trace privatisation indicates that more works in privatising packet trace are needed. Therefore, this Thesis also aims to address the need for more privatisation of packet traces by implementing differential privacy on, TCP SYN packet counts, a different network trace than in literature.

Chapter 3

Methodology and Data Description

This Chapter provides a description of the experimental approach, techniques and data that are used to evaluate the first Thesis statements in Section 1.3. The first Thesis statement states that the achievable upper bounds on the performance of network intrusion detection systems that make use of multiple sensors combining techniques can be derived via an information theoretic approach with given sensor specification. Two combined IDSs methods are considered in this work, namely, sensor fusion and ensemble methods. This Thesis also evaluates the performance of existing individual IDSs in order to illustrate what the existing IDSs can currently achieve. Therefore, techniques that are used to build individual IDSs are also presented in this Chapter. The individual and combined IDSs and their corresponding techniques/algorithms that are used in this work are presented in Table 3.1.

TABLE 3.1: The types of IDSs and their corresponding algorithms used

IDS Types	Algorithms Used
Individual IDSs	Adaptive Threshold Algorithm, Cumulative Sum (CUSUM) based Algorithm, Fuzzy Logic and Decision Tree.
Sensor Fusion based IDSs	Bayesian Inference, Dempster-Shafer Belief Theory, Voting Fusion Theory, Neural Network and Logic OR operator.
Ensemble based IDSs	Boosting and Bootstrap Aggregating

The rest of this Chapter is as follows, Section 3.1 presents the experimental approach that will be used to test the first Thesis statement, which will be followed by a description of intrusion detection techniques used in individual and combining multiple intrusion detection systems in Sections 3.2 and 3.3 respectively. Section 3.4 presents the selection of techniques to be used in this Thesis. The Chapter is concluded by providing a detailed description of the data that will be used in this Thesis in Section 3.5 and a brief summary.

3.1 Experimental Approach

The knowledge of the performance bounds of IDSs that use multiple sensor combining techniques for event aggregation would help researchers with designing the system without relying only on simulation of the system. Moreover, these bounds would allow researchers to determine how far the current IDS that uses the multiple sensors combining techniques for event aggregation performance falls short of what is achievable. Therefore the first ultimate objective of this Thesis is to provide the upper bound on the performance of the ensemble of classifiers based network intrusion detection system (NIDS) under the following specifications:

a) classifiers in the ensemble are dependent and

b) classifiers in the ensemble are independent.

Classifiers are independent if the joint probability density/mass function of their outcomes on any class i is equal to the product of the marginal density/mass functions of their outcomes on any class i, otherwise, they are dependent. The general experimental process used in determining the performance upper bound of an ensemble of classifiers based NIDS is depicted in Figure 3.1. In the data pre-processing step, the classes that are of interest are decided upon and the dataset is extracted according to these classes. Features that are going to be used in classifying the network traffic are selected. All features or a subset of features selected based on a feature selection method that helps to decide on the importance of a feature or criteria can be used in fitting/training the classification model. In the classification model training step, a classification model is trained on the features obtained in the data pre-processing step. This step is followed by testing the performance of the model on a new dataset that the model has not seen. If the model performs well then the model training stops otherwise optimisation techniques are used to improve the performance of the model. On the model performance optimisation phase, the optimisation techniques that are used to boost the performance of the classification model to its optimality are applied to the model. One of those optimisation methods is adaptive boosting (AdaBoost) algorithm. A detailed description and pseudocode of AdaBoost algorithm is given in Section 3.3 under the ensemble of classifiers methods. The model performance evaluation step presents the results of the optimised model and the upper bound on its performance. The results of the model are commonly presented on a confusion matrix form that includes false positives, false negatives, true positives and true negatives.

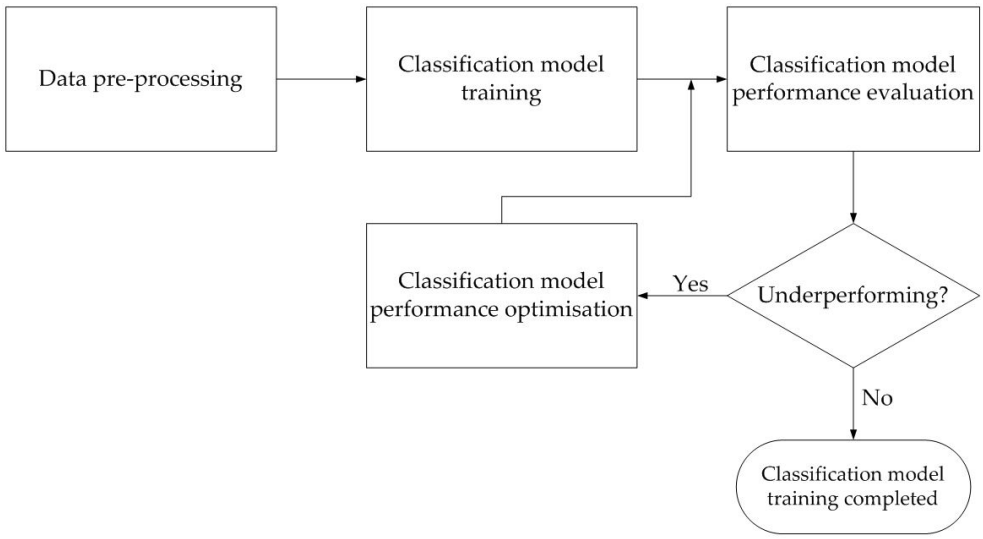


FIGURE 3.1: General experimental process used to determine the performance upper bound of an ensemble of classifiers based NIDS.

True Positives (TP) are the number of positive instances that were correctly classified as positives.

True Negatives (TN) are negative instances that were correctly predicted as negatives.

False Negatives (FN) are positive instances that were misclassified as negative.

False Positives (FP) are negative instances that are misclassified as positive.

Accuracy is the proportion of correctly classified instances in the dataset.

True Positive Rate (TPR) or Sensitivity is the positive class accuracy or is the probability of correctly classifying the attacks.

True Negative Rate (TNR) is the probability of correctly classifying the non-attacks.

False Positive Rate (FPR) is the probability of incorrectly classifying normal data as attacked data.

False Negative Rate (FNR) or Specificity is the probability of incorrectly classifying attacked data as normal data.

Detection delay (DD) is the average time at which the algorithm was able to detect the occurrence of the attacks.

Equal Error Rate (EER) is an error rate value where the FPR is equal to the FNR.

The equations for Accuracy, TPR, TNR, FPR and FNR are given below.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}. \tag{3.1}$$

$$TPR = \frac{TP}{TP + FN}. \tag{3.2}$$

$$TNR = \frac{TN}{FP + TN}. \tag{3.3}$$

$$FPR = \frac{FP}{FP + TN} \text{ and.} \tag{3.4}$$

$$FNR = \frac{FN}{TP + FN}. \tag{3.5}$$

3.2 Intrusion Detection Techniques used in Individual IDSs

A standard NIDS monitors network traffic on a target network while attempting to detect malicious activities [17]. In general, NIDSs attempt to identify attacks carried out over a network that targets some end node and attacks that target the network itself. The NIDS's purpose is to reliably detect (in terms of the detection probability) and/or classify attacks in as short a time span as possible (i.e. the detection delay) while maintaining a low false positive rate. In this section, four intrusion detection techniques or algorithms that are used by individual IDSs to classify events as attacks or normal network behaviour are described. These are the adaptive threshold algorithm, cumulative sum (CUSUM) based algorithm, fuzzy logic and decision tree. Literature reveals that the first three techniques are easy to implement and the CUSUM algorithm is one of the mostly used algorithms in anomaly based intrusion.

3.2.1 Adaptive Threshold Algorithm

This algorithm tests whether the traffic measurement, number of Transmission Control Protocol (TCP) Synchronise (SYN) packets in a given time interval, exceeds a certain threshold. The trends and seasonality (weekly and daily variations) are addressed by adaptively setting the threshold value from an estimate of the mean of the traffic measurements. If x_n is the traffic measurement in the n-th time interval and μ_{n-1} is the mean rate estimated from measurements prior n, then at time n the alarm is signalled if

$$x_n \geq (\alpha + 1)\mu_{n-1}. \tag{3.6}$$

where $\alpha > 0$ is a parameter that the percentage above the mean value that is considered to be anomalous behaviour. Some past time window or exponential

weighted moving average (EWMA) of past measurements can be used to calculate the mean μ_n as:

$$\mu_n = \beta\mu_{n-1} + (1-\beta)x_n. \tag{3.7}$$

where β is the EWMA factor. Normal traffic, now and again, can violate the threshold, therefore, relying on a single threshold violation as a detection condition may increase the number of false positives. To enhance the performance of the algorithm, the algorithm's alarm condition is changed such that the alarm is signalled after a minimum number of successive threshold crossings i.e. If

$$\sum_{i=n-k+1}^{n} 1_{\{x_i \geq (\alpha+1)\mu_{i-1}\}} \geq k. \tag{3.8}$$

then an alarm is raised at time n , where $k > 1$ is a parameter that represents the number of successive intervals the threshold must be crossed for an alarm to be signalled. The tuning parameters of the above algorithm are α, k, β and the size of the time interval over which traffic measurements are taken.

3.2.2 Cumulative Sum (CUSUM) based Algorithm

The CUSUM algorithm comes from the family of change point detection algorithms that are hypotheses testing based and was developed for independent and identically distributed random variables $\{y_i\}$. The CUSUM algorithm consists of two hypotheses θ_0 and θ_1, where the first hypothesis corresponds to the statistical distribution before a change and the second hypothesis to the distribution after a change. The test for indicating a change is based on the log-likelihood ratio S_n given by

$$S_n = \sum_{i=1}^{n} s_i \text{ where } s_i = \ln \frac{p_{\theta_1}(y_i)}{p_{\theta_0}(y_i)}. \tag{3.9}$$

The log-likelihood ratio S_n tends to include a negative drift prior to a change and a positive drift after the change. Therefore, the relevant information for detecting a change is found on the difference between the log-likelihood ratio value and its current minimum value [80]. Hence, the alarm is raised at time n if $g_n \geq h$, where

$$g_n = S_n - m_n \text{ and } m_n = \min_{1 \leq j \leq n} S_j. \tag{3.10}$$

and h is a threshold parameter. The direct form of the CUSUM algorithm is presented in Algorithm 1

Using the repeated sequential probability ratio test (its definition can be found in [80], (3.10) can be recursively written as

$$g_n = [g_{n-1} + s_n]^+. \tag{3.11}$$

Recalling that $s_n = \ln \frac{p_{\theta_1}(y_n)}{p_{\theta_0}(y_n)}$, if the $\{y_i\}$ are assumed to be independent random variables that follow a Gaussian distribution with a known variance σ^2, which is assumed to stay the same after the change, and μ_0 and μ_1 the mean prior to and after the change respectively then

$$\begin{aligned} s_n &= \ln\left(\frac{\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(y_n-\mu_1)^2}{2\sigma^2}}}{\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(y_n-\mu_0)^2}{2\sigma^2}}}\right) \text{ since } p_{\theta(y_n)} = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(y_n-\mu)^2}{2\sigma^2}} \\ &= \ln\left(\frac{e^{-\frac{(y_n-\mu_1)^2}{2\sigma^2}}}{e^{-\frac{(y_n-\mu_0)^2}{2\sigma^2}}}\right) \\ &= \left(-\frac{(y_n-\mu_1)^2 m}{2\sigma^2} + \frac{(y_n-\mu_0)^2 m}{2\sigma^2}\right) \\ &= \frac{1}{2\sigma^2}\Big(2y_n(\mu_1-\mu_0) - (\mu_1+\mu_0)(\mu_1-\mu_0)\Big) \\ &= \frac{(\mu_1-\mu_0)}{\sigma^2}\left(y_n - \frac{(\mu_1+\mu_0)}{2}\right). \end{aligned} \tag{3.12}$$

Substituting (3.12) to (3.11) results in (3.13).

$$g_n = [g_{n-1} + \frac{\mu_0-\mu_1}{\sigma^2}(y_n - \frac{\mu_1+\mu}{2})]^+. \tag{3.13}$$

The algorithm assumes that y_i are independent random variables that follow a Gaussian distribution which is not true for the number of SYN packets due to seasonal variations, trends, and time correlations. Such non-stationarity behaviour needs to be removed prior to using the CUSUM algorithm. The tuning parameters of the above algorithm are α h, β and the size of the time interval over which traffic measurements are taken.

Algorithm 1 The direct form of the CUSUM algorithm

Initialisation

- set the detection threshold $h > 0$
- $k = 0$

end

While the algorithm is not stopped do

- measure the current sample $s[k]$
- $s[k] = \ln\left(\frac{p(s[k],\theta_1)}{p(s[k],\theta_0)}\right)$
- $S[k] = \sum_{n=0}^{k} s[n]$;
- $G_{(X)} = S[k] - \min_{1 \leq n_c \leq k} S[n_c - 1]$
- if $G_{(X)} > h$ then
  - $n_d < -k$
  - $n_c = \arg\min_{1 \leq n_c \leq k} S[n_c - 1]$
  - stop or reset the algorithm

  end
- $k = k + 1$

end

3.2.3 Fuzzy Logic

Zadeh [81] conceived the concept of fuzzy logic as a way of processing data by using a fuzzy set instead of using a crisp set. In a fuzzy set, partial membership to a set is allowed whereas in crisp set an instant either belongs to a set or does not belong to a set. It provides very valuable flexibility for reasoning that takes uncertainties and inaccuracies into considerations [82]. It provides a simple way of reaching a definite conclusion based upon ambiguous, vague, imprecise, noisy or missing input information. This definite conclusion is reached using the following

steps: fuzzification, fuzzy rule generation, fuzzy inferencing and finding the crisp value of the output variable. In fuzzification all input and output variable values are fuzzified into fuzzy membership functions, that is, the range of values taken by each input variable are divide into fuzzy sets and a fuzzy membership function is determined based on expert knowledge to assign the degree of membership to a fuzzy set of the input and output variables. An example of a membership function for an input variable called same host connection counts (which is the number of connections to the same host as the current connection in the past two seconds) whose input values are broken down to fuzzy sets No Attack, Medium Attack and Heavy Attack is given in Figure 3.2. In fuzzy rule generation, the rules are generated in the form of If $X = x$ and $Y = y$ then $Z = z$ statement and are used to describe the desired system output. Given an instance, some of the fuzzy rules will be activated. In fuzzy inferencing, the activated fuzzy rules are aggregated to obtain the fuzzy output distribution. Three fuzzy inference methods exist, namely, Mamdani, Sugeno and Tsukamoto with Mamdani being the most commonly used fuzzy inference method [83]. The fundamental difference between the three inference methods is the way they obtain their crisp value from the fuzzy inputs. Mamdani inference method defuzzifies the fuzzy output distribution whereas the Sugeno and Tsukamoto inference methods use a weighted average to determine the crisp value [83].

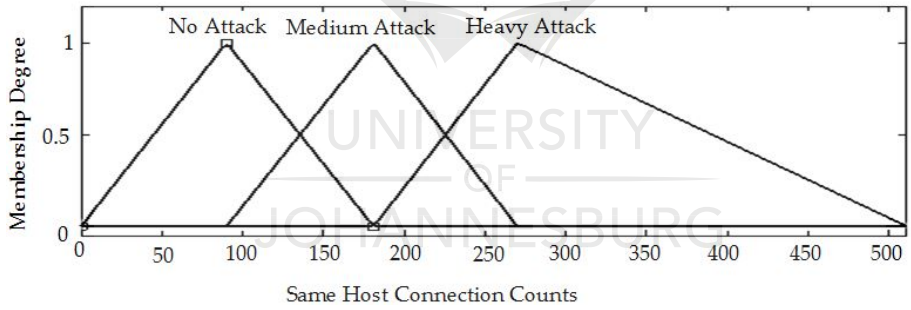


FIGURE 3.2: Membership function for variable same host connection counts.

3.2.4 Decision Tree

Decision trees (DT) are among the popular classification models due to their ease to implement and understand. A decision tree is made up of three basic elements:

1. a decision node which specifies a test attribute;
2. a branch which corresponds to one of the possible values of the test attribute; and
3. a leaf that contains the class to which the object belongs.

To use a decision tree for classification of instances, two phases should be ensured. The first phase is to build the decision tree that involves the selection of the appropriate test attributes for each decision node and defining the class label for each leaf given training data. The second phase involves the classification of instances where new instance classification begins at the root of the decision tree where the attribute specified by this node is tested. The result of this test allows moving down the tree branch that corresponds to the attribute value of the given instance. This process is repeated until a leaf is reached. The instance is then classified in the class that is described by the reached leaf (descend strategy). To ensure the descend strategy the following components are required

a) the attribute selection measure that determines the ability of the features to split the training dataset into classes and assign the root of the tree. Many attribute selection measures exist in literature with Shannon's entropy [84] and Information Gain [84] being the most common measures.

b) the partitioning strategy aims at dividing the current training dataset by taking into account the selected test attribute.

c) the stopping criterion is used to stop the tree growing process. There are two possible ways of stopping the tree growing process: stop if all the leaf nodes contain one class or stop when the number of test cases has fallen below a certain threshold.

Several algorithms have been developed to construct decision trees with the C4.5 and ID3 algorithms developed by Quinlan [85] being probably the most popular.

Decision trees are, however, known to be unstable [86] (i.e small changes in the training data may lead to drastic changes in the structure of the tree which may cause the performance of the tree to change). To address this shortfall of the DT, ensemble methods that combine many trees into a single model that has better predictive performance than a single tree have been developed by researchers. Bootstrap aggregating (Bagging) is one example of such ensemble methods.

A decision stump is a one level decision tree. A decision stump uses only one feature for splitting at the root level and is often used in the ensemble method called boosting as a weak learner. A weak learner is a learner that has a classification accuracy on new instances of just above random guessing.

Due to the complexity of the network traffic, NIDSs that use a single sensor cannot give a complete runtime situational awareness of a complex system [7], [26]. This has led designers to use multiple distinct sensors in modern NIDSs. The ensemble of classifiers and sensor fusion techniques have been developed to answer the question of how to effectively combine multiple sensor outputs to perform classification and are discussed in the next section.

3.3 Techniques Used in Combining Multiple IDSs

3.3.1 Sensor Fusion based Intrusion Detection Techniques

Sensor fusion refers to the concept of combining sensor data collected from multiple distinct sources [87]. Its goal is to improve the accuracy or dependability of the data as compared to the case where the data from each source is used individually. Sensor fusion has been identified as a possible solution for improving the performance of IDSs. In general, sensor fusion techniques for improving IDS performance involve multiple observations, combinations of decisions and inferences via scenarios and models [18]. Most of the sensor fusion techniques that have been successfully applied to network intrusion detection are mainly based on methods like Bayesian Inference, Dempster-Shafer Belief Theory, Voting Fusion Theory, Neural Network and Logic OR operator to aggregate information. An overview of these techniques is given below.

3.3.1.1 Bayesian Inference

Bayesian inference provides the mathematical framework for calculating the likelihood of a hypothesis, given one or more pieces of evidence or observations, according to the rules of probability [88]. More specifically, it establishes the relationship between the revised (a posteriori) probability of a hypothesis, its apriori probability, and the conditional probability of an observation given each hypothesis. It uses this relationship to provide a posteriori probability of a hypothesis which is the probability of a hypothesis being true given the evidence. It is defined as follows:

Given hypotheses $H_1, H_2, ..., H_M$ and suppose that they are exhaustive and mutually exclusive hypotheses that explain the observed data S, then

$$p(H_j|S) = \frac{p(S|H_j)p(H_j)}{\sum_i p(S|H_i)p(H_i)}. \tag{3.14}$$

and

$$\sum_i p(H_i) = 1. \tag{3.15}$$

where $p(H_i)$ is the probability that the hypothesis is true before considering the data (evidence), $p(S|H_i)$ is the probability of obtaining the data S given that the hypothesis H_i is true and $p(H_j|S)$ is the revised probability of H_j after considering the data S (evidence).

For data that comes from several sensors $S_1, S_2, ..., S_n$

$$p(H_j|S_1 \cap ... \cap S_n) = \frac{p(H_j)p(S_1|H_j)p(S_2|H_j)...p(S_n|H_j)}{\sum_i p(H_i)p(S_1|H_i)p(S_2|H_i)...p(S_n|H_i)}. \tag{3.16}$$

The decision is usually based on the maximum a posteriori principle.

The applicability of Bayesian inference is limited by [18]

- the difficulty of defining prior probabilities for each hypothesis,
- the requirement of having to define the problem in terms of observations that are mutually exclusive and exhaustive, and
- its computational complexity when applied to large scale problems with multiple hypotheses and observations.

3.3.1.2 Dempster–Shafer Belief Theory

The Dempster–Shafer (DS) belief theory [89] is a mathematical theory of evidence that provides a means for combining evidence from different sources to arrive at a degree of belief (represented as a belief function) in a hypothesis that takes into account of all available evidence. It is a generalization of Bayesian inference that makes use of belief functions instead of probability distributions. It defines a framework of discernment (FOD) denoted as Θ which is a finite set of all possible mutually exclusive outcomes about some problem domain. The set of the possible mutually exclusive subsets of the elements of Θ including Θ is called a power set denoted by 2^{Θ}. Each subset is defined as the hypothesis. The basic probability assignment (BPA) reflects a degree of belief in a hypothesis or the degree to which the evidence supports the hypothesis. The BPA over Θ denoted by m maps the power set Θ to the interval [0, 1] such that the following conditions hold

$$\begin{cases} m(\emptyset) = 0 \\ \sum_{A \subseteq \Theta} m(A) = 1. \end{cases} \tag{3.17}$$

The elements of the power set that have non zero values of BPA are called focal elements. From the basic probability assignment, two functions are defined, namely, belief and plausibility functions. The belief function of set A quantifies the strength of the belief that event A occurs. The belief for set A is the sum of all the BPA of the proper subsets B of set A. The plausibility function of A is the degree to which one believes that A is not false. The plausibility for set A is the sum of all the sets B that intersect set A. The two functions can be formally represented as follows:

$$Bel(A) = \sum_{B \subset A} m(B). \tag{3.18}$$

$$Pl(A) = 1 - Bel(\overline{A}) = \sum_{A \cap B \neq \emptyset} m(B). \tag{3.19}$$

The belief and plausibility provide the lower and upper bounds of the probability interval that contains the precise probability of the set of interest.

Dempster's rule of combination is a rule for combining degrees of belief when they are based on independent items of evidence [89] and outputs a fused decision. Given several belief functions, that are not entirely conflicting, on the same Θ based on distinct or independent sources of evidence a new belief function using Dempster's rule of combination can be obtained. It is called the orthogonal sum of several belief functions. The orthogonal sum $Bel_1 \oplus Bel_2$ of two belief functions over Θ is a belief function whose focal elements are all the possible intersections between the combining focal elements and whose BPA is given by

$$m(A) = \begin{cases} \frac{\sum_{A_i B_j = A} m_1(A_i) m_2(B_j)}{1 - \sum_{A_i B_j = \emptyset} m_1(A_i) m2(B_j)}, & \text{when } A \neq \emptyset \\ 0, & \text{when } A = \emptyset. \end{cases} \tag{3.20}$$

The computational complexity involved in using DS belief theory grows exponentially with the number of hypotheses associated with the problem. The computational time necessary to perform inference using DS belief theory disqualifies its use in many time critical operations.

3.3.1.3 Voting Fusion Theory

Voting is one of the oldest and the most widely used fusion decision method [90]. The fusion unit arrives at an agreement by a voting scheme like majority voting, plurality voting, weighted majority voting, etc. In majority voting each sensor gives a single class label as an output and the final class label output is assigned to the class that receives more than half of the votes. The advantages of majority voting are its simplicity and low error count since it makes an error if the majority of the sensors are wrong. The drawback of majority voting is that no prediction is made by the combined sensors if there is no class label that receives more than half of the votes and the sample is rejected by the voting method [91]. In plurality voting, each sensor gives a single class label as an output and the final output is assigned to the class where most of the sensors have chosen as a class output. Plurality voting is easy to use and simple. The downside to plurality voting is the possibility of winning on a small number of votes and thus of a minority and probably erroneous win [91]. Plurality voting coincides with majority voting if two classes are considered. In weighted majority voting, the fusion rule assigns a weight to each sensor which

indicates the degree of importance of the sensor output with respect to the final output. More weights are assigned to accurate sensors.

3.3.1.4 Neural Networks

A neural network is a machine learning based computational model that is modelled from the human brain and nervous system. It is a network of artificial neurons in which each input feature called an input node is connected to one or more output nodes. A typical artificial neuron is depicted in Figure 3.3.

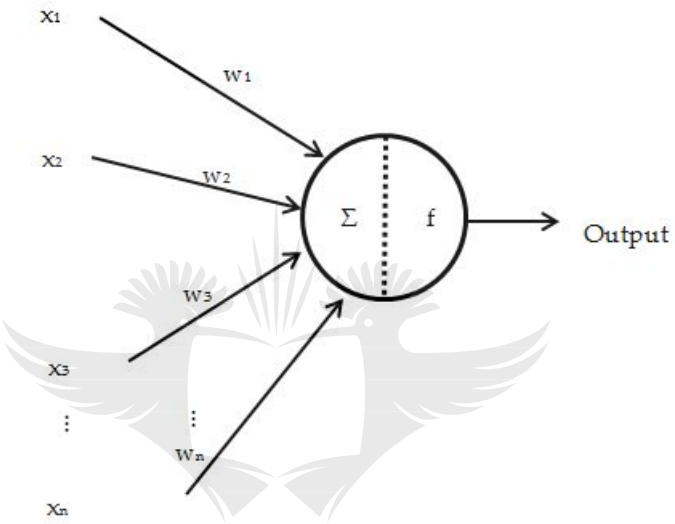


FIGURE 3.3: Artificial neuron.

The basic architecture of a neural network consists of three types of neuron layers: input, hidden and output layers. Figure 3.4 illustrates a multi-layered neural network. The nodes in the input layer represent the variables in the dataset. In the case of sensor fusion the input nodes represent the decisions of the sensors.

The hidden nodes in the hidden layer are linear combinations of the input variables (or sensor decisions) in the form

$$a_k = w_{k0} + \sum_{i}^{P} w_{ik} x_i. \tag{3.21}$$

where the parameters w_{k0} are the biases and parameters $w_{ik}$are the weights that represent the effect of the i_{th} feature on the hidden node k. The quantities a_k are typically transformed using a non-linear function $f(\cdot)$ such as sigmoid or hyperbolic tangent to give

$$h_k = f(a_k). \tag{3.22}$$

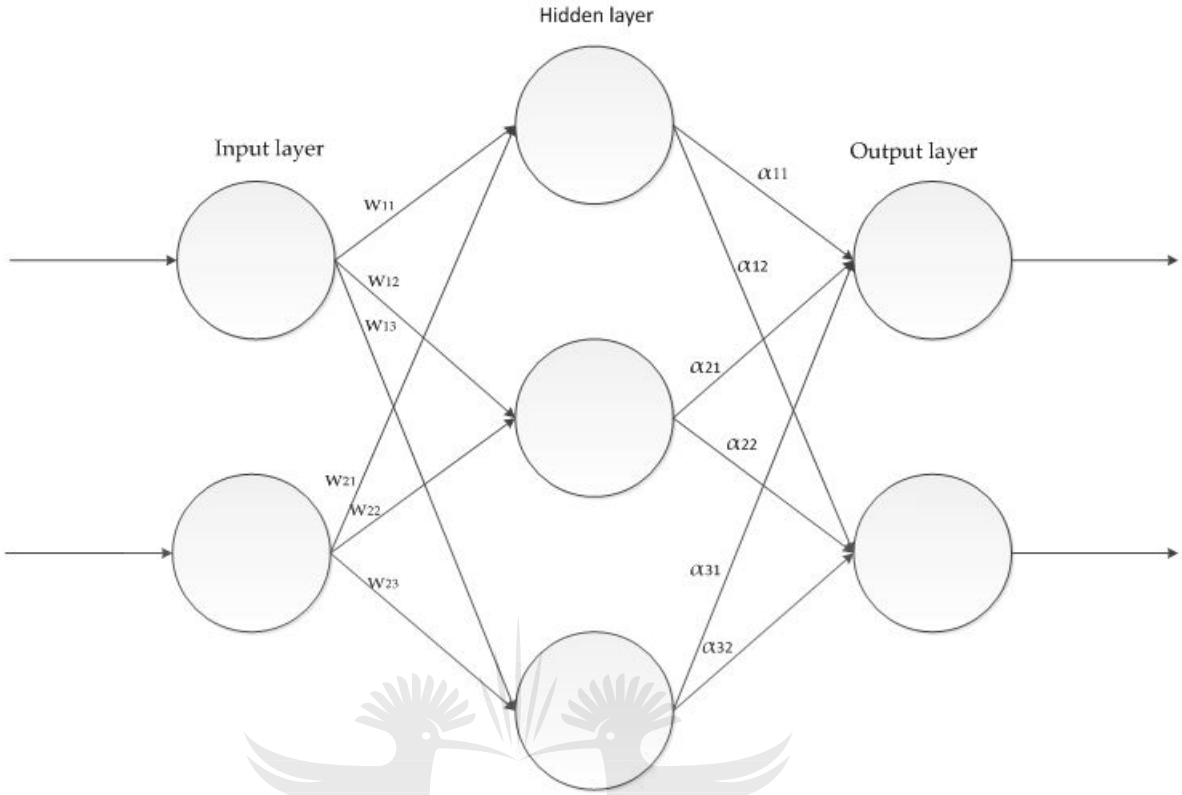


FIGURE 3.4: Multilayered artificial neural network.

A neural network may have more than one hidden layer and the number of hidden layers, H, is one of the model parameters that need to be tuned. After the number of hidden layers has been decided upon, the output is obtained from the following linear transformation:

$$o(x) = \alpha_0 + \sum_{k=1}^{H} \alpha_k h_k. \tag{3.23}$$

with $o(x)$ being the weighted sum of the hidden nodes and α_k the weight of each hidden node. The number of hidden nodes affects the ability of the neural network in separating the data. Too many ensure that the neural network correctly learns and is able to predict the data it was trained on correctly, however, its ability to generalise gets compromised. Too few may cause the neural network to fail to learn the relationships amongst the data resulting in unacceptable error. Therefore selection of the number of hidden nodes is an important decision.

Learning in neural network may fall under three categories:

- supervised learning in which an input vector is presented at the input layer together with the set of the desired outputs, one for each node, at the output layer.

- unsupervised learning in which an output unit is trained to respond to clusters of patterns with in the input. There is no prior set of classes into which the patterns are to be categorised, the neural network must develop its own representation of the input.
- reinforcement learning is learning what to do to optimise an objective function. The learner discovers the best actions by trial and error search.

The neural network is prone to overfitting due to the amount of parameters the model estimates. Several approaches have been proposed to address the overfitting in a neural network with the early stopping approach being the mostly utilised [92]. In early stopping, the neural network training is stopped at the smallest generalisation error [93].

3.3.1.5 Logic OR Operator

The logic OR operator assigns the Boolean value true if one or both operands are true and false otherwise. However, the use of the logic OR operator as a fusion method leads to a high false positive rate.

3.3.2 Ensemble Methods Used in Intrusion Detection

The ensemble learning method combines several individual classifiers to obtain better predictive performance than that obtained from the individual best classifier in the ensemble. In this method, multiple classifiers are combined together to yield a strong classifier. The use of several classifiers helps in finding the global solution that leads to reduced false alarm rate and increased detection accuracy [44]. According to Kuncheva [94], two types of ensembles exist, namely, decision optimization and coverage optimization. In decision optimization, the optimal combining method is chosen for a fixed ensemble of base classifiers (learners used in the ensemble). In coverage optimization, optimality is obtained by creating different base classifiers assuming a fixed combining method.

Several methods exist for constructing an ensemble of classifiers. These methods include manipulation of the training dataset, manipulation of the input features, manipulation of the output targets and injecting randomness into the learning algorithm. In the manipulation of the input features method, the learning algorithm is run on different subsets of the input features creating classifiers made up of different features. This approach improves the computational efficiency of the ensemble and increases its accuracy.

In the manipulation of the output targets method, the output classes of the training data that are given to the learning algorithm are altered. For example, if the number of the output labels, N, is large, a new learning problem can be constructed by randomly partitioning the N labels into two subsets A_l and B_l. The input data can then be relabelled so that any of the original in set A_l are labelled as 0 and the originals in set B_l are labelled as 1. This relabelled data is then given to the learning algorithm. If this process (generating different subsets A_l and B_l) is repeated K times an ensemble consisting of different K classifiers is created

An example of injecting randomness to the learning algorithm is when initial weights of the network are set randomly in the backpropagation algorithm for training the neural network. If the algorithm is applied to the same training instances but with different initial weights, the resulting classifiers can be quite different.

In the manipulation of the training dataset method, the learning algorithm is run on different distributions of the training dataset. This method is the mostly proposed and implemented in literature [44]. Two of the most popular methods that utilise the manipulation of training dataset method to construct an ensemble are boosting and bootstrap aggregating (Bagging) [95] and they have been successfully used in intrusion detection. Both of these techniques call a learning algorithm and run it several times on different distributions of the training datasets. An overview of the two methods is given below.

3.3.2.1 Boosting

In boosting, classifiers are trained in a sequence on different distributions of the original training dataset. One method of obtaining the various distributions of the original training dataset is to assign weights on the training dataset instances at each iteration with bigger weights given to the instances that were incorrectly classified by the previous classifier. Boosting combines the predictions of the classifiers and votes on the final prediction using weighted majority voting.

Boosting reduces the bias and variance of the predictions. Boosting methods are sensitive to outliers and noise, especially for small datasets [94]. A popular boosting algorithm is that of Freund and Schapire [96] called an adaptive boosting (AdaBoost) algorithm. The AdaBoost algorithm is an ensemble based machine learning algorithm that can be combined with other machine learning classifiers in order to boost their classification performances. In AdaBoost, a base learner is called for a specific number of iterations. For the first iteration, instances are given equal weights. For the successive iterations, the weights distribution is modified such that instances that were incorrectly classified by the base learner in the previous iteration are given bigger weights so that in the current iteration the base learner may

concentrate on classifying correctly those previously misclassified instances. The final decision of the ensemble is obtained by combining, using weighted majority voting, the decisions of the base learners in those iterations. The pseudocode for the AdaBoost algorithm from [97] can be found in Algorithm 2. AdaBoost is less prone to overfitting [97] and its implementation is easy. It is sensitive to noise and cannot adapt to new data changes. However, the latter drawback can be overlooked since the algorithm was not developed for that.

Algorithm 2 Pseudocode for the AdaBoost Algorithm

Given:$(x_1, y_1), ..., (x_m, y_m)$ where $x_i \in \mathfrak{X}$, $y_i \in \{-1, +1\}$
Initialize: $D_1(i) = 1/m$ for $i = 1, \cdots, m$.
For $t = 1, \cdots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathfrak{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error: $\varepsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$.
- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$.
- Update,for $i = 1, \cdots, m$:
  $D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
  where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis: $H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$.

3.3.2.2 Bootstraps Aggregating (Bagging)

In Bagging, each training set is created by forming a bootstrap replica of the original training set. That is, if there is a training set S with m instances a new set S' is generated from S by randomly sampling with replacement m instances. This creates samples where some of the instances will appear more than once while some do not appear at all. If T bootstrap samples are generated then for each bootstrap sample a classifier C_i is built. A final classifier $C*$ is built from these formed classifiers and the output of the final classifier is obtained by combining the outputs of the formed classifiers by majority voting with ties arbitrarily broken. The Bagging algorithm from [98] is given in Algorithm 3.

Bagging requires that the base classifiers (classifiers used in the ensemble) are unstable so that the small changes in the training set may result in different classifiers. Bagging reduces the variance of the predictions made by the base classifiers [41] and is effective on noisy data [44].

Algorithm 3 Bagging Algorithm

Input: training set S, learning algorithm $\mathcal{I}$, number of boostrap samples T

For $t = 1, \cdots, T$:

- $S' =$ bootstrap sample from S (i.i.d sample with replacement).
- $C_i = \mathcal{I}(S')$.

$C*(x) = \arg\max_{y \in Y} \sum_{i:C_i(x)=y} 1$ (the most often predicted label y)

Output: classifier $C*$.

3.4 Selection of Techniques

In intrusion detection literature techniques or algorithms used to build individual IDSs originating from different fields such as statistics, machine learning, data mining, etc. Therefore in this Thesis, we used statistical and learning based techniques to construct individual IDSs since statistical and machine learning based techniques are amongst the used techniques. The statistical based algorithms used in this work were an adaptive threshold and cumulative sum based algorithms. The adaptive threshold algorithm was used because of its ease to implement. The cumulative sum based algorithm was selected since it is commonly used in anomaly based intrusion detection. The learning based techniques used in this Thesis are decision tree, decision stump and fuzzy logic. The decision tree was used because of its ease to implement. The decision stump was chosen since it is one of the commonly used weak learners in ensemble methods. Uncertainty is an intrinsic characteristic of intrusion analysis, therefore fuzzy logic was used since it is a powerful learning based tool in reason under uncertainty.

For the sensor combining techniques used in intrusion detection, the logic OR operator as a sensor fusion method and ensemble methods boosting and Bagging were used. The logic OR operator was used since it easy to implement and always leads to improved detection probability. The outcomes of the two algorithms were simply 1 or 0 therefore it was not necessary to use techniques like DS and NN that are complex to implement or to use voting methods since we had only two outcome in the case of majority voting or having to find a strategy of assigning weights to each algorithm in order to perform weighted voting when the logic OR operator could easily combine the outcomes and provide an improved detection probability. To compensate for the high false positive rate associated with logic OR operator the parameters of the algorithms were tuned in such a way that the best accuracy was achieved. The ensemble techniques boosting and Bagging were selected since they are the mostly used ensemble methods and have the ability to optimise the performance of an IDS. This ability to optimise the performance of an IDS is utilised in finding the performance bounds of IDSs.

3.5 Datasets

The DARPA 99, NSL KDD and CICIDS2017 datasets were utilised in this Thesis. The individual IDSs were built and tested using the DARPA 99 and NSL KDD datasets. Even though the DARPA dataset is old, it was used in this study since it is the mostly used dataset in literature in testing detection and false alarm rates of an IDS [99]. Kumar and Kumar [44] criticised the use of the KDD 99, hence in this work the NSL KDD which is a refined KDD Cup 99 data was used. The NSL KDD advantage over the Kyoto 2006+ is that it specifies the attacks whereas the Kyoto 2006+ does not. The inability of the Kyoto 2006^{+} dataset to specify attacks is limiting in the sense that researchers may never be able to build IDSs for specific attacks. The IDS built using the Kyoto 2006+ will have to be used to detect all types of attacks. However, the work of Kayacik et al. [100] has indicated that certain attributes are more relevant in detecting certain attacks which means this finding can never be used to improve the performance of IDSs in the context of the Kyoto 2006^{+}. Therefore in this study, the NSL KDD dataset which allows the selection and use of relevant features in detecting specific attacks as indicated in [100] was used and using relevant features in detecting a specific attack improves the performance of the built IDS.

In this Thesis, the performance bounds of the NIDSs are obtained via a supervised learning approach. In supervised learning the classification algorithm learns from labelled training data. Therefore the NSL KDD and the CICIDS2017 datasets were used.

3.5.1 DARPA 99

Simulated network traffic obtained from MIT Lincoln Laboratory, i.e. DARPA 1999 dataset [101], was used in this study. The dataset consists of three weeks of training data where the training data for the first and third weeks are attack free. This training data was provided for offline evaluation of intrusion detection systems. The attack free training data was provided for training anomaly based IDSs. The test dataset consists of two weeks of network based attacks in the midst of normal background data. The dataset consists of different types of attacks that are categorised into probes, denial of service attack, user to root attack and remote to user attack. We used a training data that had no attacks taken on a Monday of the first week, with the packets collected between 08:00 to 17:00. Attack free data was selected so that we could have control on the attacks launched and test the utility of the privatised network trace in terms of false positive rate.

3.5.2 NSL KDD

The NSL KDD [102] dataset was also utilised in this study. The NSL KDD dataset was created from the KDD99 dataset [103] by taking out duplicate and redundant instances and decreasing the size of the dataset. The KDD99 dataset is a revised version of the DARPA 98 MIT Lincoln Lab dataset [101] that was summarized into network connections. Each network connection is a single row vector that consists of 41 features and is marked as either a particular attack type or normal. The list of all 41 features can be found in [100]. In this work, the network connections are also referred to as cases.

The dataset consists of different types of attacks that are categorised into denial of service attack, remote to user attack, user to root attack and probes and are depicted in Table 3.2. The four attack categories are described below.

- In denial of Service (DoS) legitimate users are prevented from utilising a service.
- In remote to local (R2L) the attacker attempts to get access to a machine (victim) that they do not have an account on.
- In user to root (U2R) the attacker attempts to get super user rights to a machine that they have local access to.
- In probe, the attacker attempts to get information about the target host.

TABLE 3.2: The different attacks in the KDD99 dataset that fall into the four attack categories

Attack Categories	Attacks
Denial of Service	Land, back, Neptune, smurf, pod and teardrop
User to Root	Buffer overflow, load module, perl and rootkit
Remote to local	Imap, ftp write, guess passwd, phf , multihop,spy, warezmaster and warezclient
Probes	Satan, ipsweep, nmap and portsweep

3.5.3 CICIDS2017

The Canadian Institute for Cybersecurity created the CICIDS2017 dataset [104] in 2017. It consists of realistic benign traffic and updated family of attacks [105].The dataset consists of 80 features. The dataset includes seven attack families, namely, brute force, heartbleed, botnet, denial of service, distributed denial of service, web and infiltration.

3.6 Summary

In this Chapter, some of the intrusion detection techniques used in individual and combining multiple intrusion detection systems used for classifying instances or detecting attacks were presented. These include: adaptive threshold algorithm, cumulative sum based algorithm, decision tree and fuzzy logic for individual IDSs and Bayesian inference, Dempster Shafer belief theory, voting fusion theory, neural networks, logic OR operator, boosting and Bagging for combining multiple IDSs. A brief description of the datasets that will be used to test the Thesis statements was given.

Chapter 4

Performance Evaluation of Network Intrusion Detection Systems for Detecting Transmission Control Protocol Synchronised Flooding Attack

This Chapter evaluates the performances of existing individual NIDSs and NIDSs based on combining multiple sensors for the detection of the TCP SYN flooding attack in order to illustrate what the existing NIDSs can achieve. This performance serves as a benchmark to what is currently achievable. The individual based intrusion detection algorithms that are considered in this study are the adaptive threshold algorithm, cumulative sum based algorithm, the decision tree and the fuzzy logic based system. The combining multiple sensors algorithm that is used is the logic OR operator.

4.1 Introduction

Transmission Control Protocol Synchronised (TCP SYN) flooding attack is an attack that leads to denial of service to legitimate clients. In this attack, the attacker sends a large number of TCP SYN messages to a server. The server replies with Synchronised/Acknowledgement (SYN/ACK) messages, however, the attacker never acknowledges these SYN/ACK messages resulting in a large number of half open connections on the server using up its resources. This continues until the entire server's resources are used up and the server is no longer able to accept any new TCP SYN connection request which leads to legitimate clients being unable to access the services offered by the server. Unavailability of service to clients is very costly to organisations that provide online services, for example, if an organisation

sells products online TCP SYN flooding attack will have adverse effect on the sales of those organisation's products and hence affect the revenue and profit made by that organisation. Network security measures are therefore needed to protect information systems against this threat or attack. The network security research community has proposed several methods of detecting TCP SYN flooding attack [36], [106]–[111]. These methods have their origin in statistics, machine learning and data mining , to mention a few. These methods utilise different network traffic measurements to detect TCP SYN flood attack, for example, the number of SYN packets received in a given time interval, filtering the network packets in terms of the TCP header and Internet Protocol header characteristics using payload, etc. Beaumont-Gay [112] compared three TCP SYN flood attack methods, which used different traffic measurements, in terms of detection time and quiescence time. Beaumont-Gay reported that one method was good at both times, while another method was good in detection time and the last method was good in quiescence time.

The existing individual NIDSs used in this Chapter originate from statistical and learning methods. The statistical based algorithms utilised are anomaly based algorithms called adaptive threshold and cumulative sum based algorithms. The learning based algorithms used are the decision tree and fuzzy logic system. The outcomes of the two anomaly based algorithms in detecting the TCP SYN flood attack are combined using the logic OR operator to improve the algorithms detection probability.

The rest of the Chapter is organised as follows, Section 4.2 implements two anomaly based algorithms for the detection of the TCP SYN flooding attacks. Furthermore, their outcomes are combined using the logic OR operator to improve the true positive rate (TPR) and detection delay (DD) of the NIDSs. Followed by the implementation of two learning based NIDSs for predicting the TCP SYN flood attack in Section 4.3. The Chapter is concluded by a brief summary.

4.2 Implementation of the Anomaly based Intrusion Detection Algorithms for Detecting the TCP SYN Flooding Attack

Two anomaly detection algorithms for detecting the TCP SYN flooding attack from the work of Siris and Papagalou [36] are implemented on synthetic attacks generated according to a Poisson process as in [36] and constant rate arrivals. The two algorithms are a cumulative sum (CUSUM) based algorithm and an adaptive threshold algorithm which have their origin in Statistics. Furthermore, the decisions of the algorithms are combined using the logic OR operator in order to improve the detection probability since sensor fusion is known for improving the detection probability. In their work [36], the cumulative sum (CUSUM) based algorithm

performed better than the adaptive threshold algorithm in terms of detecting low intensity SYN flood attacks. Hence, in this research the decision of the two algorithms are combined to get the improved detection probability. This work extends the work of [36] and literature by introducing new synthetic attacks that are generated according to constant rate arrivals, evaluating the performance of [36] algorithms in these new attacks and comparing the performance of these algorithms on the two attacks. The detecting probability of the two algorithms is enhanced by fusing the algorithms' outcomes.

4.2.1 Dataset

Simulated network traffic obtained from MIT Lincoln Laboratory, i.e. DARPA 1999 dataset, was used in this section and the detailed description of this data is given in Section 3.5. We used attack-free data taken on a Monday, with the packets collected between 08:00 to 17:00 i.e. collected over 9 hours. The data was filtered for the Transmission Control Protocol (TCP) Synchronise (SYN) packets. The number of TCP SYN packets in a 10 second interval was the traffic measurement of interest in this study. The number of TCP SYN packets in 10 second intervals over the 9 hours were determined and used to detect the TCP SYN flooding attack using the two algorithms. Algorithm 4 illustrates how the number of TCP SYN packets in 10 second intervals over the 9 hours were determined.

Algorithm 4 Attack free TCP SYN packet counts in every 10s interval.

Given the arrival times of the attack free TCP SYN packets

- Break the range of the TCP SYN packets arrival times to 10s intervals
- Calculate the number of TCP SYN packets that arrived in each 10s interval
- Store these packet counts in a vector

4.2.2 Attack Generation

In this section, the attacks were generated synthetically. We generated two types of attacks, namely, those that arrived in accordance to a Poisson process as in [36] and those that arrived at a constant rate. In a Poisson, process the arrivals follow a Poisson distribution and the interarrival times between successive arrivals are independent and exponentially distributed. In the constant arrival rate, the interarrival times between successive arrivals are the same. Similarly to the work of [36], we generated the Poisson attacks such that the interarrival times between successive attacks follow an exponential distribution with a mean of 460 time intervals. This resulted in the average number of attacks to be approximately seven attacks in 9 hours (08:00 to 17:00). The work of [36] used synthetic attacks (Poisson process attacks) in order to have control over the characteristics of the

attacks. Similarly to the work of [36], we used synthetic attacks in order to have control of the characteristics of the attacks. The constant rate attacks were chosen arbitrarily as different attacks since this work extends the work of [36] by comparing the performance of their algorithms to two different attacks. Different attack arrival rates were not explored since this work extends the work of [36] by comparing the performance of the two anomaly based algorithms on the two attacks. The duration of both attacks was 300s since in [36] the attacks on average had a duration of 600s.

Siris and Papagalou [36] generated low and high intensity attacks. The high intensity attacks had a mean amplitude that was 250% more than the mean traffic rate while the low intensity attacks had a mean amplitude that was 50% of the mean traffic rate. In this study, we only considered low intensity attacks since being able to detect them is important because they are not so obvious in the network traffic and they were defined exactly like in [36]. Algorithms 5 and 6 respectively illustrates how the attacks arriving according to a constant rate and Poisson process were added to the traffic free data.

Algorithm 5 Addition of the attacks arriving according to a constant rate.

Do

Break the range of 10s intervals obtained in Algorithm 4 into 300s intervals (30 10s intervals, since the duration of the attack is 300s)
For every 300s intervals

1. Calculate the mean number of attack free TCP SYN packets in each 300s interval, denoted by meanRate.
2. Using the above mean, calculate the number of TCP SYN attack packets to be added in the intervals such that the mean amplitude of the traffic in each 300s interval is 50% of the mean traffic rate as follows:

   number of attack TCP SYN packet to be added = $0.5 * \text{meanRate}$
3. Add the obtained number of attack TCP SYN packets in each 300s interval to the number of attack free TCP SYN packets in the corresponding 300s intervals.

end

4.2.3 Performance Metrics and Parameters

Siris and Papagalou [36] used detection delays (DD), true positive rate (TPR) or detection probability (DP) and false positive rate (FPR) or false alarm rate (FAR) as performance metrics. In this research, the same performance metrics were used. Algorithms 7 and 8 illustrate how the true positive and false positive rates of the two anomaly based intrusion detection algorithms were respectively obtained.

We included the false negative rate (FNR) in order to determine the equal error rate (EER). The EER was used in choosing the two algorithms' detection thresholds to be used in the fusion of the two algorithms' outcomes. Accuracy was also included as a

Algorithm 6 Addition of the attacks arriving according to a Poisson process.

Do

1. Generate Poisson arrivals to get attack arrival times.
2. For each generated attack arrival time determine the 10s interval it falls in.
3. From each determined 10s interval add 29 more 10s intervals to make 300s (since the attack duration is 300s)

For each of the determined 300s intervals

1. Calculate the mean number of attack free TCP SYN packets in each 300s interval first, denoted by meanRateP.
2. Using the above mean, calculate the number of TCP SYN attack packets to be added in the intervals such that the mean amplitude of the traffic in each 300s interval is 50% more than the mean traffic rate as follows:

   number of attack TCP SYN packet to be added = $0.5 * \text{meanRateP}$
3. Add the obtained number of attack TCP SYN packets in each 300s interval to the number of attack free TCP SYN packets in the corresponding 300s intervals.

end

end

performance metric that was used to tune the parameters of the two anomaly based algorithms. These performance measures were described in Section 3.1.

Initially, the two anomaly based algorithms were implemented for both the Poisson process and constant rate attacks using the parameter values that were used by Siris and Papagalou [36] of an α of 0.5 and a β of 0.98. The detection threshold parameters k and h ranging between 1 and 10 and 1 and 20 respectively were used. The detection threshold values were selected such that the TPR and FPR values were forced to approach zero in order to obtain the EER value. The EER value was utilised to decide on the detection thresholds of the two anomaly based algorithms to be used in fusing the two algorithm outcomes.

For the constant rate attacks,

- the EER of the adaptive threshold algorithm fell between $k = 2$ and $k = 3$, therefore, our optimal k is any $k \leq 2$ i.e. where the false negative rate is less than the false positive rate. Figure 4.1a depicts this.
- the EER of the CUSUM algorithm fell between $h = 6$ and $h = 7$ as shown in Figure 4.1b. Therefore the optimal h is any $h \leq 6$.
- the results of the adaptive threshold algorithm for $k = 1, 2$ and 3 thresholds were combined with the results of the CUSUM algorithm for $h = 5, 6$ and 7 thresholds using the logic OR operator. We included $k = 3$ and $h = 7$ thresholds so that we do not compare the results for only two threshold values and these thresholds can be used when one wants to also operate around the EER. We chose $h = 5$ and 6 thresholds (instead of $h = 1$ and 2) since they are two thresholds away from the EER like $k = 1$ and 2 are two thresholds

Algorithm 7 True positive rate calculation for the two anomaly based intrusion detection algorithms for the two attacks.

Given the attacked data and the 300s intervals (30 10s intervals) where the attacks were added.
Do,

- For each of the considered 300s intervals
  check if there is at least one detection in the 30 10s intervals
  if so
  true positive = 1
  else
  true positive = 0
  end
  store the outcome (0 or 1) in a vector.
  end

- $TPR = \frac{\text{number of ones in the above vector}}{\text{number of the considered 300s intervals}}$

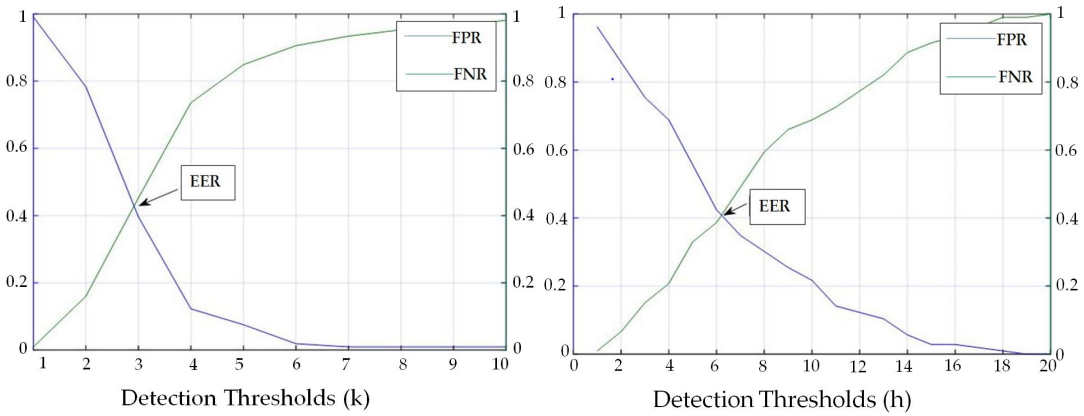
end

Algorithm 8 False positive rate calculation for the two anomaly based intrusion detection algorithms for the two attacks.

Given the attack free data and the 300s intervals (30 10s intervals) that correspond to the
300s intervals where the attacks were added.

- For each of the considered 300s intervals check if there is at least one detection in the 30 10s intervals
  if so
  false positive = 1
  else
  false positive = 0
  end
  store the outcome (0 or 1) in a vector.
  end
  $FPR = \frac{\text{number of ones in the above vector}}{\text{number of the considered 300s intervals}}$

away from the EER. Therefore we used two thresholds before EER (optimal thresholds) and one threshold after the EER as a standard procedure for choosing thresholds for fusing the decisions of the two anomaly based NIDSs using the logic OR operator.



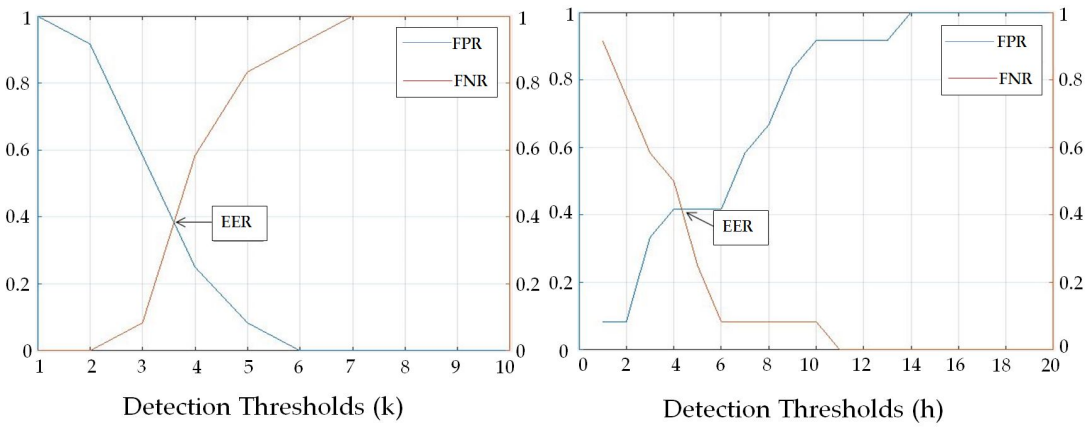
(A) adaptive threshold algorithm.

(B) CUSUM based algorithm.

FIGURE 4.1: False negative rates and false positive rates plotted against the detection thresholds of the two algorithms for the constant rate attacks at α of 0.5 and a β of 0.98.

For the Poisson process attacks,

- the EER of the adaptive threshold algorithm fell between $k = 3$ and $k = 4$, therefore, the optimal k is any $k \leq 3$. Figure 4.2a illustrates this.
- the EER of the CUSUM algorithm fell between $h = 4$ and $h = 5$ as shown in Figure 4.2b. Therefore the optimal h is any $h \leq 4$.
- the results of the adaptive threshold algorithm for $k = 2, 3$ and 4 thresholds were combined with the results of the CUSUM algorithm for $h = 3, 4$ and 5 thresholds using the logic OR operator. The three thresholds per threshold parameter were chosen since we use two thresholds before EER (optimal thresholds) and one threshold after the EER as a standard procedure for choosing thresholds for fusing the decisions of the two anomaly based NIDSs using the logic OR operator.



(A) adaptive threshold algorithm.

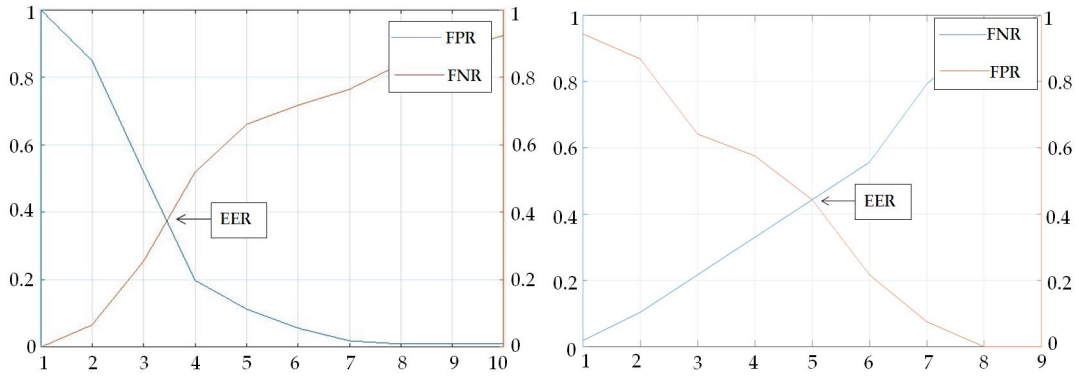
(B) CUSUM based algorithm.

FIGURE 4.2: False negative rates and false positive rates plotted against the detection thresholds of the two algorithms for the Poisson process attacks at α of 0.5 and a β of 0.98.

We then tuned the parameters for the constant rate and Poisson process attacks instead of just using the parameters proposed by [36]. The parameters were chosen based on the alpha beta combination that yielded the highest accuracy across all thresholds. For the constant rate attacks the chosen parameters were an α of 0.2 and a β of 0.99 with k ranging between 1 and 10 for the adaptive threshold algorithm and an α of 0.21 and a β of 0.98 with h ranging between 1 and 9 for the CUSUM based algorithm. For the Poisson process attacks the chosen parameters were an α of 0.6 and a β of 0.98 with k ranging between 1 and 10 for the adaptive threshold algorithm and an α of 0.8 and a β of 0.97 with h ranging between 1 and 12 for the CUSUM based algorithm.

For the constant rate attacks,

- the EER of the adaptive threshold algorithm fell between $k = 3$ and $k = 4$, therefore, the optimal k is any $k \leq 3$. Figure 4.3a shows this.
- the EER of the CUSUM algorithm fell at $h = 5$ as depicted in Figure 4.3b. Therefore the optimal h is any $h \leq 5$.
- the results of the adaptive threshold algorithm for $k = 2, 3$ and 4 thresholds were combined with the results of the CUSUM algorithm for $h = 3, 4$ and 6 thresholds using the logic OR operator. The three thresholds per threshold parameter were chosen since we use two thresholds before EER (optimal thresholds) and one threshold after the EER as a standard procedure for choosing thresholds for fusing the decisions of the two anomaly based NIDSs using the logic OR operator.



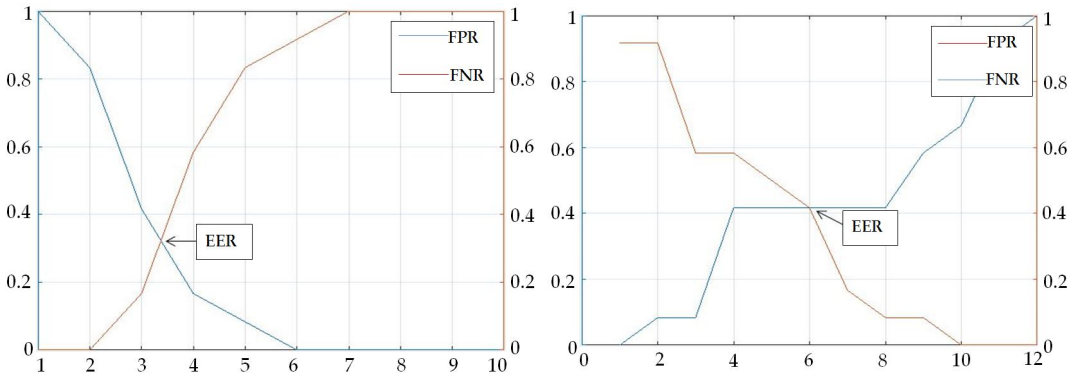
(A) adaptive threshold algorithm.

(B) CUSUM based algorithm.

FIGURE 4.3: False negative rates and false positive rates plotted against the detection thresholds of the two algorithms for the constant rate attacks after tuning the parameters.

For the Poisson process attacks,

- the EER of the adaptive threshold algorithm fell between $k = 3$ and $k = 4$, therefore, the optimal k is any $k \leq 3$. Figure 4.4a depicts this.
- the EER CUSUM algorithm was at $h = 6$ as shown in Figure 4.4b. Therefore the optimal h is any $h \leq 5$.
- the results of the adaptive threshold algorithm for $k = 2, 3$ and 4 thresholds were combined with the results of the CUSUM algorithm for $h = 4, 5$ and 7 thresholds using the logic OR operator. The three thresholds per threshold parameter were chosen since we use two thresholds before EER (optimal thresholds) and one threshold after the EER as a standard procedure for choosing thresholds for fusing the decisions of the two anomaly based NIDSs using the logic OR operator.



(A) adaptive threshold algorithm.

(B) CUSUM based algorithm.

FIGURE 4.4: False negative rates and false positive rates plotted against the detection thresholds of the two algorithms for the Poisson process attacks after tuning the parameters.

Based on the optimal thresholds the average accuracy of the CUSUM based algorithm was the lowest of the three algorithms in both attacks whereas in the work of [36] the CUSUM based algorithm outperformed the adaptive threshold algorithm. To improve the performance of the CUSUM based algorithm we changed the calculation of variance from using the previous thirty 10s interval samples to using the exponential weighted moving variance for both attacks. We tuned the CUSUM based algorithm parameters to obtain the alpha and beta combination that leads to the highest average accuracy across all thresholds. For the constant rate attacks, the chosen parameters were an α of 1 and a β of 0.46 with h ranging between 1 and 100. For the Poisson process attacks, the chosen parameters were an α of 1 and a β of 0.4 with h ranging between 1 and 100.

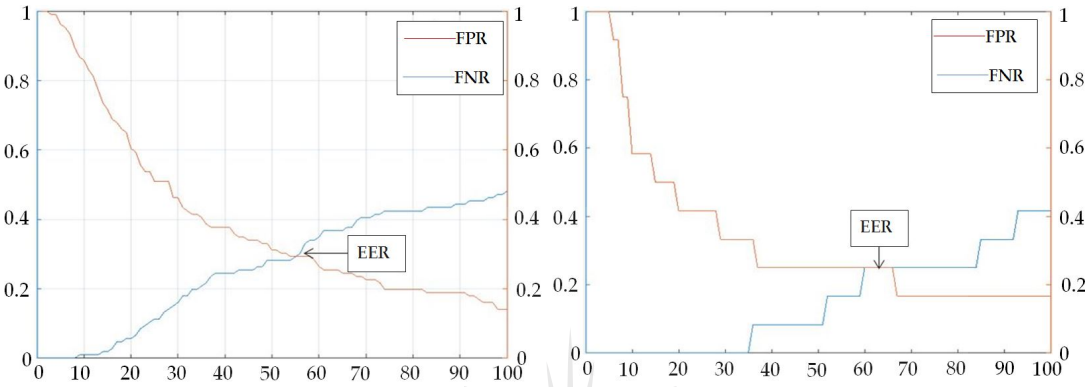
For the constant rate attacks,

- the CUSUM algorithm EER was at $h = 55$ as illustrated in Figure 4.5a . Therefore the optimal h is any $h \leq 54$.
- the results of the adaptive threshold algorithm for $k = 2, 3$ and 4 thresholds (previously obtained when we tuned the parameters the first time) were combined with the results of the CUSUM algorithm for $h = 53, 54$ and 56 thresholds using the logic OR operator. The three thresholds were chosen since we use two thresholds before EER (optimal thresholds) and one threshold after the EER as a standard procedure for choosing thresholds for fusing the decisions of the two anomaly based NIDSs using the logic OR operator.

For the Poisson process attacks,

- the CUSUM algorithm EER was at $h = 60$ to 66 as illustrated in Figure 4.5b. Therefore the optimal h is any $h \leq 59$.

- the results of the adaptive threshold algorithm for $k = 2, 3$ and 4 thresholds (previously obtained when we tuned the parameters the first time) were combined with the results of the CUSUM algorithm for $h = 58, 59$ and 67 thresholds using the logic OR operator. The three thresholds were chosen since we use two thresholds before EER (optimal thresholds) and one threshold after the EER as a standard procedure for choosing thresholds for fusing the decisions of the two anomaly based NIDSs using the logic OR operator.



(A) Constant rate attacks. (B) Poisson process attacks.

FIGURE 4.5: False negative rates and false positive rates plotted against the detection thresholds of the CUSUM based algorithm after modifying the variance of the CUSUM based algorithm for the two attacks.

4.2.4 Results

4.2.4.1 Poisson Process Attacks Results

Results presented in Tables 4.1, 4.2 and 4.3 show that the logic OR operator detects TCP SYN flood attacks significantly better and quicker, however, its false positive rate is higher than those of the other two algorithms. Furthermore, the accuracy of the logic OR operator is worse than those of the other two algorithms. Based on the optimal threshold values in Tables 4.1, 4.2 and 4.3 the logic OR operator's average detection probability is significantly better than those of the two algorithms with the percentage improvement of 45% to that of the CUSUM based algorithm and 14% to that of the adaptive threshold algorithm.

4.2.4.2 Poisson Process Attacks Results after Tuning the Parameters

Results presented in Tables 4.4, 4.5 and 4.6 show that the logic OR operator detects TCP SYN flood attacks significantly better and quicker. Furthermore, the logic OR operator has an accuracy that is slightly better than that of CUSUM based algorithm,

TABLE 4.1: Adaptive threshold algorithm results for detecting Poisson process attacks at $\alpha = 0.5$ and $\beta = 0.98$

Optimal Threshold	Detection Probability	False Positive Rate	Accuracy	Detection Delay (10s interval)
k = 2	1.0000	0.9167	0.5417	10.4
k = 3	0.9167	0.5833	0.6667	14.3
k = 4	0.4167	0.2500	0.5833	20.8
Average	0.7778	0.5833	0.5972	15.1

TABLE 4.2: Cumulative sum based algorithm for detecting Poisson process attacks at $\alpha = 0.5$ and $\beta = 0.98$

Optimal Threshold	Detection Probability	False Positive Rate	Accuracy	Detection Delay (10s interval)
h = 3	0.6667	0.5833	0.54167	17.3
h = 4	0.5833	0.5000	0.54167	18.2
h = 5	0.5833	0.2500	0.6667	20.7
Average	0.6111	0.4444	0.5833	18.7

TABLE 4.3: Logic OR operator results for detecting the Poisson process attacks

Optimal Threshold	Detection Probability	False Positive Rate	Accuracy	Detection Delay (10s interval)
k = 2, h = 3	1.0000	0.9167	0.5417	9.3
k = 3, h = 4	0.9167	0.9167	0.5000	12.2
k = 4, h = 5	0.7500	0.5833	0.5834	16.8
Average	0.8889	0.8056	0.5417	12.7

however, its false positive rate is higher than those of the other two algorithms. Based on the optimal thresholds values in Tables 4.4, 4.5 and 4.6 the logic OR operator's:

- average detection probability is significantly better than those of the two algorithms with the percentage improvement of 52% to that of the CUSUM based algorithm and 19% to that of the adaptive threshold algorithm.
- average accuracy is slightly better than that of the CUSUM based algorithm by 3%.

TABLE 4.4: Adaptive threshold algorithm results for detecting Poisson process attacks after tuning the parameters with $\alpha = 0.6$ and $\beta = 0.98$

Optimal Threshold	Detection Probability	False Positive Rate	Accuracy	Detection Delay (10s interval)
k = 2	1.0000	0.8333	0.5833	10.4
k = 3	0.8333	0.4167	0.7083	15.1
k = 4	0.4167	0.1667	0.6250	20.8
Average	0.7500	0.4722	0.6389	15.4

TABLE 4.5: Cumulative sum based algorithm for detecting Poisson process attacks after tuning the parameters with $\alpha = 0.8$ and $\beta = 0.97$

Optimal Threshold	Detection Probability	False Positive Rate	Accuracy	Detection Delay (10s interval)
h = 4	0.5833	0.5833	0.500	17.8
h = 5	0.5833	0.5000	0.5417	19.3
h = 7	0.5833	0.1667	0.7083	21.6
Average	0.5833	0.4167	0.5833	19.6

TABLE 4.6: Logic OR operator results for detecting the Poisson process attacks after tuning the parameters

Optimal Threshold	Detection Probability	False Positive Rate	Accuracy	Detection Delay (10s interval)
k = 2, h = 4	1.0000	0.9167	0.5417	9.3
k = 3, h = 5	0.9167	0.7500	0.5834	13.1
k = 4, h = 7	0.7500	0.3333	0.7084	16.8
Average	0.8889	0.6667	0.6111	13.1

4.2.4.3 Poisson Process Attacks Results with Modified CUSUM based Algorithm Variance

Results presented in Tables 4.4, 4.7 and 4.8 show that the logic OR operator detects TCP SYN flood attacks significantly better and quicker. Furthermore, the logic OR operator has an accuracy that is slightly better than that of the adaptive threshold algorithm, however, its false positive rate is higher than those of the other two algorithms. Based on the optimal threshold values in Tables 4.4, 4.7 and 4.8 the logic OR operator's:

- average detection probability is significantly better than those of the two algorithms with the percentage improvement of 14% to that of the CUSUM based algorithm and 22% to that of the adaptive threshold algorithm.
- average accuracy is slightly better than that of the adaptive threshold algorithm by 2%.

Modifying the variance of the CUSUM based algorithm also lead to a higher average detection probability, average accuracy and average detection delay and lower average false positive rate for the Poisson attacks than those of the adaptive threshold algorithm. This result agrees with the work of [36] which means how the variance is calculated in the CUSUM algorithm has an effect on the performance of this algorithm.

TABLE 4.7: Cumulative sum based algorithm for detecting Poisson process attacks with modified CUSUM based algorithm variance with $\alpha = 1$ and $\beta = 0.4$

Optimal Threshold	Detection Probability	False Positive Rate	Accuracy	Detection Delay (10s interval)
h = 58	0.8333	0.2500	0.7917	13.8
h = 59	0.8333	0.2500	0.7917	13.8
h = 67	0.7500	0.1667	0.7917	14
Average	0.8056	0.2222	0.7917	13.9

TABLE 4.8: Logic OR operator results for detecting the Poisson process attacks with modified CUSUM based algorithm variance

Optimal Threshold	Detection Probability	False Positive Rate	Accuracy	Detection Delay (10s interval)
k = 2, h = 58	1.0000	0.9167	0.5417	7.0
k = 3, h = 59	0.9167	0.5833	0.6667	8.9
k = 4, h = 67	0.8333	0.3333	0.7500	10.9
Average	0.9167	0.6111	0.6528	8.9

4.2.4.4 Discussion of the Algorithms Performance on Detecting the Poisson Process Attacks

What is observed from the results in Subsubsections 4.2.4.1, 4.2.4.2 and 4.2.4.3 is that

- the logic OR operator has outperformed the two anomaly based algorithms in terms of detection probability and detection delay. These results are to be expected since the logic OR operator takes the best outcome(s) of the two anomaly based algorithms. That is, if at least one algorithm correctly detected the attack then the logic OR operator will detect that attack.
- the logic OR operator has the worst false positive rate, this is justified since it aggregates the errors of the two anomaly based algorithms. That is, if at least one of the two anomaly based algorithm incorrectly labels normal traffic as an attack then the logic OR operator will label that instant as an attack even if the other algorithm correctly labelled that instant as normal. This increases the number of false positives and hence the false positive rate.
- the accuracy of the logic OR operator is at least worse than one of the anomaly based algorithms. This is to be expected due to the high false positive rate of the logic OR operator.

4.2.4.5 Comparing the Algorithms Results for the Poisson Process Attacks at the Different Parameters

In this work, the performance of this algorithm at different parameters is based on the tradeoff between the FNR and the FPR where the EER is used to gauge the performance of the algorithm. The algorithms' results at $\alpha = 0.5$ and $\beta = 0.98$, Figures 4.2a and 4.2b, are used as a baseline for comparison. When the parameters of the adaptive threshold and CUSUM based algorithms were tuned (Figures 4.4a and 4.4b), based on the optimal thresholds the new parameters lead to the following:

The adaptive threshold algorithm's EER at $\alpha = 0.6$ and $\beta = 0.98$ is approximately 0.34 (as seen in Figure 4.4a) and is lower than the algorithm's EER at $\alpha = 0.5$ and $\beta = 0.98$ of approximately 0.37 (as seen in figure 4.2a). This means the algorithm with $\alpha = 0.6$ and $\beta = 0.98$ is more accurate.

For the CUSUM based algorithm the obtained parameters $\alpha = 0.8$ and $\beta = 0.97$ led to the same EER of approximately 0.41 as the algorithm at $\alpha = 0.5$ and $\beta = 0.98$ (see Figures 4.2b and 4.4b). This means the accuracy of the algorithm at the different parameters is the same.

4.2.4.6 Constant Rate Attacks Results

Results presented in Tables 4.9, 4.10 and 4.11 show that the logic OR operator detects TCP SYN flood attacks significantly better and quicker. Furthermore, the logic OR operator has improved accuracy as compared to the adaptive threshold algorithm, however, its false positive rate is higher than those of the other two algorithms. Based on the optimal thresholds values in Tables 4.9, 4.10 and 4.11 the logic OR operator's:

- average detection probability is significantly better than those of the two algorithms with the percentage improvement of 48% to that of the CUSUM based algorithm and 12% to that of the adaptive threshold algorithm.
- average accuracy is slightly better than that of the adaptive threshold algorithm by 1.8%.

TABLE 4.9: Adaptive threshold algorithm results for detecting constant rate attacks at $\alpha = 0.5$ and $\beta = 0.98$

Optimal Threshold	Detection Probability	False Positive Rate	Accuracy	Detection Delay (10s interval)
k = 1	0.9906	0.9906	0.5	4.1
k = 2	0.8396	0.7830	0.5283	12.1
k = 3	0.5472	0.3962	0.5755	19.3
Average	0.7925	0.7233	0.5346	11.8

TABLE 4.10: Cumulative sum based algorithm for detecting constant rate attacks at $\alpha = 0.5$ and $\beta = 0.98$

Optimal Threshold	Detection Probability	False Positive Rate	Accuracy	Detection Delay (10s interval)
h = 5	0.6698	0.5566	0.5566	14.9
h = 6	0.6132	0.4245	0.5944	17.3
h = 7	0.5094	0.3491	0.5802	19.9
Average	0.5975	0.4434	0.5770	17.4

TABLE 4.11: Logic OR operator for detecting constant rate attacks

Optimal Threshold	Detection Probability	False Positive Rate	Accuracy	Detection Delay (10s interval)
k = 1, h = 5	1.0000	1.0000	0.5	3.1
k = 2, h = 6	0.9245	0.8396	0.5425	8.3
k = 3, h = 7	0.7358	0.5566	0.5896	14.1
Average	0.8868	0.7987	0.5440	8.5

4.2.4.7 Constant Rate Attacks Results after Tuning the Parameters

Results presented in Tables 4.12, 4.13 and 4.14 show that the logic OR operator detects TCP SYN flood attacks significantly better and quicker. Furthermore, the logic OR's false positive rate is the highest resulting in the worst accuracy. Based on the optimal thresholds values in Tables 4.12, 4.13 and 4.14 the logic OR operator's:

- average detection probability is significantly better than those of the two algorithms with the percentage improvement of 26% to that of the CUSUM based algorithm and 11% to that of the adaptive threshold algorithm.

TABLE 4.12: Adaptive threshold algorithm results for detecting constant rate attacks after tuning the parameters to $\alpha = 0.2$ and $\beta = 0.99$

Optimal Threshold	Detection Probability	False Positive Rate	Accuracy	Detection Delay (10s interval)
k = 2	0.9340	0.8491	0.5425	8.8
k = 3	0.7453	0.5189	0.6132	14.8
k = 4	0.4811	0.1981	0.6415	20.8
Average	0.7201	0.5220	0.5991	14.8

TABLE 4.13: Cumulative sum based algorithm for detecting constant rate attacks after tuning the parameters to $\alpha = 0.21$ and $\beta = 0.98$

Optimal Threshold	Detection Probability	False Positive Rate	Accuracy	Detection Delay (10s interval)
h = 3	0.7831	0.6415	0.5708	11.0
h = 4	0.6698	0.5755	0.5472	14.1
h = 6	0.4434	0.2170	0.6132	22.9
Average	0.6321	0.4780	0.5770	16.0

TABLE 4.14: Logic OR operator for detecting constant rate attacks after tuning the parameters

Optimal Threshold	Detection Probability	False Positive Rate	Accuracy	Detection Delay (10s interval)
k = 2, h = 3	0.9623	0.9245	0.5189	5.1
k = 3, h = 4	0.8679	0.7547	0.5566	8.8
k = 4, h = 6	0.566	0.3774	0.5943	18.1
Average	0.7987	0.6855	0.5566	10.7

4.2.4.8 Constant Rate Attacks Results with Modified CUSUM based Algorithm Variance

Results presented in Tables 4.12, 4.15 and 4.16 show that the logic OR operator detects TCP SYN flood attacks significantly better and quicker. Furthermore, the logic OR operator has improved accuracy as compared to the adaptive threshold algorithm, however, its false positive rate is higher than those of the other two algorithms. Based on the optimal threshold values in Tables 4.12, 4.15 and 4.16 the logic OR operator's:

- average detection probability is significantly better than those of the two algorithms with the percentage improvement of 30% for the CUSUM based algorithm and 28% for the adaptive threshold algorithm.
- average accuracy is slightly better than that of the adaptive threshold algorithm by 5%.

Modifying the variance of the CUSUM based algorithm also lead to a lower average false positive rate and a higher average accuracy for the constant rate attacks than those of the adaptive threshold algorithm. Similarly to the Poisson process attacks, this result means the manner in which the variance is calculated in the CUSUM algorithm has an effect on the performance of this algorithm.

4.2.4.9 Discussion of the Algorithms Performance on Detecting the Constant Rate Attacks

The logic OR operator performed in a similar manner to the Poisson attacks in detecting the constant arrival attacks.

4.2.4.10 Comparing the Algorithms Results for the Constant Rate Attacks at the Different Parameters

The algorithms' results at $\alpha = 0.5$ and $\beta = 0.98$, Figures 4.1a and 4.1b are used as a baseline for comparison. When the parameters of the adaptive threshold and

TABLE 4.15: Cumulative sum based algorithm with modified variance for detecting constant rate attacks at $\alpha = 1$ and $\beta = 0.46$

Optimal Threshold	Detection Probability	False Positive Rate	Accuracy	Detection Delay (10s interval)
h =53	0.7170	0.3019	0.7075	16.3
h = 54	0.7170	0.2925	0.7123	16.3
h = 56	0.6981	0.2925	0.7028	16.6
Average	0.7107	0.2956	0.7075	16.4

TABLE 4.16: Logic OR operator results with modified CUSUM based algorithm variance

Optimal Threshold	Detection Probability	False Positive Rate	Accuracy	Detection Delay (10s interval)
k = 2, h = 53	0.9811	0.8962	0.5425	5.8
k = 3, h = 54	0.9151	0.6415	0.6368	9.1
k = 4, h = 56	0.8679	0.4151	0.7264	11.4
Average	0.9214	0.6509	0.6352	8.8

CUSUM based algorithms were tuned (Figures 4.3a and 4.3b), based on the optimal thresholds the new parameters lead to the following:

For the adaptive threshold algorithm the obtained parameters $\alpha = 0.2$ and $\beta = 0.99$ led to EER of approximately 0.37 (as seen in figure 4.3a) and is lower than the algorithm's EER at $\alpha = 0.5$ and $\beta = 0.98$ of approximately 0.425 (as seen in figure 4.1a). This means the algorithm with $\alpha = 0.2$ and $\beta = 0.99$ is more accurate.

For the CUSUM based algorithm the obtained parameters $\alpha = 0.21$ and $\beta = 0.98$ led to an EER of 0.44 (as seen in figure 4.3b) and is higher than the algorithm's EER at $\alpha = 0.5$ and $\beta = 0.98$ of approximately 0.41 (as seen in figure 4.1b). This means the algorithm with $\alpha = 0.21$ and $\beta = 0.98$ is less accurate as compared to the algorithm with $\alpha = 0.5$ and $\beta = 0.98$. This is contrary to expectation since the selected parameters led to the highest average accuracy. This may indicate that choosing parameters based on the average accuracy does not always give the best parameters.

4.2.4.11 Comparing the Algorithms Results for the Constant Rate Attacks vs Poisson Process Attacks

The algorithms' results are discussed based on the optimal thresholds at $\alpha = 0.5$ and $\beta = 0.98$, after tuning the parameters and after modifying the variance of the CUSUM based algorithm. For the constant rate attacks at $\alpha = 0.5$ and $\beta = 0.98$ (Tables 4.9, 4.10 and 4.11 vs 4.1, 4.2 and 4.3),

- the three algorithms detected the attacks quicker as compared to the Poisson process attacks.

- the use of the logic OR operator and the CUSUM based algorithm present lower average detection probability whereas the adaptive threshold algorithm has a higher average detection probability as compared to the Poisson process attacks.
- the use of the logic OR operator and the adaptive threshold algorithm present higher average false positive rates whereas the CUSUM based algorithm had a lower average false positive rate as compared to the Poisson process attacks.
- the use of the three algorithms present lower average accuracies as compared to the Poisson process attacks.

For the constant rate attacks after tuning the parameters (Tables 4.12, 4.13 and 4.14 vs 4.4, 4.5 and 4.6),

- the three algorithms detected the attacks quicker as compared to the Poisson process attacks.
- the use of the logic OR operator and the adaptive threshold algorithm presented lower average detection probability whereas the CUSUM based algorithm had a higher average detection probability as compared to the Poisson process attacks.
- the use of the three algorithms present higher average false positive rates as compared to the Poisson process attacks.
- the use of the three algorithms present lower average accuracies as compared to the Poisson process attacks.

For the constant rate attacks after modifying the variance of the CUSUM based algorithm (Tables 4.12, 4.15 and 4.16 vs 4.4, 4.7 and 4.8),

- the CUSUM based algorithm took longer to detect the attacks whereas the adaptive threshold algorithm and the logic OR operator were quicker as compared to the Poisson process attacks.
- the use of the logic OR operator presented higher average detection probability whereas the CUSUM based algorithm had a lower average detection probability as compared to the Poisson process attacks.
- the use of the logic OR operator and the CUSUM based algorithms presented higher average false positive rates as compared to the Poisson process attacks.
- the use of the logic OR operator and the CUSUM based algorithms presented lower average accuracies as compared to the Poisson process attacks.

The results indicate that the overall detection of Poisson process attacks was better than that of the constant rate attacks. When we counted the number of TCP SYN packets in the intervals where the Poisson attacks were added for both the

Poisson process and the constant rate attacks, the number of TCP SYN packets for the Poisson process attacks were higher than those of the constant rate attacks as depicted in Figure 4.6. This means the Poisson attacks were more likely to cross the alarm threshold than the constant rate attacks, hence the better detection.

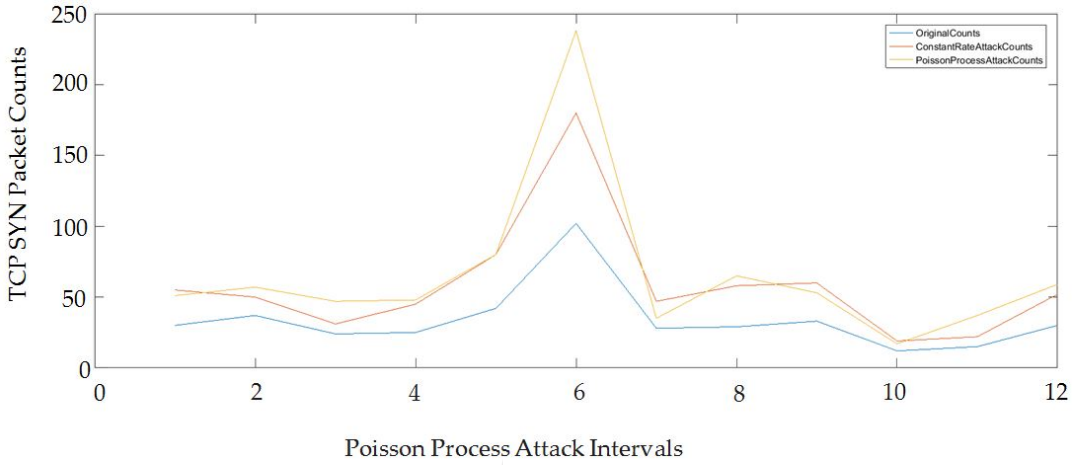


FIGURE 4.6: The number of TCP SYN packets for the Poisson process attacks vs the constant rate attacks.

4.3 Implementation of Learning based Network Intrusion Detection Algorithms

The intrusion detection problem is a classification problem that involves categorising the network traffic into normal network traffic or intrusive network traffic. Learning based algorithms are used to build classification models, therefore they can be utilised in the problem of intrusion detection. The intrinsic characteristic of intrusion analysis is uncertainty. Therefore classification models that are able to analyse data in the presence of uncertainty are indispensable. In this section, a decision tree which is a popular machine learning technique that is used for classification of instances and fuzzy logic which is a learning based powerful tool for reasoning under uncertainty are used to detect Neptune which is a type of a Transmission Control Protocol Synchronized (TCP SYN) flooding attack.

4.3.1 Dataset

In this section, the NSL KDD training and test data were filtered for Neptune (a type of a DoS attack) and normal connections that are referred to as Neptune and normal cases respectively. The resultant training data had 10858 cases with 41215 Neptune cases and 67343 normal cases. The resultant test data had 14368 cases with 4657 Neptune cases and 9711 normal cases. Our interest was in predicting the actual proportion of Neptune cases in the test data where the actual proportion of Neptune

cases in the test data is the number of Neptune cases in the test data divided by the number of cases in the test data.

The NSL KDD dataset has 41 attributes and Kayacik et al. [100] recommended ten attributes as relevant in identifying Neptune. The rule base of a fuzzy logic system tends to be complex quickly if a large number of input and output attributes are used, therefore, four input attributes were selected and used to train and test the fuzzy logic based NIDS. These attributes were the percentage of connections that have "SYN" errors, percentage of connections to the same service, percentage of connections to the different service and count of connections having the same destination host and using the same service. In this research, they are denoted as SynErrorRate, SameSrvRate, DiffSrvRate and DStHostSrvCount respectively. These attributes were also used to train the decision true. The extracts of the training and test data consisting the four attributes are given in Tables 4.17 and 4.18. DstHostSrvCount attribute was not included in the resultant decision tree. This occurs when the accuracy of the decision tree is not improved by the attribute [113]. This means that this attribute is redundant and including a redundant attribute adversely affects the accuracy of a classifier [114]. The fuzzy logic based NIDS was implemented using the attributes that were in the constructed decision tree.

TABLE 4.17: An extract of the training data from the NSL KDD dataset

SynErrorRate	SameSrvRate	DiffSrvRate	DstHostSrvCount	Class Label
0	1	0	25	Normal
0	0.08	0.15	1	Normal
1	0.05	0.07	26	Neptune
0.2	1	0	255	Normal
0	1	0	255	Normal
0	0.16	0.06	19	Neptune
1	0.05	0.06	9	Neptune
1	0.14	0.06	15	Neptune
1	0.09	0.05	23	Neptune
1	0.06	0.06	13	Neptune
0	0.06	0.06	12	Neptune
1	0.02	0.06	13	Neptune
0	1	0	219	Normal

4.3.2 The Fuzzy Logic based Network Intrusion Detection System

In this subsection, the fuzzy logic based NIDS used in detecting Neptune is built and tested. The building phase involves the fuzzification of the input and output attributes into fuzzy membership functions, the generation of the fuzzy rules that are used to describe the desired system output and the determining of the fuzzy inferencing and defuzzification methods.

TABLE 4.18: An extract of the test data from the NSL KDD dataset

SynErrorRate	SameSrvRate	DiffSrvRate	DstHostSrvCount	Class Label
0	0.04	0.06	10	Neptune
0	0.01	0.06	1	Neptune
0	1	0	86	Normal
0	1	0	255	Normal
0	1	0	28	Normal
0	1	0	255	Normal
0	1	0	129	Normal
0	0.02	0.07	2	Neptune
1	1	0	171	Neptune
0	1	0	73	Normal
0	1	0	255	Normal
0	1	0	255	Normal
0	1	0	255	Normal

4.3.2.1 Fuzzification and Membership Functions

In fuzzification, all input and output variable values are fuzzified into fuzzy membership functions, that is, the range of values taken by each input/output variable are divided into fuzzy sets and a fuzzy membership function is determined to assign the degree of membership to a fuzzy set of the input and output variables. In this study, the fuzzy sets are referred to as membership values. To derive the membership values for each of the input attributes, the range of values each attribute takes were observed and the minimum, maximum and average values for each attribute in the normal, attack and mixed (consists of both attack and normal) data of the training data were calculated and are depicted in Tables 4.19, 4.20 and 4.21. For each attribute, we defined three membership values, namely, L (low), M (medium) and H (high). The attributes membership values were chosen based on the maximum, average and minimum values of each attribute.

TABLE 4.19: The minimum, maximum and average values fo the three input attributes for normal only data

Attribute Value	SynErrorRate	SameErrorRate	DiffSrvRate
Minimum	0	0	0
Maximum	1	1	1
Average	0.0134	0.9694	0.0288

TABLE 4.20: The minimum, maximum and average values fo the three input attributes for mixed (normal and attack) data

Attribute Value	SynErrorRate	SameErrorRate	DiffSrvRate
Minimum	0	0	0
Maximum	1	1	1
Average	0.3242	0.6398	0.0454

TABLE 4.21: The minimum, maximum and average values fo the three input attributes for attack (Neptune) only data

Attribute Value	SynErrorRate	SameErrorRate	DiffSrvRate
Minimum	0	0	0
Maximum	1	1	1
Average	0.8319	0.1012	0.0725

The commonly used triangular membership function was utilised to assign the degree of membership to each attribute membership value as shown in Figure 4.7. The output is defined as the percentage of intrusion in the data (% Intrusion). It is also fuzzified into three membership values, namely, L (low), M (medium) and H (high) and the triangular membership function was also used to assign the degree of membership to each membership value of the output attribute.

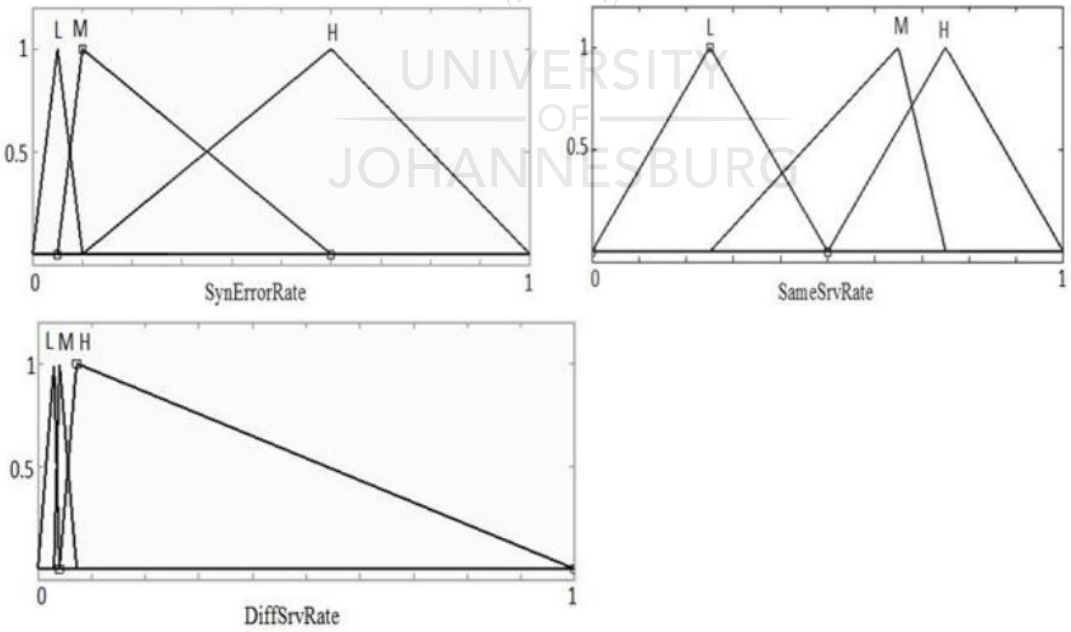


FIGURE 4.7: The membership functions of the fuzzy logic based system

4.3.2.2 Fuzzy Rules Generation

The rules were generated in the form of IF $X = x$ and $Y = y$ THEN $Z = z$ statements. All possible permutations of the membership values of the three attributes were enumerated. Initially we used all twenty seven permutations to create the rules, however, for some permutations, it was difficult to infer the consequent which resulted in poor performance of the system. Therefore, only the permutations that led to an easy way to deduce the consequent of the rules were selected and it resulted in only nine permutations. The permutations were used as the antecedent for each rule. From the training data, the average value for each attribute in the attack, normal and mixed data was observed. It was noticed that the average value decreased in the presence of attacks for some attributes while it increased for some attributes. The consequent of each rule was then based on the behaviour of the average value of each attribute in the absence or presence of an attack. The generated rules are given below.

Rule 1: IF Average of SynErrorRate = L AND Average SameSrvRate = L AND Average of DiffSrvRate = L THEN % Intrusion = L.

Rule 2: IF Average of SynErrorRate = L AND Average SameSrvRate = H AND Average of DiffSrvRate = L THEN % Intrusion = L.

Rule 3: IF Average of SynErrorRate = H AND Average SameSrvRate = L AND Average of DiffSrvRate = H THEN % Intrusion = H.

Rule 4: IF Average of SynErrorRate = H AND Average SameSrvRate = L AND Average of DiffSrvRate = L THEN % Intrusion = M.

Rule 5: IF Average of SynErrorRate = H AND Average SameSrvRate = H AND Average of DiffSrvRate = H THEN % Intrusion = M.

Rule 6: IF Average of SynErrorRate = L AND Average SameSrvRate = L AND Average of DiffSrvRate = H THEN % Intrusion = M.

Rule 7: IF Average of SynErrorRate = M AND Average SameSrvRate = M AND Average of DiffSrvRate = M THEN % Intrusion = M.

Rule 8: IF Average of SynErrorRate = H AND Average SameSrvRate = H AND Average of DiffSrvRate = L THEN % Intrusion = L. Rule 9: IF Average of SynErrorRate = L AND Average SameSrvRate = H AND Average of DiffSrvRate = H THEN % Intrusion = L.

4.3.2.3 Fuzzy Inferencing and Defuzzification

The Mamdani fuzzy inferencing was adopted in this study since it is the commonly utilised method [83]. The fuzzified inputs are combined to obtain the strength

of each rule. The strength of each rule is determined by combining the input membership indices corresponding to the input attribute averages (fuzzified inputs) using the fuzzy "and" operator. The fuzzy "and" operator returns the minimum membership index associated with the fuzzified inputs. Each rule strength is combined with the output membership function to get the membership index of the fuzzy consequent/output of each rule. Some membership indices of the rule consequents will be zero and some will be greater than zero. The rules whose fuzzy consequent membership indices are more than zero become activated. The fuzzy outputs/ consequents of all the activated rules are combined to obtain one fuzzy output distribution. The outputs associated with the activated are usually combined using the fuzzy "or" operator that returns the maximum membership indices associated with the fuzzy outputs.

One of the commonly used defuzzification techniques called centroid is utilised to find the crisp value of the output. This technique returns the centre of the area under the fuzzy output distribution as the crisp value.

4.3.2.4 Prediction with the Fuzzy Logic based System

For each attribute an average was calculated from the test data and the obtained averages for SynErrorRate, SameSrvRate and DiffSrvRate were $0.1077, 0.6904$ and 0.0339 respectively. These averages were tested against the generated rules of the proposed system to determine the strength of each rule. Each rule strength was combined with the output membership function to get the fuzzy consequent/output of each rule. Figure 4.8 presents the fuzzy consequent for each rule. From Figure 4.8 some membership indices of the rule consequents are zero and some are greater than zero.

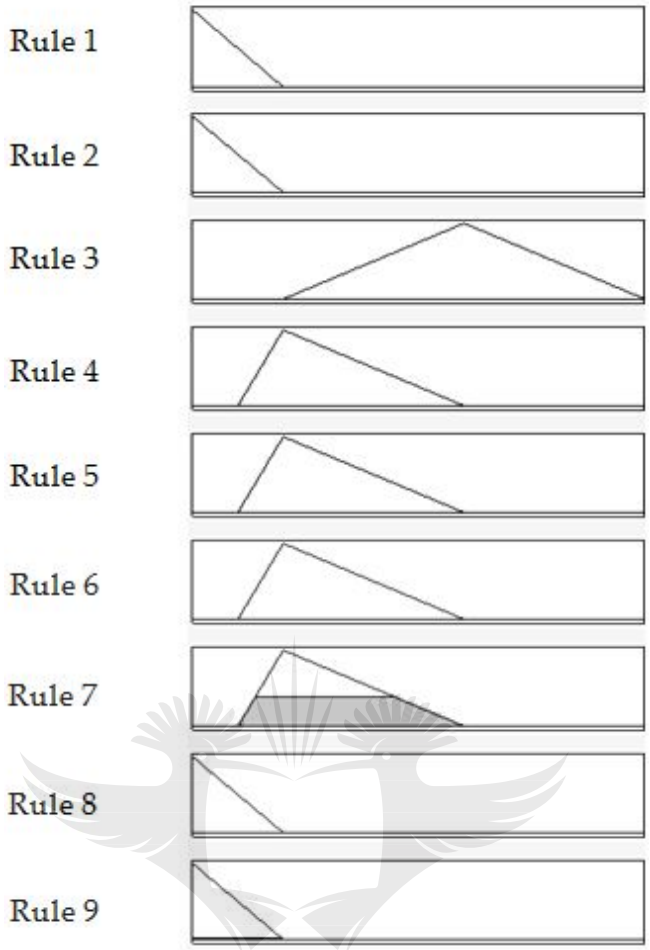


FIGURE 4.8: The fuzzy consequents of each fuzzy rules.

Rule 7 and Rule 9 that are respectively:

Rule 7: IF Average of SynErrorRate = M AND Average SameSrvRate = M AND Average of DiffSrvRate = M THEN % Intrusion = M.

Rule 9: IF Average of SynErrorRate = L AND Average SameSrvRate = H AND Average of DiffSrvRate = H THEN % Intrusion = 1.

have membership indices that are more than zero and are therefore activated. The fuzzy consequents of the two activated rules are combined using the fuzzy "or" operator to obtain the fuzzy output distribution in Figure 4.9. The crisp value of the output attribute was determined using the centroid approach that finds the centre of the obtained fuzzy output distribution and is illustrated by the solid vertical line in Figure 4.9.

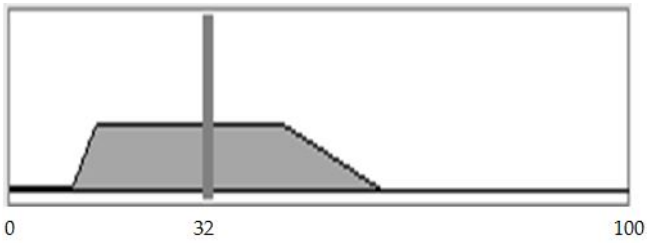


FIGURE 4.9: The crisp value of the output variable, % intrusion.

4.3.3 Decision Tree Construction

The decision tree was built from the training data set and Figure 4.10 depicts the resultant decision tree. We used the constructed decision tree to predict Neptune from the processed test data.

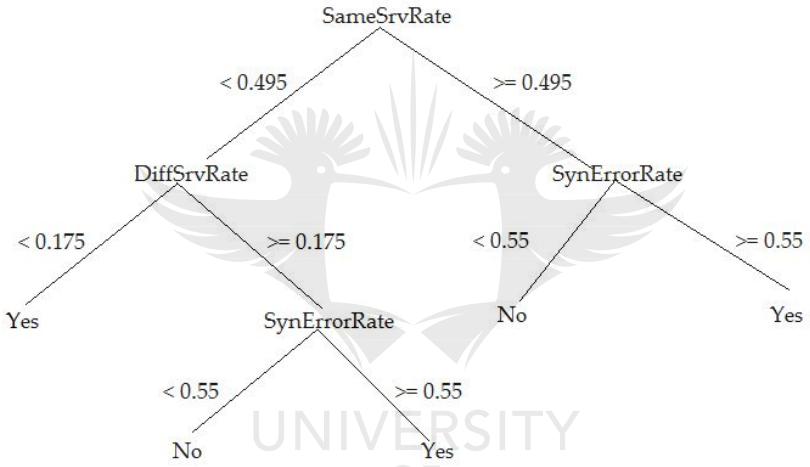


FIGURE 4.10: The constructed decision tree for predicting Neptune.

4.3.4 Results

The actual proportion of Neptune cases (attacks) in the test data is 0.3241. The proportion of Neptune cases predicted by the two algorithms as well as algorithms' accuracies are presented in Table 4.22. The results in Table 4.22 show that the accuracy of the fuzzy system in predicting the proportion of Neptune cases in the test data is less than the decision tree accuracy.

TABLE 4.22: The results of the fuzzy logic based system and the decision tree in terms of predicted attack proportion and accuracy

Performance Metrics	Fuzzy Logic based System	Decision Tree
Predicted Proportion of Attacks	0.3200	0.3223
Accuracy	0.9324	0.9964

The results in Table 4.23 show that the difference between the performances of the fuzzy system and the decision tree, in terms of sensitivity, is negligible and the specificity of the fuzzy system is worse than that of the decision tree. The results in

TABLE 4.23: The results of the fuzzy logic based system and the decision tree in terms of sensitivity and specificity

Performance Metrics	Fuzzy Logic based System	Decision Tree
Sensitivity	0.9873	0.9944
Specificity	0.9060	0.9973

Table 4.24 show that the decision tree has lower false positive and false negative rates as compared to the fuzzy logic based system. The decision tree has performed better than the fuzzy logic based system. One of the challenges of working with fuzzy logic is to set the location of the membership function of the features which is solved by trial and error [115], which is the challenge we also encountered. Specifically, we struggled with setting the boundaries of the fuzzy sets of the membership functions of the features. For different boundaries we obtained different true positive rates and false positive rates, therefore through trial and error we obtained the recorded TPR and FPR which were the best of the other obtained true positive and false positive rates. This means a different researcher could set different fuzzy set boundaries for the same features that may lead to better or worse TPR and FPR than those of a decision tree or our IDS.

TABLE 4.24: The results of the fuzzy logic based system and the decision tree in terms of false positive and false negative rates

Performance Metrics	Fuzzy Logic based System	Decision Tree
False Positive Rate	0.0940	0.0027
False Negative Rate	0.0127	0.0056

4.4 Summary

The main aim of this Chapter was to illustrate the achievable performance of existing network intrusion detection systems for detecting SYN flooding attacks. Firstly two anomaly detection algorithms from the work of [36] for detecting SYN flooding attack were implemented to detect synthetic attacks generated according to a Poisson process and constant rate arrivals. Furthermore, the decisions of the two algorithms were combined at different detection thresholds of the two algorithms using the logic OR operator for both attack types. Detection probability, false positive rate, accuracy and detection delay were utilised as measures of performance. Initially, parameters from the work of [36] were used in the implementation of the two anomaly based algorithms, followed by tuning the parameters of the two anomaly based algorithms and using the parameter combination that leads to the highest accuracy the best parameters were chosen. The

CUSUM based algorithm performance was lower than that of the adaptive threshold algorithm, however, in the work of [36] the CUSUM algorithm outperformed the adaptive threshold algorithm so to improve the performance of the CUSUM based algorithm we modified its variance and also chose the best parameter combination in the presence of the modified variance.

For each set of parameters, where the equal error rate of the two anomaly based algorithms were used to select the parameter (detection threshold) values to be used, the logic OR operator performed better than the two anomaly based algorithms in terms of detection probability and detection delay. However, the logic OR operator had the worst false alarm, as a result, it had the worst accuracy most of the time for both attacks.

For the Poisson process attacks, the tuning of the parameters led to the reduction of the false alarm rate which caused accuracy of the three algorithms to improve or, at least, remained the same. For the constant rate attacks, the tuning of the parameters led to the reduction of the false alarm rate for the adaptive threshold algorithm which resulted in an improvement to its accuracy.

Modifying the variance of the CUSUM based algorithm led to a higher average detection probability, average accuracy and average detection delay. The modification led to a lower average false positive rate for the Poisson attacks along with a lower average false positive rate and a higher average accuracy for the constant rate attacks than those of the adaptive threshold algorithm.

The three algorithms, at most, performed better in detecting the Poisson attacks as compared to the constant rate attacks.

Lastly, two learning based NIDSs fuzzy logic based system and decision tree, for predicting Neptune were implemented. The results also indicate that the accuracy, specificity, false negative and false positive rates of the decision tree are better than those of the fuzzy logic based system in predicting the proportion of Neptune attacks.

Chapter 5

The Ensemble of Classifiers based Network Intrusion Detection System Performance Bounds

This Chapter empirically determines the achievable performance bounds of two ensemble of classifiers based NIDSs. The first NIDS is an AdaBoost based ensemble that has a decision stump as a base learner. The second NIDS is a Bagging based ensemble that has a decision tree as a base learner. The information theoretic measure, information gain and entropy are used to define the bounds.

5.1 Introduction

The ensemble of classifiers and sensor fusion methods that combine predictions/ decisions from a set of IDSs to make the final decision have been implemented successfully in intrusion detection to improve the performance of intrusion detection systems. Thomas and Balakrishna [11], Tian et al. [12] and Gong et al. [20] presented some of the studies that implemented sensor fusion in intrusion detection. Hu et al. [19], Borji [38], Govindarajan [39], Sornsuwit and Jaiyen [40], Prusti [41], Natesan et al. [43] and Kumar and Kumar [44] are some of the works that implemented ensemble methods in intrusion detection.

However, the achievable performance bounds of these systems are not known beforehand. Currently, researchers have to implement the systems before they can determine what their systems performances will be. To gauge the performance of the NIDSs, researchers compare their systems' performances to those of other existing systems. This means a performance bound that can be used as reference point for gauging NIDSs' performances is needed. Even though the newly developed algorithms may outperform the existing algorithms, the lack of the knowledge of the achievable performance bound disadvantages the researcher in the sense that researchers cannot determine how far the performance of their system is

from optimality in order to decide if their NIDSs need improvement or not. The knowledge of the achievable performance bounds of NIDSs will help researchers to predict their NIDSs performances before they even implement them. This achievable performance bound will serve as a performance reference point that can be used by researchers to gauge the performance of their systems and help them determine if there is a need for improvement on the performance of their systems.

This Thesis provides the achievable performance upper bounds of network intrusion detection systems (NIDSs) that use ensembles of classifiers. The ensemble of classifiers methods combine several classifiers in a certain way to produce an optimal classifier whereas in sensor fusion the multiple IDSs are combined in such a way that the produced IDS performs better than the individual IDS not necessarily optimal. The ensemble of classifier methods are chosen in this work due to their ability to obtain optimal performance. The ensemble of classifiers can be built using one base classifiers or different base classifier with their earlier ensemble referred to as a homogeneous ensemble and the latter as a heterogeneous ensemble. Since this Thesis is the first attempt in determining the achievable performance bounds of an NIDSs, the performance bounds of two homogeneous ensembles are determined. As mention in Chapter 3, there are different ways of building the ensemble of classifiers with the ensemble of classifiers built from manipulating the dataset being the popular method, this work considers the ensemble of classifiers built from manipulating the dataset. Two popular methods of building an ensemble of classifiers from manipulating the dataset are boosting and Bagging and are considered in this works since, apart from bringing an optimal classifier, they also bring the aspect of independence/dependency of the base classifiers in the ensemble. In boosting the classifiers are assumed to be dependent since the decision of the current classifier is influenced by the decision of the previous classifier. In Bagging the classifiers are assumed to be independent since they are built from independent bootstrap samples of the original dataset so the decisions made by those classifiers are independent. In boosting the base classifiers are required to be weak base classifiers. In Bagging the base classifiers are required to be unstable.

This work, therefore, provides the achievable performance bounds of two kinds of NIDSs, namely, a NIDS that uses an ensemble of classifiers with dependent base classifiers and a NIDS that uses an ensemble of classifiers with independent base classifiers. The performance upper bounds are defined in terms of an information theoretic measure called information gain and entropy. Specifically, the average information gain associated with the features used in building the ensembles of classifiers and the dataset entropy are used to define the two performance bounds. The information gain and entropy descriptions are provided in Section 5.2. In the NIDS that uses an ensemble of classifiers with dependent base classifiers, the decision stump is used as the weak base classifier and its performance is boosted to optimality using the boosting ensemble method. In the NIDS that uses an

ensemble of classifiers with independent base classifiers the decision tree is used as the unstable base classifier and its performance is optimised using the Bagging ensemble method.

Different proportions of the NSL KDD dataset that were filtered for normal and Neptune connections were used as different datasets in this research in order to observe the behaviour of the performance of the ensemble. The CICIDS2017 was used to test if the bounds hold for a different dataset. The bounds are based on the performance of these ensembles in classifying the Neptune and normal connections. The performance metric used in this study was classification accuracy. This Thesis also provides the false positive and true positive rates associated with these accuracy upper bounds. The rest of this Chapter is organised as follows: Section 5.2 provides a brief description of information theoretic measures and performance metrics used in this Chapter. The empirical studies are presented in Section 5.3. Section 5.4 presents the results and the Chapter is concluded by a summary in Section 5.6.

5.2 Information Theoretic Measure

5.2.1 Entropy

The entropy $H(X)$ of a discrete random variable X measures the uncertainty associated with the values of X and is defined as

$$H(X) = -\sum_{x}(p(x)log(p(x))), \tag{5.1}$$

where $p(x) = P(X = x)$ and P is the probability. The small value of $H(X)$ indicates that X is more regular(less uncertain).

5.2.2 Information Gain

Information gain (IG) measures the amount of information a feature gives about the classification feature. Features that discriminate the data perfectly result in maximum information and irrelevant features give no information. Information gain is calculated using the following equation

$$IG(A) = H(S) - \sum_{i} \frac{S_i}{S} H(S_i). \tag{5.2}$$

Where $H(S)$ is the entropy of the given dataset S in terms of the classification feature and $H(S_i)$ is the entropy of the subset created by partitioning S with respect to feature A [116].

5.3 Empirical Studies

In this section, empirical studies are conducted in order to determine the performance bounds of the two kinds of NIDSs. The two kinds of NIDSs are a NIDS that uses an ensemble of classifiers with dependent base classifiers and a NIDS that uses an ensemble of classifiers with independent base classifiers. The first ensemble uses the decision stump as the base classifier while the second ensemble uses the decision tree as the base classifier. The performance metric considered in this work was classification accuracy. Therefore, the empirical studies are for determining the upper bounds on the accuracies of the two NIDSs. The performance bounds of the two kinds of NIDSs are defined in terms of information gain of the features used to build the two ensembles of classifiers based NIDSs and dataset entropy with respect to the features. This section begins by describing the dataset used in the study, followed by the calculation of the information gain associated with each feature used in the ensembles, which is followed by the definition of the performance metric used in this study and is concluded by describing how the upper bounds on the accuracy of the two ensembles of classifiers based NIDSs were determined.

5.3.1 Dataset

The NSL KDD dataset (both training and test) was filtered for normal and Neptune connections. Initially, 100% of the NSL KDD dataset was used and then we randomly selected 25%, 50% and 75% of the NSL KDD dataset to create different datasets since different datasets were needed in order to observe the consistence of classification accuracy bound (the performance metric in this study). For the CICIDS2017 dataset, the dataset for normal traffic, DoS attacks and Heartbeat attack was filtered for DoS attacks and normal traffic and the DoS attacks were grouped into one category called attacks. Therefore the dataset consisted of attacks and normal samples.

One of the ensemble methods used in this study requires that the base learner to be weak, where weak learners have accuracy of just above random guess on new instances. Therefore, continuous features that resulted to decision stumps with an accuracy of less than 70 percent were selected from the five datasets since weak learners have an accuracy of just above random guess on new instance. For the

- NSL KDD dataset, the resultant features are presented in Table 5.1 for the 100% NSL KDD dataset and Table 5.2 for the 25% NSL KDD dataset, 50% NSL KDD dataset and 75% NSL KDD dataset. However, two continuous features that had decision stumps with an accuracy of less than 70 percent, namely, number of wrong fragments and number of outbound commands in an ftp session, were not included in the study since the two features lead to decision stumps

with only one class at the root node with no branches. For these features, it was not possible to calculate their information gain values.

- CICIDS dataset, the resultant features are presented in Table 5.3.

The different datasets with their selected features were used to determine the upper bounds of the two NIDSs. The CICIDS2017 dataset was used to test if the bounds hold in a new dataset.

TABLE 5.1: The resultant features for the 100 percent NSL KDD dataset.

Dataset	Features
100% NSL KDD	duration (D), hot (H), failed logins (FL), num compromised (NC), urgent (U), num shells (NS), num root (NR),num file creations (NCF), num access files (NAF), srv count (SC), srv diff host rate (SDHR),dst host srv diff host rate (DHSDHR) and dst host count (DHC)

TABLE 5.2: The resultant features for the 75 percent NSL KDD, 50 percent NSL KDD and 25 percent NSL KDD datasets.

Dataset	Features
75% NSL KDD, 50% NSL KDD and 25% NSL KDD	duration (D), hot (H), failed logins (FL), num compromised (NC), urgent (U), num shells (NS), num root (NR),num file creations (NCF), num access files (NAF), srv count (SC), srv diff host rate (SDHR),dst host srv diff host rate (DHSDHR), dst host rerror rate (DHER),dst host srv rerror rate (DHSER), rerror rate (RER) and srv rerror rate (SRER)

TABLE 5.3: The resultant features for the CICIDS2017 datasets.

Dataset	Features
CICIDS2017	forward packet length mean (FPLM), forward header length (FHL), syn.flag.count (SFC), rst.flag.count (RFC), psh.flag.count (PFC), ack.flag.count (AFC), urg.flag.count (UFC), ece.flag.count (EFC), down.up.ratio (DUR), avg.fwd.segment.size (AFSS), fw.header.length.1 (FHL1), lmt_win_bytes_fwd (LWBF), act_data_pkt_fwd (ADPF), min_seg_size_fwd (MSSF), active_std (AS) and idle_std (IS)

5.3.2 Information Gain Calculation

The continuous features were discretised by splitting each feature values into two categories. The splitting value of each decision stump obtained in Subsection 5.3.1 was used to split the values of each feature. After discretising the continuous features, we used (5.2) to calculate the information gain values for the features. We repeated this to all different datasets. Tables 5.4, 5.5, 5.6, 5.7 and 5.8 depict the information gain values for the resultant features of the different datasets.

For each dataset, we needed a way of using the information gain of the features in the ensemble to define the upper bound on the accuracy of the ensemble. It was observed that the information gain values of the features used in the ensemble did not follow any particular trend that could be used to define the bound. We, therefore, decided to find a measure that was able to represent the information gain values for all the features in a single value. Different measures that can represent a set of data in a single value exist, namely, mean (average) which is a measure of central tendency, measures of variation, measures of skewness and measures of kurtosis. In this work, the average information gain amongst the features in the ensemble was used to represent the information gain associated with the features in the ensemble. Where the average information gain amongst the features in the ensemble ($AveIG$) is defined as the sum of the information gain values of the features used in the ensemble divided by the number of those features. Given features $A_1, ..., A_N$ the equation for calculating the average information gain amongst the features in the ensemble is

$$AveIG = \frac{\sum_{i=1}^{N} IG(A_i)}{N}. \tag{5.3}$$

where $IG(A_i)$ is the information gain of the i-th feature and N is the number of features. $AveIG$ decreases as the number of features increases since we added the features that have lower information gain values last in the ensemble.

5.3.3 Performance Metric

The performance measure used in this Chapter is classification accuracy and is calculated using (3.1). The results are reported with respect to the range of values of the dataset entropy and the range of values of the average information gain amongst the features that were used in constructing the two ensembles.

TABLE 5.4: Information gain for the 100 percent NSL KDD dataset resultant features

Features	Information Gain
DHC	0.2665
DHSDHR	0.2458
SDHR	0.1674
SC	0.0606
D	0.0529
H	0.0080
NR	0.0040
NC	0.0024
NAF	0.0023
NCF	0.0016
FL	0.000432
NS	0.00025
U	3.81E-05

TABLE 5.5: Information gain for the 75 percent NSL KDD dataset resultant features

Features	Information Gain
DHSDHR	0.2458
SDHR	0.1674
SC	0.0606
DHSER	0.0548
D	0.0526
H	0.0076
DHER	0.0425
SRER	0.033
RER	0.0326
NR	0.004
NAF	0.0023
NC	0.0023
NCF	0.0016
FL	0.000459
NS	0.00021
U	2.55E-05

5.3.4 Empirical Determination of the Performance Upper Bound for the Two Network Intrusion Detection Systems

5.3.4.1 Decision Stump Ensemble based Network Intrusion Detection System

For the different datasets, the accuracy of the decision stump was boosted using the popular AdaBoost algorithm until it converged onto a particular value and this value is the optimal accuracy of the ensemble. In AdaBoosting, the base classifier is called over several iteration and the accuracy value of the ensemble tends to improve as the number of iterations increases. However, as the number of iterations increases

TABLE 5.6: Information gain for the 50 percent NSL KDD dataset resultant features

Features	Information Gain
DHSDHR	0.2458
SDHR	0.1672
SC	0.0820
DHSER	0.0551
D	0.0523
DHER	0.0428
RER	0.0330
SRER	0.0334
H	0.0076
NR	0.004
NC	0.0022
NAF	0.0022
NCF	0.0015
FL	0.000496
NS	0.00023
U	1.27E-05

TABLE 5.7: Information gain for the 25 percent NSL KDD dataset resultant features

Features	Information Gain
DHSDHR	0.2461
SDHR	0.1656
SC	0.0818
DHSER	0.0544
D	0.0533
DHER	0.0433
SRER	0.0341
RER	0.0338
H	0.0074
NR	0.0038
NC	0.0023
NAF	0.0022
NCF	0.0018
FL	0.000533
NS	0.00023
U	2.54E-05

the accuracy value of the ensemble converges onto a particular value and remains at that value even if more iterations are performed. Figure 5.1 illustrates this. This reached accuracy value is the optimal ensemble accuracy.

Algorithm 9 explains how the features were selected and add to the ensemble in order to train the ensemble, the ensemble was tested and the optimal accuracy of the ensemble was determined.

TABLE 5.8: Information gain for the CICIDS2017 dataset resultant features

Features	Information Gain
FPLM	0.0433
FHL	0.1802
SFC	0.0121
RFC	0.00023
PFC	0.0156
AFC	0.0906
UFC	0.0359
EFC	0.00023
DUR	0.0683
AFSS	0.0433
FHL1	0.1802
LWBF	0.2409
ADPF	0.0376
MSSF	0.0201
AS	0.0282
IS	0.0116

For all the datasets, we noticed that as we added new features to the ensemble the optimal accuracy value tended to increase, therefore we grouped the optimal accuracy values according to the average information gain of the features that resulted in that optimal accuracy. Figure 5.3 illustrate this. Algorithm 10 describes how the results displayed in Figures 5.3 and 5.4 were obtained.

5.3.4.2 Decision Tree Ensemble based Network Intrusion Detection System

For the different datasets, the accuracy of the decision tree was improved using the Bagging algorithm until it converged onto a particular value and this value is the optimal accuracy of the ensemble. In Bagging as more and more bootstrap sample based classifiers are added in the ensemble the accuracy of the ensemble improves until it reaches a particular value and remains at that value even if the number of classifiers (resulting from more bootstrap samples) in the ensemble is increased. This reached accuracy value is the optimal ensemble accuracy.

However, the ensemble accuracy changed as we repeated the experiment due to the random sampling in the bootstrap samples. This means that, depending on the bootstraps samples, different accuracy values will be obtained. To address this variation, we first tested if the Bagging algorithm was stable using the bounded input and bounded out (BIBO) method. To test for the stability of the algorithm we used the training data of the NSL KDD dataset to train and test the algorithm. We set different bounds for the training data (input) and trained a Bagging based ensemble of two trees. The ensemble was tested 1000, 2000, 4000 and 6000 times to determine

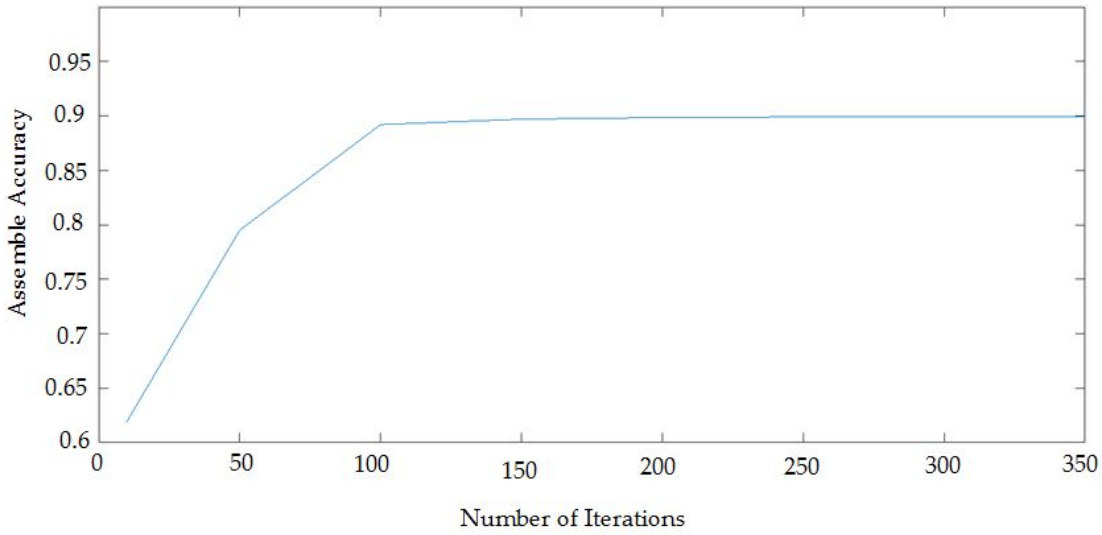


FIGURE 5.1: Assemble accuracy vs the number of iterations.

if the ensemble's accuracy (output) was bounded. Figure 5.2 depicts the outcomes of the stability test. From the stability test results, we concluded that the Bagging algorithm was stable. This means that we expect the outcomes of the algorithm no to vary uncontrollably but to be within a controlled/closed range of values.

Secondly, we ran each ensembles (starting with an ensemble of size 2) 100 times (instead of 10000 times which is roughly 10% of the training data size) since the algorithm was stable and averaged their accuracies instead of using a single accuracy value. The upper bound on the decision tree ensemble accuracy was based on the optimal average accuracy and it was reported with error to accommodate the random behaviour of the individual ensemble accuracies due to the randomness of the bootstraps. The optimal average ensemble accuracy was the average ensemble accuracy that the ensemble converges onto as the ensemble size grows.

The features were added in this ensemble similarly to the decision stump ensemble, see Algorithm 9.

5.4 Results on Determining the Upper Bounds of the two Network Intrusion Detection Systems

5.4.1 Results on the Decision Stump Ensemble based Network Intrusion Detection System

The optimal ensemble accuracies for the different values of the average information gain amongst the features used in the ensemble ($AveIG$) for the four datasets are presented in Figure 5.3.

Algorithm 9 Building and testing of the ensemble

Given:

1. Training dataset D_1.
2. Test dataset D_2.
3. Features $A_1, A_2, ..., A_N$ in both D_1 and D_2 with $IG(A_1) > IG(A_2) > ... > IG(A_N)$ where $IG(A)$ is the information gain of feature A.

To start the ensemble: Select two features from D_1 with the highest information gain.
ensemble sizes = $\text{size}_1, \text{size}_2, ..., \text{size}_m$
While not all features in D_1 are in the ensemble
For ensemble size = smallest to largest
At the current ensemble size:

- Train the ensemble with the selected features from D_1.
- Test the trained ensemble with the corresponding features in D_2.
- Calculate the accuracy of the trained ensemble.
- Record the obtained accuracy.

Move to the next ensemble size.
end
Determine the optimal accuracy (the accuracy value to which the recorded accuracies converge)
Select a new feature from D_1, of the features that have not been used in the ensemble, with highest IG.
Add the selected feature to the features previously selected from D_1.
end

For the four datasets, the curves in Figure 5.3 show that as more features are added to the ensemble, represented by smaller values of $AveIG$, the optimal accuracy of the ensemble increases. The curves for all datasets also indicate that optimal accuracy of the ensemble stopped to increase although more features were added.

The upper bound on the accuracy of the ensemble and the range of values of $AveIG$ for the different datasets are presented in Table 5.9. The results in Table 5.9 show that the differences between the range of values of $AveIG$ and the accuracy upper bounds for the 25% NSL KDD dataset, 50% NSL KDD dataset and 75% NSL KDD dataset are small. The 100% NSL KDD has a slightly higher range of values of $AveIG$ and a slightly lower accuracy upper bound as compared to those of the to the other datasets.

Table 5.10 presents the false positive and true positive rates associated with accuracy

Algorithm 10 The resultant optimal accuracy vs *AveIG*

Given:
dataset D_1 and dataset D_2 and features $A_1, A_2, ..., A_N$ as described in Algorithm 9.
Do,
Select two features from D_1 with the highest information gain.

1. Train the ensemble with the selected features from D_1.
2. Test the trained ensemble with the corresponding features in D_2.
3. Determine the optimal accuracy of the build ensemble and record it.

While not all features in D_1 are in the ensemble

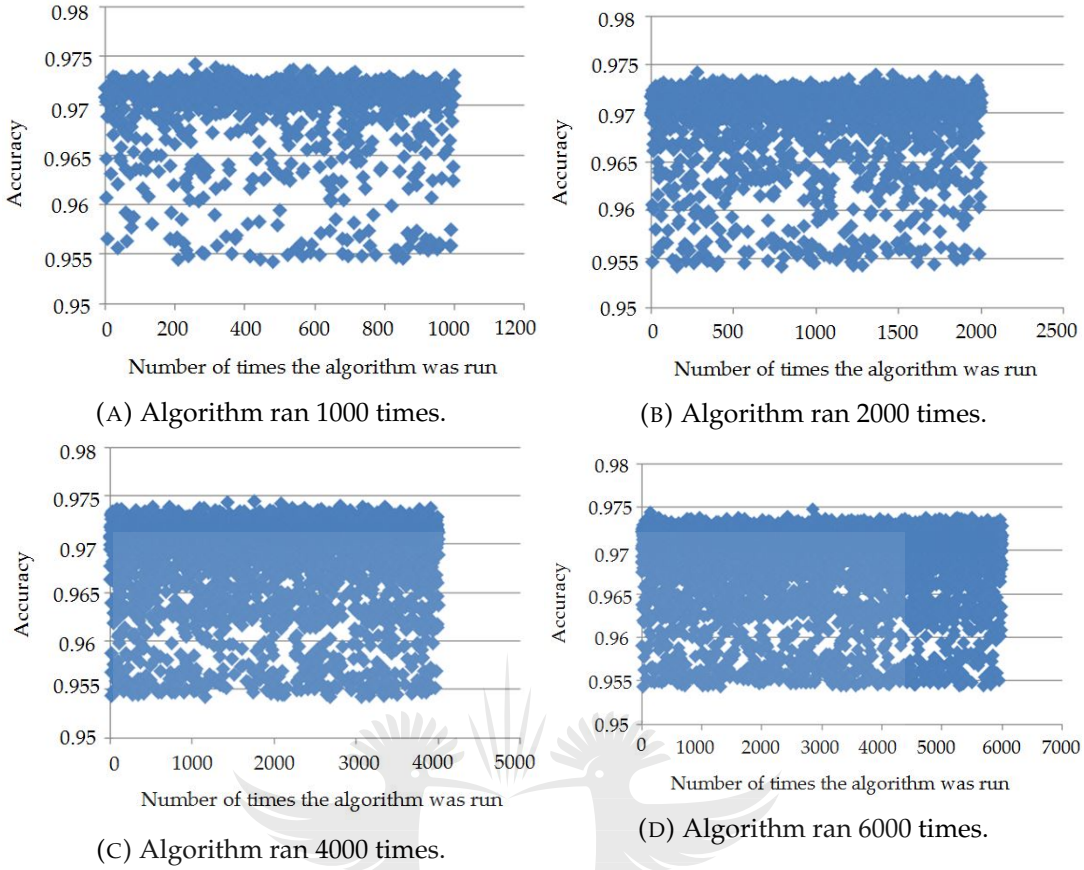
- Add a new feature in the ensemble.
- Repeat steps 1 and 3.
- If the optimal accuracy keeps on changing as more features are added,
  keep adding new features
  else
  use (5.3) to calculate the *AveIG* that led to the same optimal accuracy (referred to as resultant optimal accuracy) as they were successively added to the ensemble before the optimal accuracy changed.
- Record the obtained *AveIG* and the resultant optimal accuracy.

end

TABLE 5.9: Optimal accuracy upper bound against the range of values of AveIG for the different datasets

NSL KDD %	AveIG Range of Values	Accuracy Upper Bound
100	0.062478 to 0.25615	0.8780
75	0.045631 to 0.20695	0.9027
50	0.045615 to 0.2065	0.9003
25	0.045668 to 0.20585	0.9065

upper bounds for the different datasets. The results in Table 5.10 show that there is a small difference on the true positive rates associated with accuracy upper bounds for all datasets NSL KDD datasets. The 100% NSL KDD dataset has slightly higher false positive rate as compared to those of the other datasets.



(A) Algorithm ran 1000 times.

(B) Algorithm ran 2000 times.

(C) Algorithm ran 4000 times.

(D) Algorithm ran 6000 times.

FIGURE 5.2: Accuracy for less than the upper quarter of each feature vs the number of time the Bagging algorithm was run (a) the algorithm was run 1000 times, (b) the algorithm was run 2000 times, (c) the algorithm was run 4000 times and (d) the algorithm was run 6000 times.

TABLE 5.10: False positive and true positive rates associated with accuracy upper bounds for the different datasets

NSL KDD	Accuracy Upper Bound	True Positive Rate	False Positive Rate
100%	0.8780	0.9242	0.1442
75%	0.9027	0.9364	0.1183
50%	0.9003	0.9354	0.1212
25%	0.9065	0.9334	0.1101

From the experimental results in Table 5.9 the upper bound on the accuracy of the assemble was obtained by determining the highest accuracy upper bound, the lower limit of the range of values of *AveIg* was obtained by determining the smallest lower limit of the range of values of *AveIg* and the upper limit of the range of values of *AveIg* was obtained by determining the largest upper limit of the range of values of *AveIg* amongst the four datasets which correspond to 0.9065, 0.045615 and 0.25615 respectively. Therefore we infer that, if the *AveIg* falls between 0.045615 and 0.25615 then the ensemble accuracy will be at most 0.9065. The true positive and false

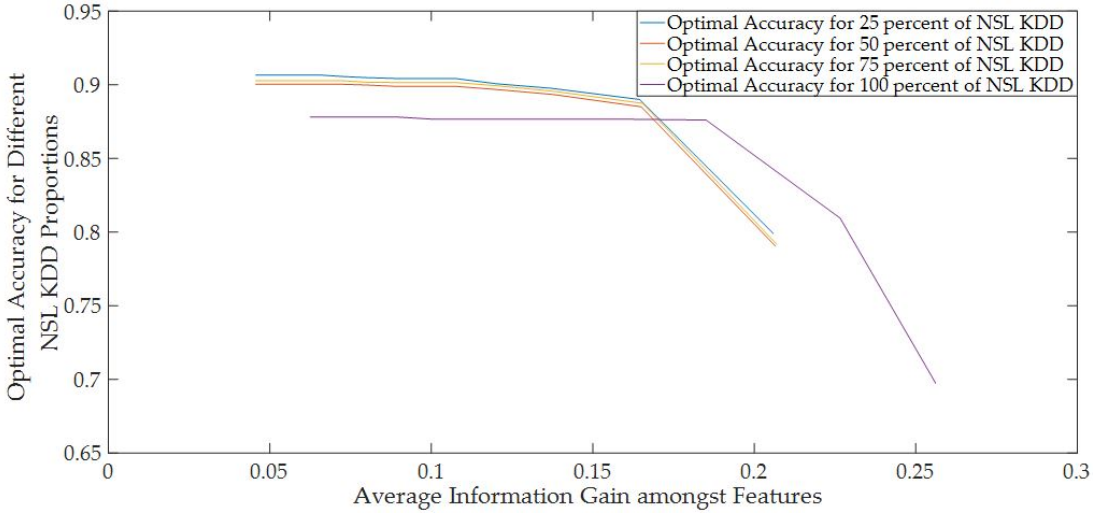


FIGURE 5.3: Optimal accuracy vs average information gain amongst features used in the decision stump based ensemble for the different proportions of the NSL KDD Dataset.

positive rates associated with this bound are 0.9334 and 0.1101.

5.4.2 Results on the Decision Tree Ensemble based Network Intrusion Detection System

The optimal average accuracies of the ensemble for the different *AveIG* values for the four datasets are presented in Figure 5.4. The optimal average ensemble accuracy is the average ensemble accuracy that the ensemble converges to as the ensemble size grows.

For the four datasets, the curves in Figure 5.4 show that as more features are added to the ensemble, represented by smaller values of *AveIG*, the optimal average accuracy of the ensemble tends to increase. However, after increasing to a certain value the ensemble accuracy starts to decrease continuously for the 75% NSL KDD and the 50% NSL KDD datasets while for the 100% NSL KDD and 25% NSL KDD datasets the ensemble accuracy decreases and then stops decreasing (remains at the same value) as more features are added. This value that the optimal average accuracy increases up to before it starts decreasing is used as the optimal average ensemble accuracy for each dataset that is going to be used to determine the upper bound on the accuracy of decision tree based ensemble of classifiers.

The range of values of *AveIG* and optimal average ensemble accuracy for the four datasets are presented in Table 5.11. The results in Table 5.11 show that there is a small difference in the optimal average accuracies for the 75% NSL KDD, 50% NSL KDD and 25% NSL KDD datasets while the optimal average accuracy for the 100% NSL KDD dataset is the lowest .

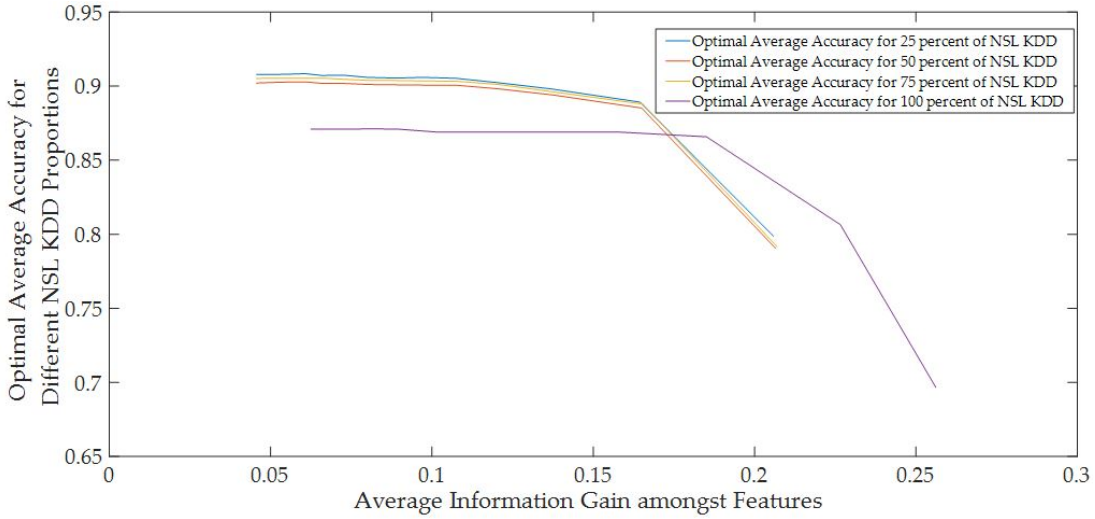


FIGURE 5.4: Optimal average accuracy vs average information gain amongst features used in the decision tree based ensemble for the different proportions of the NSL KDD dataset.

TABLE 5.11: Optimal average accuracy against the range of values of AveIG for the different datasets

NSL KDD %	AveIG Range of Values	Optimal Average Accuracy
100	0.062478 to 0.25615	0.8712
75	0.045631 to 0.20695	0.9054
50	0.045615 to 0.2065	0.9029
25	0.045668 to 0.20585	0.9084

The average false positive and true positive rates associated with optimal average accuracy upper bounds for the different datasets are presented in Table 5.12. The results in Table 5.12 show that there is a small difference in the average true positive rates associated with optimal average accuracy upper bounds for the 70% NSL KDD, 50% NSL KDD and 25% NSL KDD datasets. The 100% NSL KDD dataset has slightly higher average false positive and true positive rates as compared to those of the other datasets.

TABLE 5.12: Average false positive and true positive rates associated with the optimal average accuracy for the different datasets

NSL KDD	Optimal Average Accuracy	Average TPR	Average FPR
100%	0.8712	0.9792	0.1806
75%	0.9054	0.9380	0.1148
50%	0.9029	0.9390	0.1193
25%	0.9084	0.9344	0.1076

From the experimental results in Table 5.11 the optimal average ensemble accuracy to be used in determining the upper bound in the accuracy of the ensemble is obtained by determining the highest optimal average ensemble accuracy amongst

the four datasets which corresponds to 0.9084. In Subsubsection 5.3.4.2 we had said the upper bound on the ensemble would be reported with an error. To determine the error, the highest accuracy (irrespective of the number of features used in the ensemble) achieved by the ensemble using the 25% dataset (dataset corresponds to the highest optimal average accuracy) and its corresponding minimum accuracy were determined and subtracted from the optimal average ensemble accuracy. The maximum difference from the optimal average accuracy was used as the error. There were two records that led to the same highest accuracy with different minimum accuracies. Therefore we averaged the accuracies of the two records to get the minimum and the maximum accuracies of the ensemble. The minimum ensemble accuracy is 0.8976. The maximum ensemble accuracy is 0.9098. The maximum difference from the optimal average accuracy is 0.01085. Therefore, we propose that the upper bound of the ensemble accuracy falls within the optimal average accuracy and the maximum difference from the optimal average ensemble accuracy, which is, 0.9084 ± 0.01085. The lower and the upper limits of the range of values of *AveIG* are obtained similarly to the decision stump based ensemble. Therefore we infer that, if the *AveIG* falls between 0.045615 and 0.25615 then the ensemble accuracy will at best be 0.9193 .

5.5 Results on Testing if the Obtained Bounds Hold

Table 5.13 presents the optimal accuracies for the two NIDSs obtained using the CICIDS2017 dataset. As can be seen from the results, the optimal accuracies are higher than the proposed bounds. From the definition of the bounds, it is expected that the optimal accuracies of the two NIDSs would be within the two bounds despite the dataset as long as the range of the values of the average information gain amongst the features used in the two NIDSs is between 0.045615 and 0.25615. Indeed, the range of the values of the average information gain amongst the features used in the ensemble was within the given range. The cause for the departure from expected was investigated. It was discovered that features with the same/similar information gain coming from different datasets may have different feature average entropy(average entropy resulting from partitioning the dataset with respect to a specific feature). This means these features have different distributions with respect to the target. The differences in feature average entropies of the features with the same information gain are due to the differences amongst the different dataset entropies. As a result, a dataset with low dataset entropy will lead to lower feature average entropy as compared to a dataset with high dataset entropy, Table 5.14 depicts this. This means features with low average entropy are better features and will result in better accuracy. Based on these findings, the data entropy was added as one of the conditions for defining the performance bounds.

TABLE 5.13: Optimal accuracy against the range of values of AveIG for the CICIDS2017 dataset

Dataset Entropy	AveIG Range of Values	NIDS	Optimal Accuracy
0.9462	0.063022 to 0.21055	Decision Stump	0.9807
		Decision Tree	0.9943

TABLE 5.14: Effect of dataset entropy on the feature average entropy for features with same/similar information gain

Dataset	Dataset Entropy	Feature	Feature Average Entropy	Information Gain
NSL KDD 100 %	0.9578	DHSDHR	0.7120	0.2458
		SDHR	0.7904	0.1674
NSL KDD 75 %	0.9586	DHSDHR	0.7131	0.2456
		SDHR	0.7986	0.1683
NSL KDD 50 %	0.9583	DHSDHR	0.7125	0.2458
		SDHR	0.7911	0.1672
		NS	0.9581	0.00023
NSL KDD 25 %	0.9578	DHSDHR	0.7117	0.2461
		SDHR	0.7922	0.1656
		NS	0.9575	0.00023
CICIDS2017	0.9462	LWBF	0.7052	0.2409
		FHL	0.7052	0.1802
		EFC	0.9459	0.00023

Based on the dataset entropies of the different NSL KDD dataset, we deduce that the obtained bounds that are defined in terms of the average information gain amongst features hold if the dataset entropy falls between 0.9578 and 0.9586. Where the lower limit on dataset entropy was obtained by determining the lowest dataset entropy and the upper limit on the dataset entropy was obtained by determining the highest dataset entropy (of the 25%, 50%, 75% and 100% NSL KDD datasets) which led to 0.9578 and 0.9586. Outside this range of values of the dataset entropy, the bounds may not hold. Therefore we infer that, if the dataset entropy falls between 0.9578 and 0.9586 and the $AveIg$ falls between 0.045615 and 0.25615 then the ensemble accuracy will be at most 0.9065 for the decision stump based NIDS and at the best be 0.9193 for the decision tree based NIDS.

5.6 Summary

The main aim of this Chapter was to determine the upper bounds on the accuracy of an AdaBoost based NIDS that utilises a decision stump as a weak learner and a Bagging based NIDS that utilises a decision tree as a base learner. From using the different proportions of the NSL KDD dataset that were filtered for normal and

Neptune connections, the continuous features that resulted to decision stumps with accuracy of less than 70% and splitting these features based on the split point of these decision stumps to calculate the information gain values of these features resulted in an accuracy of at most 0.9065 for the AdaBoost based NIDS and 0.9193 for the Bagging based NIDS. These bounds are defined in terms of the range of the values of the dataset entropy and the range of values of the average information gain amongst the features used in these ensembles ($AveIG$). The bounds only hold if continuous features that lead to decision stumps with accuracies of less than 70% are used.

It was observed that the range of values of $AveIG$ was dependent on how the splitting point of the continuous feature is selected and the way the features are added to the ensemble. In this research the splitting point of the continuous features was based on the split point of the decision stump used to select the features, as described under Subsection 5.3.2. Two features that had the highest information gain values were used to start the ensemble. The average information gain value of the two features was important since it provided the upper bound of the range of values of $AveIG$. Therefore, this research only presents the range of values of $AveIG$ that are obtained according to Subsection 5.3.2 to define the performance bounds of the two ensembles. If a different way of splitting the continuous features in order to obtain their information gain values is used or if features are added to the ensemble differently then $AveIG$ may be different. It has also been observed that the bounds are dependent on the dataset entropy.

From the empirical studies conducted in this Chapter, we deduce that the accuracy upper bounds of the ensembles vary with the range of values of the average information gain amongst features used in the ensemble and may vary with different dataset entropy and also the information gain values of the continuous features depend on how the features are discretised.

Chapter 6

Differentially Private Transmission Control Protocol Synchronize Packet Counts

This Chapter presents the use of differential privacy as a means of providing privacy to network trace. The application of differentially privacy to the network is done similarly to the work of [117]. Specifically, in this study the number of Transmission Control Protocol Synchronise packets associated with HTTP requests made to a web server(s) by employees of an organisation on an eight hour working day are monitored with differential privacy. These packet counts are monitored over every 10 second interval of an eight hour working day. The differential privacy randomisation mechanism called the Laplace mechanism is utilised. Laplace mechanism adds noise to the aggregated statistics of the data (the number of TCP SYN packets, also referred to as TCP SYN counts, in this study). Releasing a series of aggregates with differential privacy tend to lead to high perturbation error more especially if the data values are aggregated over a long period [117]. Therefore to improve the accuracy (the closeness to the original aggregates) of the released aggregates, the added noise is reduced (filtered) using the filtering component of [25]. The research utility of the released aggregates is tested using two utility metrics and by comparing the performance of two anomaly based intrusion detection algorithms on the original aggregates and the released aggregates. The utility measures are utilised to establish if the inferences made using the released aggregates are close to those reached using the original aggregates.

The rest of this Chapter is organised as follows, the methodology and definition of differential privacy and its associated theorems and concepts are presented in Sections 6.1 and 6.2. Section 6.3 presents the problem statement. Section 6.4 provides the application of differential privacy to address the problem statement. The experimental work is carried in Section 6.5. The results are reported in Section 6.6. The Chapter concludes with a brief summary and discussion in Section 6.7.

6.1 Methodology

The methodology framework of the experimentation for this Chapter is outlined in this section. The second ultimate objective of this Thesis is to release differentially private Transmission Control Protocol (TCP) Synchronise (SYN) packet counts. Figure 6.1 presents the general experimental approach that will be followed in creating differentially private TCP SYN Packet Counts. The first step is to extract the relevant network trace from the network traffic. Followed by determining the aggregate statistic of interest per time stamp from the extracted trace. Differential privacy technique is applied on the aggregate statistics. This technique involves adding noise to the aggregate statistics of the original trace and this may cause the statistics to be of no useful value. The accuracy of the privatised aggregate statistics is improved by filtering some of the noise from the privatised aggregate statistics. Kalman filter, has been successfully used in literature to filter such noise. To measure the utility of the noise filtered privatised aggregate statistics, one can use general utility measures like average relative error. Detection algorithms can also be implemented on the original and noise filtered privatised aggregate statistics and determine if what is inferred using the original data is close to what is inferred using the privatised data.

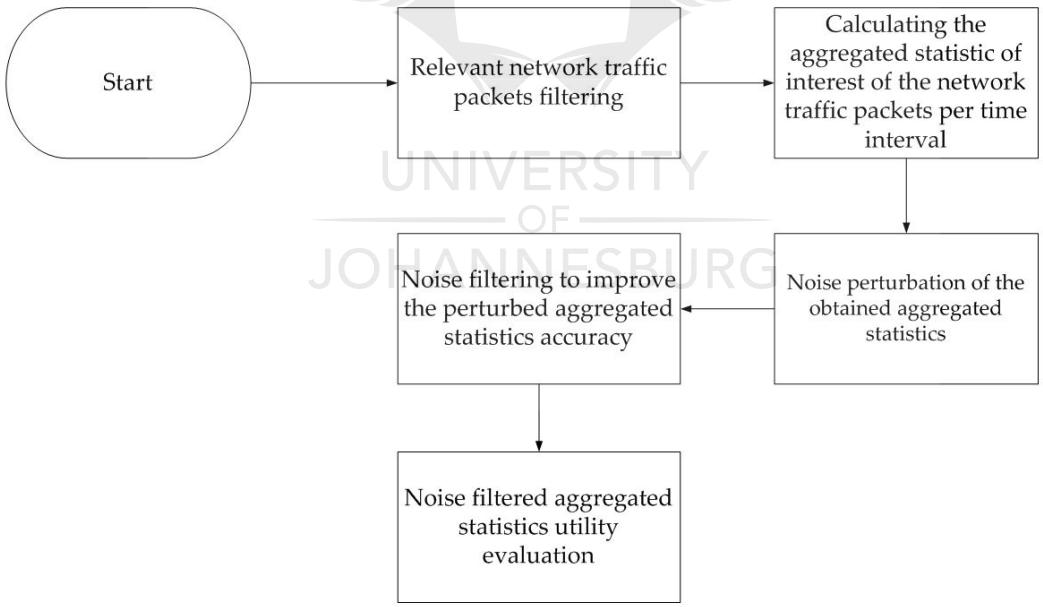


FIGURE 6.1: General experimental process that will be used to create differentially private TCP SYN packet counts.

6.2 Differential Privacy

In this work, we aim to provide differential private TCP SYN packet counts. A mechanism is differentially private if its outcome is not significantly affected by

the removal or addition of any record. Therefore at the release of the outcome, an adversary learns almost the same information about any individual record, regardless of its presence or absence in the original database.

Definition 1, (ε-differential privacy [118]). A privacy mechanism A satisfies ε-differential privacy if for any dataset D_1 and D_2 differing on at most one record, and for any possible anonymised dataset $D \in$ Range(A),

$$\Pr[A(D_1) = D] \leq e^{\varepsilon} \Pr[A(D_2) = D.] \tag{6.1}$$

where the probability is taken over the randomness of A.

The privacy parameter ε also called the privacy budget [62], specifies the degree of privacy offered. Intuitively, a lower value of ε implies stronger privacy guarantee and a larger perturbation noise, and a higher value of ε implies a weaker guarantee while possibly achieving higher accuracy. Two databases D_1 and D_2 that differ on at most one record are called neighbouring databases. In our problem definition, a database "record" represents a new connection request to the web server, i.e. the record is associated with the sending of the TCP SYN packet to the web server by the client (web browsing employee) and therefore our work is designed to protect the presence or absence of every web browsing employee.

Laplace Mechanism. Dwork et al. [119] show that ε-differential privacy can be achieved by adding independent and identically distributed noise to query result $q(D)$:

$$q(D) = q(D) + (N_1, ..., N_m). \tag{6.2}$$

$$N_i = Lap(0, \frac{GS(q)}{\varepsilon}) for i = 1, ..., m. \tag{6.3}$$

where m represents the dimension of $q(D)$. The magnitude of N_i conforms to a Laplace distribution with 0 mean and $GS(q)/\varepsilon$ scale, where $GS(q)$ represents the global sensitivity [119] of the query q.

Global sensitivity. The global sensitivity [119] is the maximum L1 distance between the results of q from any two neighbouring databases D_1 and D_2. Formally, it is defined as follows:

$$GS(q) = \max ||q(D_1) - q(D_2)||. \tag{6.4}$$

Composition. The composition properties of differential privacy provide privacy guarantees for a sequence of computations as outlined in theorem 1 below.

Theorem 1. [Sequential composition [62]].
Let each A_i provide ε_i-differential privacy. A sequence of $A_i(D)$ over the dataset D provides $\sum_i \varepsilon_i$-differential privacy.

From theorem 1, a baseline solution to sharing differentially private time series can be deduced, specifically, a Laplace perturbation is applied at every time series time stamp to guarantee (ε/T)-differential privacy, where T is the length of the entire series [25].

6.3 Problem Statement

This section formally defines the problem of monitoring, using differential privacy, the new connections to the web server(s) initiated by employees of an organisation that are browsing the web in a given working day (eight hours). Specifically, the number of TCP SYN packets sent to the webserver during each 10s interval of a given working day resulting from the new connection request to the web server(s) by employees of an organisation that are browsing the web are released using differential privacy to protect the identity of web browsing employees from being inferred by an adversary from the original number of TCP SYN packets using possible background knowledge about the employees' web browsing patterns. That is, if the adversary knows the surfing behaviours of employees in an organisation releasing original HTTP associated TCP SYN packet counts can result in an adversary identifying the presence or absence of at least one employee in the organisation's database of HTTP associated TCP SYN packets. For an example, if the adversary knows that employee A surfs the net noticeably more (more HTTP associated TCP SYN packets generated for this employee) than the other employees and this employee surfs the net at a particular time interval during the day then the presence or the absence of that employee can be determined by the adversary since if employee A is present in the database the TCP SYN packet counts in that period will be noticeably higher than the TCP SYN packet counts in that period in a database that has the same records as the first database except that employee A has been removed. Therefore that noticeable difference in the TCP SYN counts in that period between the two databases has to be masked and differential privacy is capable of doing so. Furthermore, according to Yurcik et al. [120] TCP flags can be

used to finger print different operating systems. Therefore releasing raw TCP SYN packets can expose the different operating systems of the machines in use.

The Hypertext Transfer Protocol (HTTP) is the application layer protocol that defines how web clients (browsers) request web pages from the web servers and how these web pages are transferred by the servers to the clients. The HTTP version, HTTP/1.0 and HTTP/1.1 use TCP as their underlying transport layer protocol. The HTTP client first initiates a TCP connection with the server. Specifically, when a client request a TCP connection to the server a three way handshake process is carried out. Firstly the client sends a request, which is a TCP SYN packet, to the server to establish a connection. The server replies with a TCP SYN/ACK packet. Finally, the client sends a TCP ACK packet to the server and the data transfer is started.

In this work, the TCP SYN packets that initiate new TCP connections between HTTP clients (web browsing employees) and the web server(s) are monitored with differential privacy. Specifically, the number of TCP SYN packets sent to the web server(s) during each 10 second (s) interval of a given working day resulting from the new connection request to the web server(s) by employees of an organization that are browsing the web is released using differential privacy. The availability of such aggregated TCP SYN packet counts will assist the intrusion detection researchers in training their intrusion detection system in order to be able to detect attacks such as TCP SYN flooding attack. The goal of this work is to release the number of TCP SYN packets sent during each 10s interval of a given working day without disclosing the presence or absence of a particular web browsing employee.

Formally the problem statement is stated below as Private TCP SYN packet counts monitoring: Let x_t denote the number of TCP SYN packets sent to the web server(s) at time interval t, with $1 \leq t \leq T$ where T is the total number of time intervals in the monitoring period under the assumption that the time interval at which the packet was sent is the same as the time interval at which it arrives at the web server. For every time interval t, a private count s_t is to be released such that the released series $\{s_t, t = 1, \ldots, T\}$ is ε-differential private.

Furthermore, similarly to [74], we decided to have a limit on the number of webpage requests initiated by an individual employee to the web server(s) in the 8 hours, hence we set a limit on the number TCP SYN packets sent to the web server(s) by an individual employee on a given 8 hour working day, since

1. An employee should not be browsing the web the whole 8 hours (except it is their job description, in which this work excludes those types of employees or organisations or cases).

2. Any web browser can only browse a limited number of web pages in a given 8 hours.

3. From a privacy point of view, if an employee requests an unlimited number of web pages in the 8 hours then large amount of noise will be required in order to account for such influence on the aggregate. The limit to the TCP SYN packets sent by an individual employee to the web server(s) on a given eight hour working day is denoted by C_{max} and we assume $C_{max} < T$.

6.4 Differentially Private TCP SYN Packet Counts

In this section, the application of differential privacy to the TCP SYN packet counts is outlined.

6.4.1 Privacy Mechanism

The Laplace Mechanism is suitable for numerical queries [121] and is adopted in this work as the privacy mechanism since we are monitoring a numerical aggregate statistic. Algorithm 11 illustrates the perturbation of the original TCP SYN counts by adding the Laplace noise to each original TCP SYN count.

Algorithm 11 Laplace noise perturbation of the original TCP SYN count

Input: Original TCP SYN count at time t, x_t.
Output: Laplace perturbed TCP SYN counts vector, Z_t.
For t = 1 to T

1. Create a Laplace measurement at time t, ν_t from the Laplace distribution as follows: $\nu_t = \text{Laplace}(\text{GS(q)}/\epsilon_t)$.
2. Perturb x_t by ν_t to obtain a Laplace perturbed TCP SYN count, z_t which is $z_t = x_t + \nu_t$.
3. Store z_t in vector Z_t as follows: $Z_t = z_t$.
4. $t = t + 1$.

end

6.4.2 Global Sensitivity

In this section the global sensitivity for monitoring the TCP SYN packet counts per 10s interval in a given eight hour working day is analysed.

Let D be the database that consists of employees' HTTP requests to the web server in a given 8 hour working day, $q(D) = \{x_1, \ldots, x_T\}$ be the sequence of outputs from the count queries, where x_t denotes the number of TCP SYN packets sent during t-th 10s interval and T be the length of the series (number of 10s intervals in an 8 hour working day). To determine the global sensitivity $GS(q)$, we studied the HTTP related TCP SYN packets in the DARPA 1999 dataset and noticed that an individual

can request more than one web page in a given time interval t and can appear in more than one time interval. Meaning that more than one TCP SYN packet can originate from the same source in a given time interval t. The effect of this is that the removal or addition of an individual to database D would change the output by at least 1. As we have observed also that the individual can appear in more than one time interval, the global sensitivity of the count query will be affected since global sensitivity defines the maximum contribution of an individual to the function output [117]. From the DARPA 1999 dataset, we found $C_{max} = 712$, where C_{max} value is the maximum HTTP related TCP SYN packets originating from the same source over the eight hours. We, therefore, set $GS(q) = 712$ since this is the highest maximum contribution by an individual in D.

6.4.3 Filtering

As we have mentioned in the introduction, the direct application of the Laplace mechanism to the original aggregates may lead to high perturbation error and leaving the released aggregates to be of no useful value, we adopted the filtering component of [25] in order to improve the accuracy of the released aggregates. Their filtering component utilizes time series modelling and estimation algorithm. In their [25] context, filtering refers to the derivation of the posterior estimates of the original time series from the noisy measurements with the hope of removing background noise from the signal. They estimated the original time series from the noisy measurements using a Kalman filter [122] based estimation algorithm and used a state space model to describe the underlying dynamics of a time series as well as how an observation is derived from a hidden state [25]. In this work, we modelled the time series and noisy measurements and estimated the original series from the noisy estimates to obtain the posterior estimates referred to as Kalman count estimates as follows:

Time series modelling. For the TCP SYN packet count series i.e. $\{x_t, t = 1, \ldots, T\}$, we defined the following models;

process model:

$$x_t = x_{t-1} + \omega_t. \tag{6.5}$$

$$\omega_t \sim N(0, Q). \tag{6.6}$$

where ω_t denotes the process noise at time interval t ,which is assumed to be a white Gaussian noise with variance Q.
Similarly, the measurement model for the noisy observations that are obtained from

the Laplace perturbation mechanism is:

$$z_t = x_t + \nu_t. \tag{6.7}$$

$$\nu_t \sim Laplace(GS(q)\,/\varepsilon). \tag{6.8}$$

where ν_t is the measurement noise at time interval t. Fan and Xiong [25] have established that the posterior distribution cannot be analytically determined if the distribution of the measurement noise is not Gaussian and reported that it is sufficient to approximate the distribution of the measurement noise to a Gaussian distribution. Thus, the following Gaussian distribution was proposed:

$$\nu_t \sim N(0, R), \text{with } R \propto (GS(q))^2/\varepsilon^2. \tag{6.9}$$

In this work, we adopted the same approximation in (6.9).

Estimation algorithm. We adopted the estimation algorithm of [74] which is based on the Kalman filter and approximated Laplace noise with Gaussian noise as suggested by [25]. Kalman filter [122] is a recursive method that provides an efficient means to estimate the state of a linear Gaussian process, by minimizing the variance of the posterior error. It consists of two steps, namely, prediction and correction steps. In the prediction step the state is predicted with the dynamic model. In the correction step the state is corrected with the observation model such that the error covariance of the estimator is minimised. The prediction and correction algorithms adopted in this work can be found in [74].

Privacy guarantee. The estimation algorithm provides ε-differential privacy since by definition of Laplace mechanism and sensitivity analysis in Section 6.2, the Laplace perturbed values $\{z_t, t = 1, \ldots, T\}$ satisfy ε-differential privacy and similarly to [74], neither the prediction nor correction interacts with the raw data so there is no extra privacy leakage incurred by those two procedures.

6.5 Experimental Work

6.5.1 Dataset

Attack free data taken on a Monday of the first week of the DARPA 1999 was utilised in this Chapter. TCP SYN packets associated with HTTP requests to seven web servers were collected between 08:00 to 16:00, i.e., TCP SYN packets collected over

8 hours. The number of TCP SYN packets in 10 second intervals were determined. The attack free data was selected for two reasons:

1. the released differentially private TCP counts can be used to train anomaly based IDS and anomaly based IDS are trained on attack free data.

2. the usability of the differential private data using an anomaly detection algorithm is evaluated in terms of false positive rates which are easier to determine.

Seven servers were used in order to limit the number of times an individual (web browsing employee) appears in the dataset so that the restrictions set in Section 6.3 for individuals browsing the net in a given eight hour working day are met. The TCP SYN packets were collected over eight hours since this study monitors the TCP SYN packets associated with HTTP requests during an eight hour working day of an organisation. The number of TCP SYN packets in 10 second interval is an aggregated statistic of interest since it is used by some of the anomaly detection algorithms.

6.5.2 Experimental Setup

The implementation of algorithms was carried out using Matlab R2016a. The Laplace perturbation of the TCP SYN packets in 10 second intervals were performed in R Studio.

The number of TCP SYN packets in 10 second intervals, also referred to as TCP SYN counts or original counts or just counts, were determined and Figure 6.2 depicts the original counts for the first 500 10s intervals of packet arrivals.

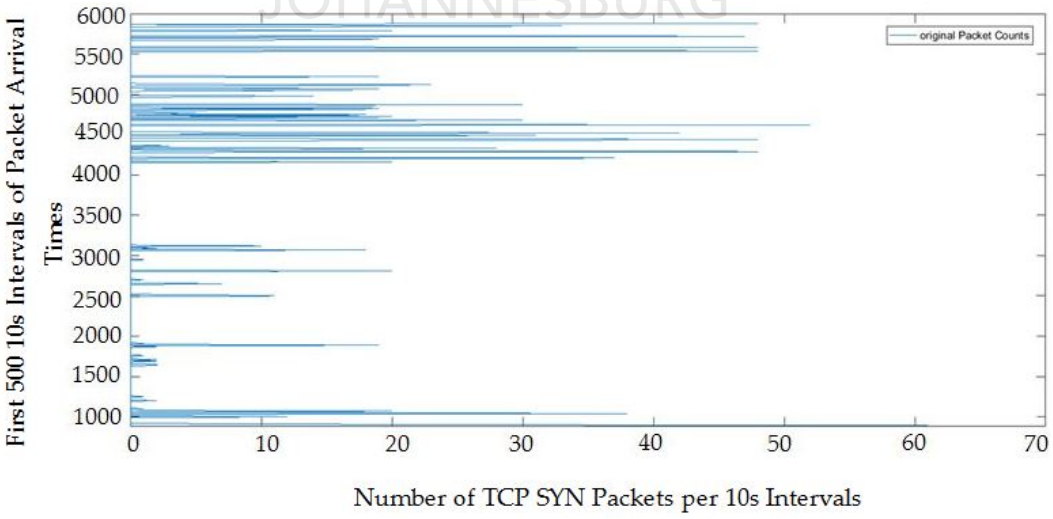


FIGURE 6.2: Original packet counts.

The Laplace noise was added to each count in each interval at the interval privacy budget of, $\varepsilon_i = 0.01$ (i.e. for each 10s interval a Laplace mechanism that provides ε_i-differential privacy was used) and a global sensitivity of $GS(q) = 712$. The interval budget of $\varepsilon_i = 0.01$ was chosen since it provides the lowest overall privacy budget (that can be obtained by using theorem 1) of the recommended privacy budgets (0.01 and 0.1) [123]. Figure 6.3 illustrates the Laplace perturbed counts for the first 500 10s intervals of packet arrivals.

As mentioned at the beginning of the Chapter that direct application of Laplace mechanism by adding the perturbation noise to the aggregated statistics at each time interval can result in a high overall perturbation error causing the released aggregates to be of no useful value especially when T is large [117]. To address this drawback Fan and Xiong [25] have proposed a filtering component that models the series using a state space model and estimates the original data from the noisy data using Kalman filter where the resulting estimates are released in the place of the noisy perturbed data. For the filtering component, the process noise of $Q = 10000$ which was empirically determined as the value that yields better estimates of the original TCP SYN packet counts given the interval privacy budget and the measurement noise of $R = (GS(q))^2/\varepsilon_i^2$ were used.

Fan and Xiong [25] also suggested the Laplace noise to be approximated by the Gaussian distribution for the Kalman filter to work. Therefore, the Kalman filter was used to estimate the original counts from the Gaussian perturbed counts (estimates of the Laplace perturbed counts as suggested by [25]). These estimates are the ones that are released instead of the noisy counts resulting from Laplace perturbation. Figure 6.4 presents the original counts and the estimates of the original counts, referred to as Kalman count estimates for the first 500 10s intervals of packet arrivals.

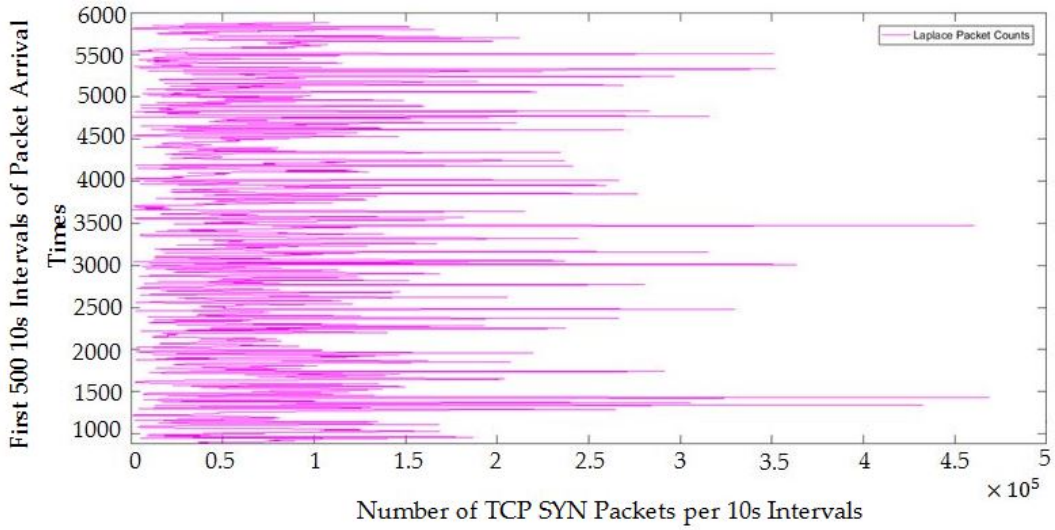


FIGURE 6.3: Laplace perturbed packet counts.

6.5.3 Kalman Count Estimates Utility Evaluation

To measure the quality of released Kalman count estimates $\{s_t, t = 1, \ldots, T\}$:

1. We used two utility metrics called average relative error (E) and utility loss (U)
2. The performances of the cumulative sum (CUSUM) and adaptive threshold algorithms on the original aggregates were compared to their performances on the released aggregates and this is illustrated in Algorithm 12

Algorithm 12 Utility evaluation of Kalman count estimates using the CUSUM and adaptive threshold algorithms

Input: Attack free original counts and Kalman count estmates.
Output: False positive rates (FPRs) for the CUSUM and adaptive threshold algorithms.
Given: Attack free original counts, Kalman count estimates and two intrusion detection algorithms: CUSUM and adaptive threshold algorithms.

Do
1. Run the CUSUM algorithm on the attack free original counts.
2. Determine the FPRs associated with the CUSUM algorithm since the algorithm did detection on attack free counts.
3. Repeat steps 1 and 2 on the Kalman count estimates.
4. Compare the CUSUM's FPRs on the original counts with CUSUM's FPRs on the Kalman count estimates.
5. Repeat steps 1-4 with the adaptive threshold algorithm.

end

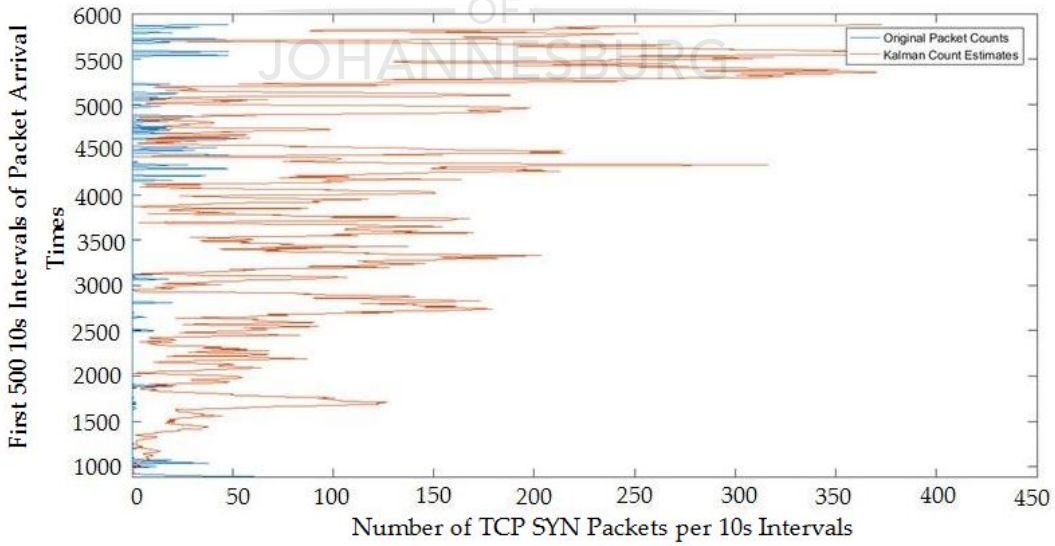


FIGURE 6.4: Original packet counts vs Kalman count estimates.

6.5.3.1 Average Relative Error

Average relative error, E, is a widely used metric to evaluate the accuracy of the data. It measures how well the released time series $\{s_t, t = 1, ..., T\}$ follows the original series $\{x_t, t = 1, ..., T\}$. It is defined as follows:

$$E = \frac{1}{T} \sum_{t=1}^{T} \frac{|s_t - x_t|}{\max\{x_t, \delta\}}. \tag{6.10}$$

where $\delta = 1$ in order to handle cases where $x_t = 0$. Smaller values of E indicate high similarity between the released and the original series.

To evaluate the usefulness of the release Kalman count estimates, the E values for the Laplace perturbed counts and the released Kalman count estimates corresponding to three interval privacy budget values $\varepsilon_i = 0.01, 0.1$ and 1 were computed. The obtained E values for the Laplace perturbed counts were compared to the E values for the Kalman count estimates.

6.5.3.2 Utility Loss

Utility loss is a relative cumulative difference between the true data points $\{x_t, t = 1, ..., T\}$ and the fuzzed data points $\{s_t, t = 1, ..., T\}$ [78]. It is defined as follows

$$U = \frac{\sum_{i=1}^{N} |s_i - x_i|}{\sum_{i=1}^{N} |x_i|}. \tag{6.11}$$

Small values of this measure indicate higher research utility [78].

We computed U values for the Laplace perturbed series and the Kalman count estimates corresponding to the three interval privacy budget values $\varepsilon_i = 0.01, 0.1$ and 1 in order to evaluate the research usefulness of the released counts. The obtained U values for the Laplace perturbed counts were compared to the U values for the Kalman count estimates.

6.5.3.3 Use of Intrusion Detection Algorithms

The CUSUM based intrusion detection algorithm and the Adaptive Threshold algorithm are used in this work to determine if inferences made using the Kalman count estimates are close to the corresponding ones using the original data. Specifically, in this work, the false positive rates obtained from the two algorithms' detection thresholds (h and k respectively) using the Kalman estimates are compared to those obtained using the original data. The comparison is done in order to determine if the two algorithms' false positives rates obtained from the Kalman

count estimates are close to those obtained from the original counts. The two algorithms' detection thresholds ranged from 1 to 10. The detection threshold values were selected such that the algorithm's false positive rates approach zero.

6.6 Results

This section presents the results of the utility evaluation of the Kalman count estimates. Figure 6.5 presents the average relative error values for the Laplace perturbed counts and Kalman count estimates at three interval privacy budget values $\varepsilon_i = 0.01, 0.1$ and 1. As indicated in Figure 6.5, the average relative errors for the Laplace perturbed counts were 67701, 6770 and 677 for $\varepsilon_i = 0.01, 0.1$ and 1, respectively while the Kalman count estimates resulted in average relative errors of 984, 434 and 136 for $\varepsilon_i = 0.01, 0.1$ and 1, respectively. These results indicate that the Kalman count estimates which are the released counts are closer to the original counts.

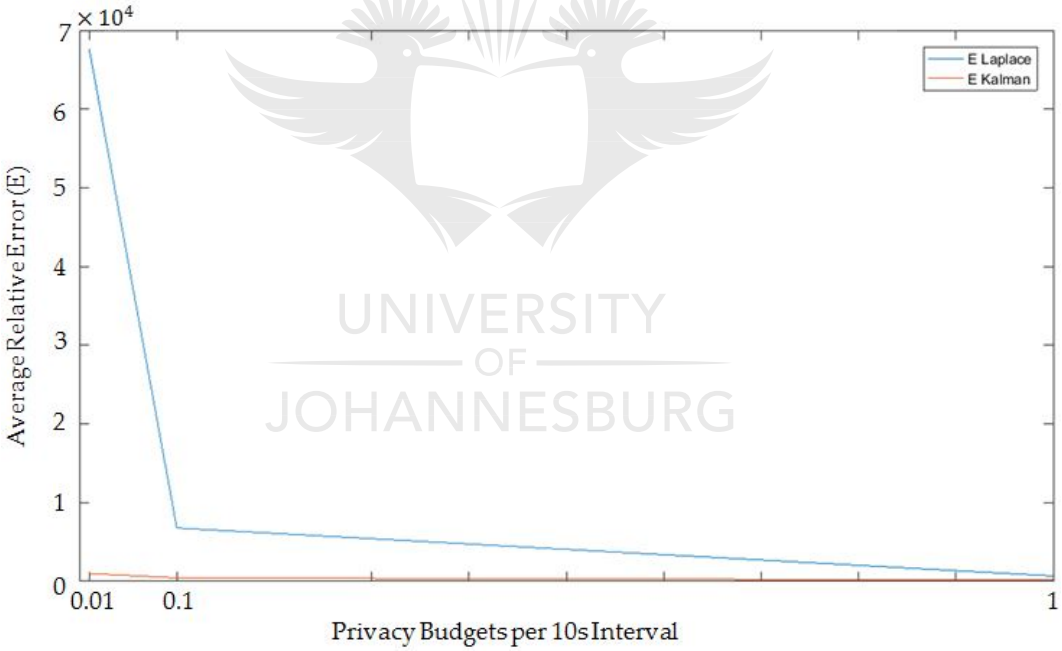


FIGURE 6.5: Average relative error comparison.

Figure 6.6 presents the utility loss values for the Laplace perturbed counts and Kalman count estimates at three interval privacy budget values $\varepsilon_i = 0.01, 0.1$ and 1. As indicated in Figure 6.6, the utility loss values for the Laplace perturbed counts were 47262, 4725 and 472 for $\varepsilon_i = 0.01, 0.1$ and 1 respectively. The Kalman count estimates resulted in utility loss values of 679, 300 and 94 for $\varepsilon_i = 0.01, 0.1$ and 1 respectively. These results indicate that the Kalman count estimates have higher research utility than the Laplace perturbed counts.

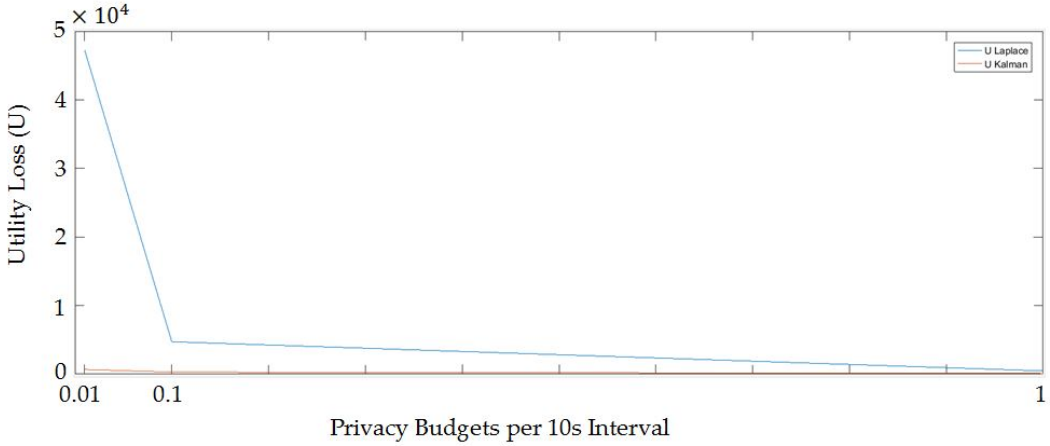


FIGURE 6.6: Utility loss comparison.

Figure 6.7 presents the CUSUM based algorithm false positive rates of the original counts against the Kalman count estimates. When one looks at the overall pattern of the curves in Figure 6.7, the Kalman count estimates curve for $h \leq 8$ tend to follow the pattern of the original counts curve for $h \leq 6$ with lag effect which means inferences made using the Kalman count estimates for $h \leq 8$ will not be too far from the inferences made using the original counts for $h \leq 6$.

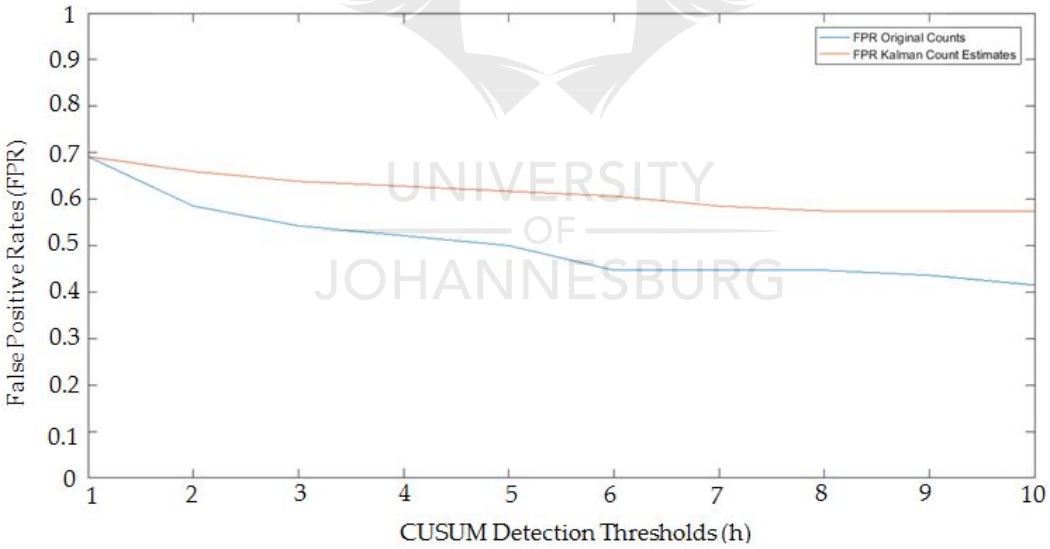


FIGURE 6.7: CUSUM false positive rates for the original counts vs Kalman estimates.

Figure 6.8 presents the Adaptive Threshold algorithm false positive rates of the original counts against the Kalman count estimates. From Figure 6.8, the Kalman count estimates curve for $3 \leq k \leq 5$ tends to follow the pattern of the original counts curve for $3 \leq k \leq 4$ with a lag effect, which means inferences made using the Kalman count estimates for $3 \leq k \leq 5$ will not be too different from the inferences made using the original counts for $3 \leq k \leq 4$.

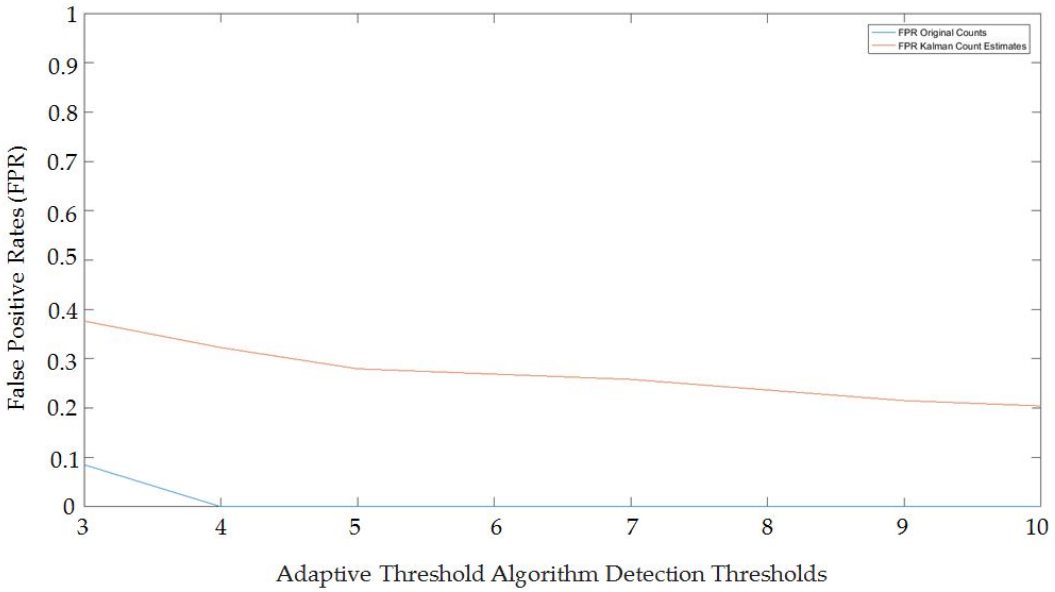


FIGURE 6.8: Adaptive threshold algorithm false positive rates for the original counts vs Kalman estimates.

6.7 Summary and Discussion

In this Chapter, differential privacy was used as a means of providing privacy to TCP SYN packets counts. The filtering component of [25] was adopted in order to improve the accuracy of the released counts. The utility of Kalman count estimates was tested by using two utility metrics, average relative error and utility loss. The performances of cumulative sum based intrusion detection and adaptive threshold algorithms on the original counts were compared to their performances on the released counts. The utility measure, average relative error at $\varepsilon_i = 0.01$, indicates that the Kalman estimates are closer to the original counts as compared to the Laplace perturbed counts. The Utility loss measure at $\varepsilon_i = 0.01$ shows that the released counts have higher research utility as compared to the Laplace counts while preserving privacy.

The cumulative sum based intrusion detection algorithm performance on the original counts and Kalman count estimates indicates that the false positive rates obtained using the Kalman count estimates are not that different from those obtained using the original counts for most of the algorithm's detection threshold, with added advantage of privacy and being estimates of the original counts instead of being simulated counts. The false positive rates obtained from the CUSUM algorithm detection thresholds using the released counts are closer to the ones obtained from the original counts as compared to those obtained from the adaptive threshold algorithm. However, the run time of the CUSUM algorithm is longer than that of the adaptive threshold algorithm.

Chapter 7

Conclusion

7.1 Summary of Conclusions

In order to illustrate the achievable performances of the existing NIDSs, this Thesis evaluated the performances of Adaptive Threshold algorithm, Cumulative Sum based algorithm, Decision Tree and Fuzzy Logic based system for individual NIDSs and the logic OR operator for combined NIDSs in detecting the TCP SYN flooding attack. The analysis of the anomaly based individual NIDSs and logic OR operator based NIDS indicates that the attacks generated using the Poisson process were detected better than those generated at constant interarrival times. Furthermore, the logic OR operator based NIDS outperformed the two individual anomaly based NIDSs in terms of detection probability and detection delay. For the learning based NIDSs, the Decision Tree performed better than the novel Fuzzy Logic based NIDS in terms of accuracy, specificity, false positive and false negative rates.

One of the aims of this research was to determine the upper bounds on the accuracies of network intrusion detection systems (NIDSs) that are based on combining multiple NIDSs. Specifically, the information theoretic measures, information gain of the features used in building the NIDS and dataset entropy with respect to the features, were used to define these bounds under two sensor specifications, namely, a NIDS that combines multiple independent sensors and a NIDS that combines multiple dependent sensors. The upper bounds on the accuracies of the two ensembles of classifiers based NIDSs were empirically determined in this study. In the NIDS that uses ensemble of classifiers with dependent base classifiers, the decision stump was used as the weak base classifier and its accuracy was boosted to optimality using the boosting ensemble method algorithm AdaBoost. In the NIDS that uses an ensemble of classifiers with independent base classifiers the decision tree was used as the unstable base classifier and its accuracy was optimised using the Bagging ensemble method. The results indicate that if the dataset entropy falls between 0.9578 and 0.9586 and the average information gain value amongst features used in the two ensembles falls between 0.045615 and 0.25615 then the accuracy of

the NIDS with dependent base learners will at most be 0.9065 while that of the NIDS with independent base learners will at best be 0.9193 .

On determining the upper bounds on the accuracies of the two NIDSs the following conclusions were reached:

a) different range of values of the average information gain amongst features used in the ensembles lead to different ensemble accuracy upper bounds. The range of values of the average information gain amongst features in the ensemble that lead to equal ensemble accuracy upper bounds would result from the addition of poor features in the ensemble that change the average information gain amongst features but not the accuracy of the ensemble.

b) the obtained ensemble accuracy bounds hold even for the detection of different attacks (normal vs any attack type) provided that the ensemble (AdaBoosted decision stump ensemble or Bagged decision tree ensemble) is built from continuous features with the same information gain values as features used in building the two ensembles and the discretisation of the continuous features is done according to this Thesis.

c) how the continuous features are discretised results to different values of the information gain of the features.

d) features with the same information gain, coming from different datasets may lead to different accuracy bounds if the different datasets have different dataset entropies.

The true positive and false positive rates associated with these bounds provide insight to NIDS designers as to what true positive and false positive rate values must they set their NIDS to in order to obtain a certain level of accuracy.

Another aim of this Thesis was to provide privacy-preserving network trace. Differential privacy, which is a privacy-preserving technique for data publishing, was implemented on the number of TCP SYN packets associated with HTTP requests. The utility analysis of the released privatised number of TCP SYN packets associated with HTTP request indicates that

- the false positive rates of the released counts obtained at some of the detection thresholds of the two anomaly based algorithms are closer to the false positive rates of the original counts at those thresholds.
- the false positive rates obtained from the CUSUM algorithm detection thresholds using the released counts are closer to the ones obtained using original counts as compared to those of the Adaptive Threshold algorithm.
- the CUSUM algorithm has more detection thresholds that lead to useful research inferences as compared to the Adaptive Threshold algorithm.

Where useful research inferences means that false positive rates from the released counts will be not that different from false positive rates obtained using the original counts. Therefore, it is concluded that the released counts

- are research useful while preserving privacy.
- will work well for some algorithms and not so well for others.

7.2 Suggestions for Future Work

This research only determined the upper bounds on the accuracies of ensemble of classifiers-based network intrusion detection systems that use decision stump and decision tree as base learners. An investigation on the upper bounds on the accuracy of network intrusion detection systems that use other base learners like neural networks can be conducted. The performance bounds of two homogenous ensembles were determined, this investigation can also be extended to heterogeneous ensembles. Both ensembles were built using continuous features only, determining upper bounds on accuracy for ensembles built from discrete features or a combination of continuous and discrete features may be another research topic. A comparison of upper bounds for ensembles built from continuous, discrete and combination of continuous and discrete features to determine which one is superior may be another research topic.

References

[1] K. Curtin, *Sa business unprepared for the growing risk of cyber attacks.* 2013. [Online]. Available: `www.fanews.co.za/article/short-term-insurance/1`.

[2] T. Seals, *Cyber-attack volume doubled in first half of 2017*, 2017. [Online]. Available: `https://www.infosecurity-magazine.com/news/cyberattack-volume-doubled-2017/`.

[3] T. Lappas and K. Pelechrinis, "Data mining techniques for (network) intrusion detection systems", *Department of Computer Science and Engineering UC Riverside, Riverside CA*, vol. 92521, 2007.

[4] C. Kruegel, F. Valeur, and G. Vigna, *Intrusion detection and correlation: challenges and solutions*. Springer Science & Business Media, 2004, vol. 14.

[5] W. Hu and W. Hu, "Network-based intrusion detection using adaboost algorithm", in *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, IEEE Computer Society, 2005, pp. 712–717.

[6] S. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review", *Applied Soft Computing*, vol. 10, no. 1, pp. 1–35, 2010.

[7] A. Mokarian, A. Faraahi, and A. G. Delavar, "False positives reduction techniques in intrusion detection systems-a review", *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 13, no. 10, p. 128, 2013.

[8] D. G. Bhatti, P. V. Virparia, and B. Patel, "Conceptual framework for soft computing based intrusion detection to reduce false positive rate", *International Journal of Computer Applications*, vol. 44, no. 13, pp. 1–3, 2012.

[9] M Khalilian, N Mustapha, M. N. Sulaiman, and A Mamat, "Intrusion detection system with data mining approach: A review", *Global Journal of Computer Science and Technology*, 2011.

[10] G. P. Spathoulas and S. K. Katsikas, "Reducing false positives in intrusion detection systems", *computers & security*, vol. 29, no. 1, pp. 35–44, 2010.

[11] C. Thomas and N. Balakrishnan, "Advanced sensor fusion technique for enhanced intrusion detection", in *Intelligence and Security Informatics, 2008. ISI 2008. IEEE International Conference on*, IEEE, 2008, pp. 173–178.

[12] J. Tian, W. Zhao, R. Du, and Z. Zhang, "Ds evidence theory and its data fusion application in intrusion detection", in *Parallel and Distributed Computing, Applications and Technologies, 2005. PDCAT 2005. Sixth International Conference on*, IEEE, 2005, pp. 115–119.

[13] C. Thomas and N. Balakrishnan, "Selection of intrusion detection system threshold bounds for effective sensor fusion.", in *Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security*, 2007, p. 657 007.

[14] C. H. Rowland, *Intrusion detection system*, US Patent 6,405,318, 2002.

[15] C. Siaterlis and B. Maglaris, "Towards multisensor data fusion for dos detection", in *Proceedings of the 2004 ACM symposium on Applied computing*, ACM, 2004, pp. 439–446.

[16] Y. Wang, H. Yang, X. Wang, and R. Zhang, "Distributed intrusion detection system based on data fusion method", in *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, IEEE, vol. 5, 2004, pp. 4331–4334.

[17] E. Menahem, G. Nakibly, and Y. Elovici, "Network-based intrusion detection systems go active!", in *Proceedings of the 2012 ACM conference on Computer and communications security*, ACM, 2012, pp. 1004–1006.

[18] C. Thomas and N. Balakrishnan, "Performance enhancement of intrusion detection systems using advances in sensor fusion", *Supercomputer Education and Research Centre Indian Institute of Science, Doctoral Thesis, 304pp. Available at: http://www. serc. iisc. ernet. in/graduation-theses/CizaThomas-PhD-Thesis. pdf*, 2009.

[19] W. Hu, J. Li, and Q. Gao, "Intrusion detection engine based on dempster-shafer's theory of evidence", in *Communications, Circuits and Systems Proceedings, 2006 International Conference on*, IEEE, vol. 3, 2006, pp. 1627–1631.

[20] W. Gong, W. Fu, and L. Cai, "A neural network based intrusion detection data fusion model", in *Computational Science and Optimization (CSO), 2010 Third International Joint Conference on*, IEEE, vol. 2, 2010, pp. 410–414.

[21] J. C. Mogul and M. Arlitt, "Sc2d: An alternative to trace anonymization", in *Proceedings of the 2006 SIGCOMM workshop on Mining network data*, ACM, 2006, pp. 323–328.

[22] R. Paul, V. C. Valgenti, and M. S. Kim, *Obfuscating and anonymizing network traffic & a new dimension to network research*, 2010. [Online]. Available: `https://research.wsulibs.wsu.edu/xmlui/bitstream/handle/2376/2655/Paul,%20R%20Obfuscating%20and%20anonymizing%20.pdf`.

[23] S. E. Coull, C. V. Wright, F. Monrose, M. P. Collins, and M. K. Reiter, "Playing devil's advocate: Inferring sensitive information from anonymized network traces.", in *NDSS*, vol. 7, 2007, pp. 35–47.

[24] F. McSherry and R. Mahajan, "Differentially-private network trace analysis", *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 123–134, 2011.

[25] L. Fan and L. Xiong, "Real-time aggregate monitoring with differential privacy", in *Proceedings of the 21st ACM international conference on Information and knowledge management*, ACM, 2012, pp. 2169–2173.

[26] A. Siraj and R. B. Vaughn, "Multi-level alert clustering for intrusion detection sensor data", in *Fuzzy Information Processing Society, 2005. NAFIPS 2005. Annual Meeting of the North American*, IEEE, 2005, pp. 748–753.

[27] J. L. Hellerstein, F. Zhang, and P. Shahabuddin, "Characterizing normal operation of a web server: Application to workload forecasting and problem detection", in *CMG-CONFERENCE-*, COMPSCER MEASUREMENT GROUP INC, vol. 1, 1998, pp. 150–160.

[28] C. Thomas and N. Balakrishnan, "Improvement in intrusion detection with advances in sensor fusion", *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 3, pp. 542–551, 2009.

[29] G. Thimm and E. Fiesler, "Pruning of neural networks", IDIAP, Tech. Rep., 1997.

[30] A. P. F. Chan, W. W. Ng, D. S. Yeung, and E. C. C. Tsang, "Comparison of different fusion approaches for network intrusion detection using ensemble of rbfnn", in *Proceedings of 2005 international conference on machine learning and cybernetics*, vol. 6, 2005, pp. 18–21.

[31] J. Kaliappan, R. Thiagarajan, and K. Sundararajan, "Fusion of heterogeneous intrusion detection systems for network attack detection", *The Scientific World Journal*, vol. 2015, pp. 1–8, 2015.

[32] V. Shah, A. K. Aggarwal, and N. Chaubey, "Performance improvement of intrusion detection with fusion of multiple sensors", *Complex & Intelligent Systems*, vol. 3, no. 1, pp. 33–39, 2017.

[33] G. L. Rogova and V. Nimier, "Reliability in information fusion: Literature survey", in *Proceedings of the seventh international conference on information fusion*, vol. 2, 2004, pp. 1158–1165.

[34] V. M. Shah and A. K. Agarwal, "Reliable alert fusion of multiple intrusion detection systems.", *IJ Network Security*, vol. 19, no. 2, pp. 182–192, 2017.

[35] N. N. P. Mkuzangwe, A. McDonald, and F. V. Nelwamondo, "Implementation of anomaly detection algorithms for detecting transmission control protocol synchronized flooding attacks", in *Fuzzy Systems and Knowledge Discovery (FSKD), 2015 12th International Conference on*, IEEE, 2015, pp. 2137–2141.

[36] V. A. Siris and F. Papagalou, "Application of anomaly detection algorithms for detecting syn flooding attacks", *Computer communications*, vol. 29, no. 9, pp. 1433–1442, 2006.

[37] G. Li, Z. Yan, Y. Fu, and H. Chen, "Data fusion for network intrusion detection: A review", *Security and Communication Networks*, vol. 2018, 2018.

[38] A. Borji, "Combining heterogeneous classifiers for network intrusion detection", in *Annual Asian Computing Science Conference*, Springer, 2007, pp. 254–260.

[39] M. Govindarajan and R. M. Chandrasekaran, "Intrusion detection using an ensemble of classification methods", in *Proc. of the World Congress on Engineering and Computer Science*, vol. 1, 2012, pp. 459–464.

[40] P. Sornsuwit and S. Jaiyen, "Intrusion detection model based on ensemble learning for u2r and r2l attacks", in *Information Technology and Electrical Engineering (ICITEE), 2015 7th International Conference on*, IEEE, 2015, pp. 354–359.

[41] D. Prusti and S. K. Jena, *An efficient intrusion detection model using ensemble methods*, 2015. [Online]. Available: `http://ethesis.nitrkl.ac.in/7304/1/Efficient_Prusti_2015.pdf`.

[42] A. Zainal, M. A. Maarof, and S. M. Shamsuddin, "Ensemble classifiers for network intrusion detection system", *Journal of Information Assurance and Security*, vol. 4, no. 3, pp. 217–225, 2009.

[43] P. Natesan, P. Balasubramanie, and G. Gowrison, "Improving attack detection rate in network intrusion detection using adaboost algorithm", *Journal of Computer Science*, vol. 8, no. 7, pp. 1041–1048, 2012.

[44] G. Kumar and K. Kumar, "The use of artificial-intelligence-based ensembles for intrusion detection: A review", *Applied Computational Intelligence and Soft Computing*, vol. 2012, p. 21, 2012.

[45] A. A. Aburomman and M. B. I. Reaz, "A survey of intrusion detection systems based on ensemble and hybrid classifiers", *Computers & Security*, vol. 65, pp. 135–152, 2017.

[46] M. Jabbar, R. Aluvalu, and S Reddy, "Cluster based ensemble classification for intrusion detection system", in *Proceedings of the 9th International Conference on Machine Learning and Computing*, ACM, 2017, pp. 253–257.

[47] W. Yassin, N. I. Udzir, Z. Muda, M. N. Sulaiman, *et al.*, "Anomaly-based intrusion detection through k-means clustering and naives bayes classification", in *Proc. 4th Int. Conf. Comput. Informatics, ICOCI*, 2013, pp. 298–303.

[48] S. S. S. Sindhu, S. Geetha, and A. Kannan, "Decision tree based light weight intrusion detection using a wrapper approach", *Expert Systems with applications*, vol. 39, no. 1, pp. 129–141, 2012.

[49] N. N. P. Mkuzangwe and F. Nelwamondo, "Ensemble of classifiers based network intrusion detection system performance bound", in *Systems and Informatics (ICSAI), 2017 4th International Conference on*, IEEE, 2017, pp. 970–974.

[50] N. Moustafa, B. Turnbull, and K.-K. R. Choo, "An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things", *IEEE Internet of Things Journal*, 2018.

[51] F. Haddadi and A. N. Zincir-Heywood, "Benchmarking the effect of flow exporters and protocol filters on botnet traffic classification", *IEEE Systems journal*, vol. 10, no. 4, pp. 1390–1401, 2014.

[52] Z. Abaid, D. Sarkar, M. A. Kaafar, and S. Jha, "The early bird gets the botnet: A markov chain based early warning system for botnet attacks", in *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, IEEE, 2016, pp. 61–68.

[53] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection", *arXiv preprint arXiv:1802.09089*, 2018.

[54] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest", in *2008 Eighth IEEE International Conference on Data Mining*, IEEE, 2008, pp. 413–422.

[55] D. Reynolds, "Gaussian mixture models", *Encyclopedia of biometrics*, pp. 827–832, 2015.

[56] S. Calinon and A. Billard, "Incremental learning of gestures by imitation in a humanoid robot", in *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, ACM, 2007, pp. 255–262.

[57] Y. Mirsky, T. Halpern, R. Upadhyay, S. Toledo, and Y. Elovici, "Enhanced situation space mining for data streams", in *Proceedings of the Symposium on Applied Computing*, ACM, 2017, pp. 842–849.

[58] *Suricata- open source ids/ ips/nsm engine.* 2017. [Online]. Available: `https://suricata-ids.org/`.

[59] Y. Song, "A behavior-based approach towards statistics-preserving network trace anonymization", PhD thesis, Columbia University, 2012.

[60] D. Koukis, S. Antonatos, D. Antoniades, E. P. Markatos, and P. Trimintzios, "A generic anonymization framework for network traffic", in *2006 IEEE International Conference on Communications*, IEEE, vol. 5, 2006, pp. 2302–2309.

[61] N. R. Adam and J. C. Worthmann, "Security-control methods for statistical databases: A comparative study", *ACM Computing Surveys (CSUR)*, vol. 21, no. 4, pp. 515–556, 1989.

[62] F. D. McSherry, "Privacy integrated queries: An extensible platform for privacy-preserving data analysis", in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, ACM, 2009, pp. 19–30.

[63] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies", in *ACM SIGCOMM Computer Communication Review*, ACM, vol. 34, 2004, pp. 219–230.

[64] B. Eriksson, P. Barford, and R. Nowak, "Network discovery from passive measurements", in *ACM SIGCOMM Computer Communication Review*, ACM, vol. 38, 2008, pp. 291–302.

[65] R. Pang and V. Paxson, "A high-level programming environment for packet trace anonymization and transformation", in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, ACM, 2003, pp. 339–351.

[66] J. Xu, J. Fan, M. H. Ammar, and S. B. Moon, "Prefix-preserving ip address anonymization: Measurement-based security evaluation and a new cryptography-based scheme", in *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*, IEEE, 2002, pp. 280–289.

[67] J. Mirkovic, "Privacy-safe network trace sharing via secure queries", in *Proceedings of the 1st ACM workshop on Network data anonymization*, ACM, 2008, pp. 3–10.

[68] N. V. Dijkhuizen and J. V. D. Ham, "A survey of network traffic anonymisation techniques and implementations", *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, p. 52, 2018.

[69] A. J. Slagell, K. Lakkaraju, and K. Luo, "Flaim: A multi-level anonymization framework for computer and network logs.", in *LISA*, vol. 6, 2006, pp. 3–8.

[70] A. J. Slagell, Y. Li, and K. Luo, "Sharing network logs for computer forensics: A new tool for the anonymization of netflow records", in *Security and Privacy for Emerging Areas in Communication Networks, 2005. Workshop of the 1st International Conference on*, IEEE, 2005, pp. 37–42.

[71] T. Gamer, C. Mayer, and M. Schöller, "Pktanon–a generic framework for profile-based traffic anonymization", *PIK-Praxis der Informationsverarbeitung und Kommunikation*, vol. 31, no. 2, pp. 76–81, 2008.

[72] T. Farah and L. Trajković, "Anonym: A tool for anonymization of the internet traffic", in *Cybernetics (CYBCONF), 2013 IEEE International Conference on*, IEEE, 2013, pp. 261–266.

[73] S. E. Coull, C. V. Wright, A. D. Keromytis, F. Monrose, and M. K. Reiter, "Taming the devil: Techniques for evaluating anonymized network data.", in *NDSS*, 2008.

[74] L. Fan, L. Bonomi, L. Xiong, and V. Sunderam, "Monitoring web browsing behavior with differential privacy", in *Proceedings of the 23rd international conference on World wide web*, ACM, 2014, pp. 177–188.

[75] J. Blocki, A. Datta, and J. Bonneau, "Differentially private password frequency lists.", *IACR Cryptology ePrint Archive*, vol. 2016, p. 153, 2016.

[76] F. McSherry and K. Talwar, "Mechanism design via differential privacy", in *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, IEEE, 2007, pp. 94–103.

[77] M. Hardt, K. Ligett, and F. McSherry, "A simple and practical algorithm for differentially private data release", in *Advances in Neural Information Processing Systems*, 2012, pp. 2339–2347.

[78] X. Deng and J. Mirkovic, "Commoner privacy and a study on network traces", in *Proceedings of the 33rd Annual Computer Security Applications Conference*, ACM, 2017, pp. 566–576.

[79] N. N. P. Mkuzangwe and F. Nelwamondo, "Differentially private transmission control protocol synchronise packet counts", *International Journal of Network Security*, vol. 21, no. 5, pp. 835–842, 2019.

[80] M. Basseville and I. V. Nikiforov, *Detection of abrupt changes: theory and application*. Prentice Hall Englewood Cliffs, 1993, vol. 104.

[81] L. A. Zadeh, "Fuzzy sets", *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.

[82] F. Dernoncourt, *Introduction to fuzzy logic*, 2013. [Online]. Available: `http://www.francky.me/doc/course/fuzzy_logic.pdf`.

[83] W. E. Sari, O. Wahyunggoro, and S. Fauziati, "A comparative study on fuzzy mamdani-sugeno-tsukamoto for the childhood tuberculosis diagnosis", in *AIP Conference Proceedings*, AIP Publishing, vol. 1755, 2016, p. 070 003.

[84] B. Hssina, A. Merbouha, H. Ezzikouri, and M. Erritali, "A comparative study of decision tree id3 and c4. 5", *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 2, 2014.

[85] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[86] L. Breiman, "Heuristics of instability and stabilization in model selection", *The annals of statistics*, vol. 24, no. 6, pp. 2350–2383, 1996.

[87] R. R. Brooks and S. S. Iyengar, *Multi-sensor fusion: fundamentals and applications with software*. Prentice-Hall, Inc., 1998.

[88] Y. Zeng, J. Zhang, and J. L. Van Genderen, "Comparison and analysis of remote sensing data fusion techniques at feature and decision levels", in *ISPRS Commission VII Mid-term Symposium" Remote Sensing: From Pixels to Processes*, 2006.

[89] D. L. Hall and S. A. H. McMullen, *Mathematical techniques in multisensor data fusion*. Artech House, 2004.

[90] B. Parhami, "A taxonomy of voting schemes for data fusion and dependable computation", *Reliability Engineering and System Safety*, vol. 52, no. 2, pp. 139–151, 1996.

[91] M. Van Erp, L. Vuurpijl, and L. Schomaker, "An overview and comparison of voting methods for pattern recognition", in *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*, IEEE, 2002, pp. 195–200.

[92] A. P. Piotrowski and J. J. Napiorkowski, "A comparison of methods to avoid overfitting in neural networks training in the case of catchment runoff modelling", *Journal of Hydrology*, vol. 476, pp. 97–111, 2013.

[93] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[94] L. I. Kuncheva, *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2004.

[95] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization", *Machine learning*, vol. 40, no. 2, pp. 139–157, 2000.

[96] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting", in *European conference on computational learning theory*, Springer, 1995, pp. 23–37.

[97] R. E. Schapire, "Explaining adaboost", in *Empirical inference*, Springer, 2013, pp. 37–52.

[98] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants", *Machine learning*, vol. 36, no. 1, pp. 105–139, 1999.

[99] D. D. Protić, "Review of kdd cup'99, nsl-kdd and kyoto 2006+ datasets", *Vojnotehnički glasnik*, vol. 66, no. 3, pp. 580–596, 2018.

[100] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "Selecting features for intrusion detection: A feature relevance analysis on kdd 99 intrusion detection datasets", in *Proceedings of the third annual conference on privacy, security and trust*, 2005.

[101] *Darpa intrusion detection data sets*, 1998. [Online]. Available: `https://www.ll.mit.edu/ideval/data/`.

[102] M. Tavallaee, E. Bagheri, W Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set", in *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, IEEE, 2009, pp. 1–6.

[103] *Kdd cup 1999 dataset*, 1999. [Online]. Available: `https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`.

[104] *Cicids2017 dataset*, 2017. [Online]. Available: www.unb.ca/cic/datasets/ids-2017.html.

[105] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization", in *ICISSP*, 2018, pp. 108–116.

[106] S. H. C. Haris, R. B. Ahmad, and M. A.H. A. Ghani, "Detecting tcp syn flood attack based on anomaly detection", in *Network Applications Protocols and Services (NETAPPS), 2010 Second International Conference on*, IEEE, 2010, pp. 240–244.

[107] T. Zhang, "Cumulative sum algorithm for detecting syn flooding attacks", *Computing Research Repository (CoRR)*, vol. abs/1212.5129, 2012. [Online]. Available: http://arxiv.org/abs/1212.5129.

[108] S. Wang, Q. Sun, H. Zou, and F. Yang, "Detecting syn flooding attacks based on traffic prediction", *Security and Communication Networks*, vol. 5, no. 10, pp. 1131–1140, 2012.

[109] G. Thatte, U. Mitra, and J. Heidemann, "Parametric methods for anomaly detection in aggregate traffic", *IEEE/ACM Transactions On Networking*, vol. 19, no. 2, pp. 512–525, 2011.

[110] R. R. Kompella, S. Singh, and G. Varghese, "On scalable attack detection in the network", in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, ACM, 2004, pp. 187–200.

[111] N. N. P. Mkuzangwe and F. V. Nelwamondo, "A fuzzy logic based network intrusion detection system for predicting the tcp syn flooding attack", in *Asian conference on intelligent information and database systems*, Springer, 2017, pp. 14–22.

[112] M. Beaumont-Gay, "A comparison of syn flood detection algorithms", in *Internet Monitoring and Protection, 2007. ICIMP 2007. Second International Conference on*, IEEE, 2007, pp. 9–9.

[113] P. Langley and S. Sage, "Pruning irrelevant features from oblivious decision trees", *target*, vol. 4, p. 5, 1993.

[114] N. K. Choudhary, Y. Shinde, R. Kannan, and V. Venkatraman, "Impact of attribute selection on the accuracy of multilayer perceptron", *Int. J. IT Knowl. Manag.(IJITKM)*, vol. 7, no. 2, pp. 32–36, 2014.

[115] P. Tahmasebi and A. Hezarkhani, "Application of adaptive neuro-fuzzy inference system for grade estimation; case study, sarcheshmeh porphyry copper deposit, kerman, iran", *Australian Journal of Basic and Applied Sciences*, vol. 4, no. 3, pp. 408–420, 2010.

[116] B. Sui, *Information gain feature selection based on feature interactions*, 2013. [Online]. Available: https://uh-ir.tdl.org/uh-ir/handle/10657/523.

[117] L. Fan and L. Xiong, "Differentially private anomaly detection with a case study on epidemic outbreak detection", in *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, IEEE, 2013, pp. 833–840.

[118] A. Blum, K. Ligett, and A. Roth, "A learning theory approach to noninteractive database privacy", *Journal of the ACM (JACM)*, vol. 60, no. 2, p. 12, 2013.

[119] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis", in *TCC*, Springer, vol. 3876, 2006, pp. 265–284.

[120] W. Yurcik, C. Woolam, G. Hellings, L. Khan, and B. Thuraisingham, "Toward trusted sharing of network packet traces using anonymization: Single-field privacy/analysis tradeoffs", *arXiv preprint arXiv:0710.3979*, 2007.

[121] T. Zhu, G. Li, W. Zhou, and S. Y. Philip, "Preliminary of differential privacy", in *Differential Privacy and Applications*, Springer, 2017, pp. 7–16.

[122] R. E. Kalman, "A new approach to linear filtering and prediction problems", *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[123] C. Dwork, "Differential privacy: A survey of results", in *International Conference on Theory and Applications of Models of Computation*, Springer, 2008, pp. 1–19.