CHAPTER ONE

INTRODUCTION

1.1 Background to the Study

The Internet has rapidly evolved into what can be termed as “Information Superhighway” such that public and government organizations, businesses, educational bodies and individual users are connecting to the Internet via various sophisticated devices. But this increasing popularity of the Internet comes with inherent security risks. While the Internet eases information access for the legitimate users, it does the same for intruders also. Consequently, Internet sites are often prime targets for malicious activity like break-ins, service disruptions, system tampering, etc.

With the present astronomical growth of the Internet and the fact that most business transactions, and even governance, are performed online, the issue of security of network systems has become more prominent than before. Adding to this is the Internet of Things (IoT) which has made online activities become ubiquitous and as such increasing the vulnerability of systems.

Attendant to the advancement of technology and global interconnectivity of organisations, places, governments and devices, via the internet are new techniques of attacks and new trends of system vulnerability. This has led to more research into techniques for protecting and safeguarding network systems.

1.2 Network Attacks and Network Security

1.2.1 Network Attacks

A network intrusion can be defined as any unauthorized activity on or against a network system, which might be from within or without the network. In common cases, such unwanted activity

consumes network resources intended for legitimate uses, and always threatens the security of the network and/or its data. Properly designing and deploying a network security system will go a long way in curbing such activities.

There are several types of network attacks and these can be classified into four main categories (Sharma and Gupta, 2015; Chen *et al.*, 2016):

i) ***Denial of Service*** (***DoS):*** in this case, an attacker makes network resources too busy to serve legitimate requests. Examples include mail bomb, apache, syn flood

ii) ***Probing*** (***Probe):*** In probing attack, the attacker scans a network device so as to gather information about weaknesses or vulnerabilities that can be exploited to compromise the target system. Examples include nmap, saint, mscan.

iii) ***User to Root*** (***U2R):*** In this category, an authorized user attempts to abuse the vulnerabilities of the system in order to gain privilege of root user he is not authorized for. Example include perl, Fd-format, xterm.

iv) ***Remote to Local*** (***R2L):*** Here, a remote user sends packets to a machine over the internet to gain access as a local user to a local machine i.e. the weaknesses of the system is exploited by an external intruder to access the privileges of a local user. Examples include phf, xlock, guest.

These various categories of network attacks aim at undermining the CIA (Confidentiality, Integrity and Availability) properties of the network (Sodiya *et al.*, 2004).

Recent Internet attacks have followed different patterns depending on the industry being attacked. For instance, in manufacturing industry, attacks usually occur when there is a new product patent (industrial espionage); in the education industry, attacks had increased with the advent and adoption

of online examinations and computer-based tests by educational institutions. Historically, the more serious attacks will often have specific catalysts such as, when a corporation builds a production facility in a developing country that is viewed as an exploitive action by one or more activist groups; a government sponsors a peace conference that is viewed as an attempt to subvert the political viability of a disaffected part of the population; a repressive regime massacres a band of rebels near the capital; an organized crime syndicate reacts to crack downs by law enforcement. Objectives can range from revenge (a disgruntled employee) to political statements (terrorism) to a full-scale attack on infrastructure as an act of warfare or at the very least part of "coercive diplomacy". First, these attacks exploit the interconnectivity of networks.

Distributed denials of service (DDoS) attacks, which is the focus of this research is a subset of DoS. It aims at attacking the availability property of the network such that the system is weighed down attending to illegitimate requests to the extent that it cannot attend to legitimate requests by legitimate users. DDoS attacks deploy a large number of compromised systems on networks to attack a victim system. A ‘botnet’, which consists of a large number of bots, can be used for massive DDoS attacks and spamming operations. Currently in addition, recent attacks are sometimes what are known as zero-day threats. A worm propagates actively over a network and spreads very rapidly before it is identified and a countermeasure developed. Application layer level DDoS attacks are now used by terrorists to seize confidential information and also to collapse various government websites. Several nations were victims of such attacks. Generally, application layer DDoS attacks are similar to Flash Crowd event (Rao *et al.*, 2012), which is a normal situation when huge number of legitimate users tries to access the server and in turn server goes down. This, in effect, has made Application layer DDoS attacks detection more and more difficult. Attack tools as well have become more sophisticated and evolutionary.

Thus, it is more difficult to discover the signature of attacks and to detect them through signature-based systems such as intrusion detection systems (IDS) or anti-virus software. Also, the attacks on infrastructure have increased. DDoS, worms, domain name system (DNS) attacks and router attacks are included in these attacks. These attacks are highly detrimental to the availability of the Internet. In 2001, the Code Red worm infected more than 359,000 systems in less than 14 hours and caused a global slowdown of the Internet (Moore *et al.*, 2002.). On 25 January 2003, the Slammer worm infected more than 90 percent of vulnerable hosts within 10 minutes (Moore *et al.*, 2003.). Due to the huge amount of network management traffic initiated by the scanning activity of the worm, Internet services were shut down for many hours in Korea. A recent occurrence of DDoS attack is the large-scale DDoS against New Hampshire-based Internet performance company, Dyn, which caused major Internet disruptions on Friday, 21st October, 2016. The attack disrupted internet service across Europe and United States of America (USA). Users were unable to access many major websites such as Twitter, Spotify, Netflix, Amazon, Tumblr, Reddit and other sites (York, 2016). According to Hilton (2016), the attack, which was a botnet coordinated through a large number of Internet of Things-enabled (IoT) devices (such as cameras, residential gateways, and baby monitors) that had been infected with Mirai malware, opened up an important conversation about internet security and volatility. Apart from highlighting the vulnerabilities in the security of IoT devices that need to be addressed, it has also generated further discussion in the internet infrastructure community concerning the future of the internet.

1.2.2 Network Security

Network security is all the provisions and policies adopted by a network administrator to prevent and monitor unauthorized access, misuse, modification, or denial of a computer network and

network-accessible resources. The main aim of network security is to ensure and protect the CIA (Confidentiality, Integrity and Availability) properties of the network systems. *Confidentiality* ensures that the data and system are not accessible to unauthorized individuals, processes, or systems; *Integrity* ensures that the meaning, completeness, consistency, intended use, and correlation to representation of the data is preserved; *Availability* ensures that the data and other system resources are accessible and usable to authorized persons and/or processes (Sodiya *et al.*, 20+04).

As earlier mentioned, network security involves the authorization of access to data in a network, which is normally controlled by the network administrator. Networks can be private, such as within a company, and others which might be open to public access. Network security is a serious matter of concern in organizations, enterprises, and other types of institutions. In its simplest form, users choose or are assigned an ID and password or other authenticating information that allows him access to information and programs within his authority. Network security covers a variety of computer networks, both public and private, that are used in everyday jobs for conducting transactions and communications among businesses, government agencies and individuals. As the name implies, it secures the network, as well as protect and oversee operations being performed. But with the growth of the internet technology and expansion of businesses, it has become imperative for organisations to employ better security tools to secure their network and data. This gave birth to the use of various sophisticated security tools such as firewalls, Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS), and recently Intrusion Forecasting System (IFS), which also encompasses intrusion prediction systems.

In today's business world, information and also availability of the network resources are valuable assets of any organization and hence requires appropriate management and protection. As organizations are increasing their reliance on the distributed computing environment, they are becoming more vulnerable to security breaches. According to Lee *et al.* (2002), any breach of security to these resources can result in tarnished reputation of the organization, loss of customer's confidence or even lawsuits. Many other mechanisms and technologies like firewalls, encryption, authentication, vulnerability checking, access control policies can offer security but it is still susceptible to attacks from hackers who take advantage of system flaws and social engineering tricks.

1.3 Intrusion Detection, Prevention and Prediction Systems

1.3.1 Intrusion Detection Systems (IDS)

Intrusion Detection Systems (IDS), since James Anderson introduced it in 1980, as cited in Govindu (2005), has become a standard component of network security architectures, complementing conventional techniques such as firewalls, encryption, and authentication. Intrusion Detection (ID) is the process of monitoring events occurring in a network system and alerting responsible parties whenever irregular (suspicious) activity occurs. The system that does this monitoring is called Intrusion Detection System (IDS). IDS scans all packets on the network and attempts to classify the network traffic as intrusive or non-intrusive. IDS tools have the capability to distinguish between insider attacks originating from within the organisation and external attacks posed by hackers.

According to Grandison *et al.* (2007), IDSs consist of:

1) an agent that collects the information on the stream of monitored events,

2) an analysis engine that detects signs of intrusion, and

3) a response module that generates responses based on the outcome from the analysis engine.

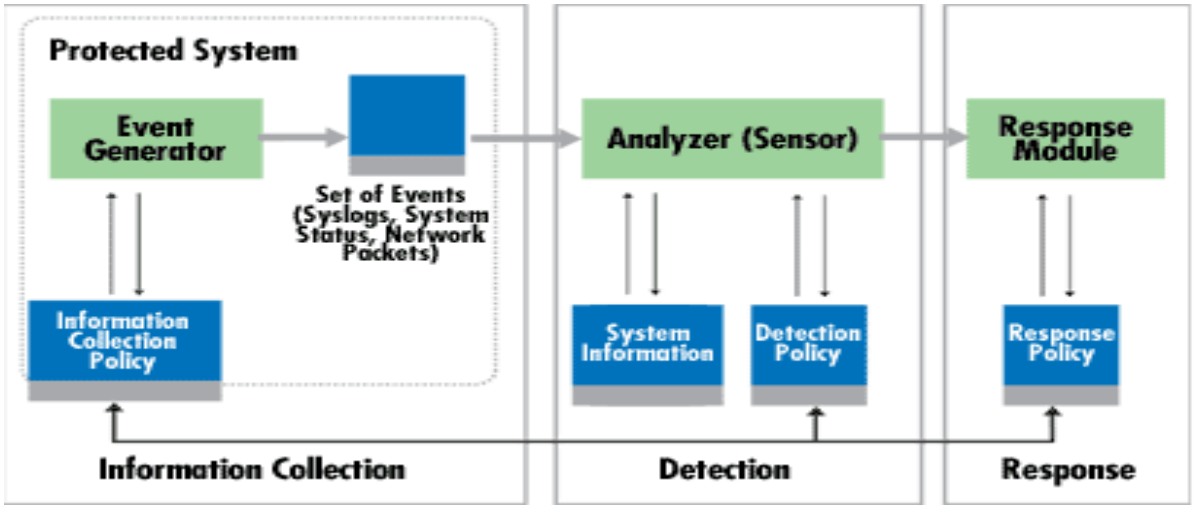


Figure 1: IDS components (Lundi and Jonsson, (2002)

1.3.1.1 Classification of Intrusion Detection Systems

IDS, as a network security tool, uses two major intrusion detection techniques (Debar *et al.*, 1999, Sodiya *et al.*, 2004). The first one is ***anomaly detection***, which explores issues in intrusion detection associated with deviations from normal system or user behaviour. The second employs ***signature detection*** to discriminate between anomaly or attack patterns (signatures) and known intrusion detection signatures. Both methods have their distinct advantages and disadvantages as well as suitable application areas of intrusion detection. Misuse detection cannot detect unknown attacks while anomaly detection suffers from a high level of false positive errors.

Based on these two techniques, therefore, IDSs can be classified as Signature-based IDS and Anomaly-based IDS (Karthikeyan and Indra, 2010; Devika and Sunny, 2014).

1) **Signature-based IDS:** In operation, Signature-based IDS, also called knowledge-based IDS, is similar to a virus scanner. This style of detection relies on rules and tries to associate probable patterns to intrusive activities. Viruses usually undergo a number of steps to penetrate a system. It is this series of steps that is compiled into a rule. Whenever the IDS software (an agent) gathers data, it compares it against the rules that have been defined and then classifies it as either a positive or negative activity.

   Signature-based IDS has advantages and disadvantages some of which are:

   Advantages:

   i) It is mostly considered to be more accurate at identifying an intrusive action.
   ii) It is easy to track down cause of alarm due to detailed log files
   iii) It is time-saving since the administrators will spend less time dealing with false positives

   Disadvantages:

   i) It can only detect an intrusive action that matches a pattern that is in the database, therefore there is need to constantly update the signature databases.
   ii) Updating signature databases with newly identified attack pattern is time-consuming.
   iii) In heavy traffic network, it is usually difficult for the IDS to inspect every single packet, which may lead to some packets being dropped thereby making it possible for attack packets to get by without detection.

iv) Systems can suffer a substantial performance degradation in terms of response time if not properly equipped with the necessary hardware to keep up with the demands.

v) difficulty in handling internal attacks. For instance, abuse of a legitimate user privileges is not sensed as a malicious activity by the system due to lack of information on user privileges and structure of attack signature.

2) **Anomaly-based IDS:** it is also referred to as behaviour-based IDS. This class of IDS studies the activities of each user and program in the system over a period of time and then establishes a normal profile of users or programs in the system. It detects attacks that have significant deviation from a normal profile. Anomaly IDS is of two types; it can be either static or dynamic. Static Anomaly-based IDS assumes that one or more sections on the host should remain constant. It focuses only on the software side and ignore any unusual changes in hardware and it is usually used to monitor data integrity. Dynamic Anomaly-based IDS depends on a baseline or profile. The baseline, which is either established by the IDS or the network administrator, tells the system the kind of traffic that looks non-intrusive or normal. It may include information such as bandwidth, ports, time frames, and so on. Some of the advantages and disadvantages of anomaly-based IDS are as below.

Advantages:

i) New threats can be detected without worrying about the up-to-date status of the databases

ii) Very little maintenance is required since, once installed, the system continually learn about network activities and thereby build its profiles.

iii) The system becomes more accurate at identifying threats with time.

Disadvantages:

i) There is tendency for the network to be in an unprotected state during profile building.

ii) If malicious activity looks like normal traffic to the system it will never send an alarm.

iii) False positives can become cumbersome with an anomaly-based IDS. Normal usage such as checking e-mail after a meeting can signal an alarm.

Whereas the Firewall can be likened to a security fence around one's property and guard post at the front gate, IDS are the equivalent of multi-sensor video monitoring and burglar alarm system. IDS does not perform any action to prevent intrusion; its main function is to alert the site security officer or network administrator of possible security violation (Sodiya *et al.*, 2007). In the process of detecting an attack, it is necessary to take corrective action to tackle the attack and ensure safety of the system. But in the case of IDS, the alarm would be raised after the theft might have taken place, which was why research moved on to Intrusion Prevention Systems (IPS).

1.3.2 Intrusion Prevention Systems (IPS)

According to Govindu (2005), an intrusion prevention system is like an armed burglar alarm system. This is due to the fact that IPSs are active and protect the system from an attack rather than just detecting the attacks. IDSs were then replaced by IPSs. As each packet comes into the system, it is deeply analyzed and a "go/no-go" decision is made as to whether it should be allowed to proceed to its destination. If the packet is malicious, it is dropped and is never even seen by the victim. IPS, however, has the following drawbacks:

i) If an abnormal behaviour is detected from a network, legitimate traffic coming from that network is also blocked (Maskat *et al.*, 2011)

ii) in a passive configuration, the IPS and the victim sees the attack at the same time, in which case the damage is often already done by the time the reset is sent.

iii) one of the defining actions of an intrusion prevention is that it actively blocks actions it determines to be malicious. However, if the action was actually a valid action by the application, it would cause the application to malfunction or fail.

For these reasons and more, efforts have moved towards developing intrusion prediction systems. A fundamental reason for forecasting is that if a forecast is accurate, it is always better to make better decisions.

1.3.3 Intrusion Forecasting Systems (IFS) and Intrusion Prediction System (IPrS)

From the afore discussion, it could be seen that IDS, and IPS are reactive kinds of network security systems; they first have to detect the intrusion before they manage it and therein lies their limitation (Pontes and Guelfi, 2010).

IFS and IPrS, on the other hand, are proactive ways of handling intrusion in networks unlike IDS which responds to Internet attacks after serious damage had been done or IPS which slows down the network in the case of heavy network traffic. The strength of IFS or IPrS is in the prediction techniques, which can protect the systems from new security breaches that can result from unknown methods of attack. The speed and precision of prediction is very important. There are three important steps in forecasting - analysis of the present state, prediction of a future state and

interpretation of the model results (Abdullah *et al.*, 2015). The architecture of IFS is similar to that of IDS in that it also consists of data collection module, data analysis, and prediction and reporting module. Also, similar to IDS, intrusion prediction system is of two kinds: signature-based IPrS/IFS and anomaly-based IPrS/IFS each class sharing the same advantages and disadvantages as their IDS counterparts.

1.3.3.1 Techniques for IFS and IPrS

Several methods such as Time Series (Seng *et al.*, 2010), Machine Learning (Zhang *et al.*, 2012; Satpute *et al.*, 2013), Markov Chain (Shin *et al.*, 2013), Hidden Markov Model (HMM) (Cheng *et al.*, 2012; Sendi *et al.*, 2012; Badajena *et al.*, 2012), Statistical Profiling (Saganowski *et al.*, 2013), Data Mining (Sodiya *et al.*, 2004; Berezinski, 2015), Neural Network (Zhang *et al.*, 2005; Kang and Kang, 2016), and combinations of these methods (hybrid methods) (Kim *et al.*, 2010; Siani *et al.*, 2014; Sharma and Manoria, 2015) have been deployed into developing efficient intrusion prediction systems. Some of these methods and techniques were inherited by IFS and IPrS from IDS. This section presents some of these techniques.

1.3.3.1.1 Machine learning

Machine learning is a special branch of artificial intelligence. It acquires knowledge from training data based on known facts. Machine learning, which majorly focuses on prediction, is defined to be a study that allows systems to learn knowledge without being programmed (Haq *et al.*, 2015; Shah *et al.*, 2015). The techniques used in Machine learning are classified into three main categories viz.: supervised learning, unsupervised learning, and reinforcement learning.

In Supervised learning, also known as classification, data instances are labelled in the training phase (Haq *et al.*, 2015). Some popular examples of supervised learning algorithms are Hidden Markov Model (HMM), Artificial Neural Network (ANN), Gaussian Process Regression, Bayesian Statistics, Lazy learning, Nearest Neighbour algorithm, Support Vector Machine (SVM), Bayesian Networks, K-Nearest Neighbour (K-NN), Naive Bayes classifier, and so on.

In unsupervised learning data instances are unlabeled. Clustering is a prominent way for this learning technique. Some common examples of unsupervised learning algorithms are Self-organizing map, Cluster analysis (K-means clustering, Fuzzy clustering), Hierarchical clustering, Apriori algorithm.

Reinforcement learning is an interactive learning algorithm. Here the system interacts with an environment to achieve a specific goal. In this approach, the user (e.g., a domain expert) can be asked to label an instance, which may be from a set of unlabeled instances.

Some of the aforementioned methods have weaknesses which include false positives, low prediction precision, high computational time and false negatives. For these reasons, efforts continue to evolve on how to improve on the techniques.

However, among the aforementioned tools, HMMs, which is an hybrid technique between time series analysis and probabilistic prediction techniques, have been proved to be very promising for anomaly prediction over several other techniques. This is due to their high accuracy in identifying attacks (Badajena *et al.*, 2012). However, research has shown that the efficiency of HMM-based algorithms is hindered by long training time during model construction (Sendi *et al.*, 2012). This study aims at overcoming this limitation by employing Variational Bayesian Inference (VBI) in optimizing the HMM learning algorithm.

1.3.4 Nature of Distributed Denial of Service (DDoS) Attack

DDoS progresses in stages and can therefore be said to have different phases. According to the experiments run by the MIT Lincoln Lab (MIT Lincoln Lab, 2000), DDoS attack session can be grouped into five phases as follows: (Lee *et al.*, 2008)

1) IPsweep to the DMZ (demilitarized zone) hosts from a remote site.
2) Probe of live IP's to look for the sadmind daemon running on Solaris hosts.
3) Breaks-in via the sadmind vulnerability, both successful and unsuccessful on those hosts.
4) Installation of the Trojan mstream DDoS software on three hosts in the DMZ.
5) Launching the DDoS.

Also, Lee *et al.* (2008) identified nine features that could be used in analyzing the characteristics of the network during a DDoS attack. The features are:

i) Entropy of source IP address,
ii) Entropy of source port number,
iii) Entropy of destination IP address,
iv) Entropy of destination port number,
v) Entropy of packet type,
vi) Occurrence rate of ICMP (Internet Control Message Protocol)
vii) Occurrence rate of UDP (User Datagram Protocol)
viii) Occurrence rate of TCP-SYN (Transmission Control Protocol-Synchronised)), and
ix) Number of packets.

While the work of Lee *et al.* (2008) identified these features and claimed that they could be studied to have an insight into the network state, it never built a model in which they were used.

The work of Afolorunso *et al.* (2016) established that these network traffic features can be taken as observable events and used to predict the states of the system and what could happen in the system in the foreseeable future.

1.4 Statement of the Problem

Global interconnectivity of systems has brought about vulnerability of inter-networked systems. Equally, advancement in technology and the new paradigms, Internet of Things (IoT) and Internet of Everything (IoE), have brought about new techniques of attacks and new trends of system vulnerability. The convolution of all these dictated the very urgent need for developing new techniques of system security. Recent happenings in the world have practically demonstrated, also, that most warfare such as industrial espionage, terrorism, international political fight for supremacy and control, and so on are now been perpetrated via the internet. Examples of such incident is the likes of the attack suspected to had been launched to interfere in the last United States and even the French elections held in May, 2017. There is also the current issue of the global cyber attack Ransomware that is locking the user's keyboard or computer thereby preventing the individual users and large organisations from accessing their data until a certain ransom is paid. These are pointers to the fact that there is need for more researches to come up with new proactive and efficient techniques for predicting and pre-empting cyber attacks and securing network systems. Several researchers such as Jemili *et al.* (2009), Kim *et al.* (2010), Pontes (2012), Sendi *et al.* (2012), Shin *et al.* (2013), Saini *et al.* (2014), amongst others have worked on attack prediction systems. Specifically, the problems identified by these research works are:

i) non-incorporation of the entropy-based features of a network traffic as the observable states of an HMM-based model.

ii) high-dimensionality of features of the intrusion specific dataset to be used as observable states together with the long training time of HMM algorithms (Lee *et al.*, 2008; Sendi *et al.*, 2012; Shin *et al.*, 2013 and Sharma and Gupta, 2015). A system with too many parameters becomes intractable computationally and expensive to implement.

iii) convergence of the traditional HMM training algorithm (Baum-Welch algorithm) to local minima or maxima, as the case may be (Rodrıguez and Torres, 2003; Xiao *et al.*, 2007; Thanthrige *et al.*, 2016). Since a globally optimal solution is a feasible solution with an objective value that is as good or better than all other feasible solutions to the model, the output of a model that converges to local maxima will be unreliable.

Several literature such as the works of Kim *et al.*, 2008; Sendi *et al.*, 2012; Shin *et al.*, 2013 and Thanthrige *et al.*, 2016 has indicated that the main problem of handling intrusion in networks is timely and precise prediction of attacks. Though, many techniques had been deployed in solving this problem, this study presents a novel approach for overcoming some of these challenges identified above

1.5 Aim and Objectives of the Study

1.5.1 Aim of the Study

The aim of this work is to develop a fast and high precision computational algorithm for the prediction of Distributed Denial of Service (DDoS) attack in network systems.

1.5.2 Objectives of the Study

The specific objectives of this research are to:

1. formulate an entropy-based state-space HMM for DDoS attack prediction
2. reduce the parameter space of the HMM in order to achieve a computationally efficient model for predicting DDoS attacks in network systems
3. simulate a globally convergent algorithm for training the formulated HMM in order to reduce its complexity.

1.6 Scope and Limitation of the Study

This work has covered the following:

i) Entropy-based HMM development
ii) HMM parameter space reduction
iii) HMM training and attack prediction
iv) The study employs simulation approach

However, only DDoS attack rather than general network attacks was considered in this study

1.7 Significance of the Study

Increase in the level of interconnectivity of organisations across the globe has led to corresponding increase in system vulnerability. This calls for new methods of securing systems. Due to global connectivity in the cyber space, a DDoS attack can span many continents. An example of this is the

recent DDoS attack on domain name system company, Dyn, of 21st October 2016 that disrupted internet services across Europe and the United States of America (Hilton, 2016). This work will go a long way in opening a new line of research into network intrusion prediction as well as provide an effective method of predicting and thereby reducing cyber crimes, terrorist attacks, industrial espionage and breaches of application databases.

1.8 Operational Definition of Terms

The major definitions used in the course of this study are as defined below:

Clustering is the process of partitioning a group of data points into a small number of clusters

Computational Time is the total running time of the model, which is the time it takes to formulate and train the model plus the prediction time.

Denial of Service (DoS) attack is one in which an attacker makes network resources too busy to serve legitimate requests.

Dirichlet distribution, named after the 19th century Belgian mathematician Johann Dirichlet and often denoted by $Dir(\alpha)$, is a family of continuous probability distributions parameterized by a vector α of positive real numbers. It is a generalization of the beta distribution. Dirichlet distributions are very often used as prior distributions in Bayesian statistics, and it is actually the conjugate prior of the multinomial distribution and categorical distribution. The infinite-dimensional generalization of the Dirichlet distribution is the **Dirichlet process**.

Distributed Denial of Service (DDoS) attack can be defined as a large-scale, coordinated attack on the provision of services of a victim system or network resources, launched indirectly through a large number of compromised computer agents on the internet.

Entropy is the measure of uncertainty in a given information or the measure of the amount of information that is missing in a transmitted message before reception . Unlike many other functions of state, entropy cannot be directly observed but must be calculated. It is expressed in terms of discrete set of probabilities p_i such that:

$$H(Y) = -\sum_{i=1}^{n} p(y_i)\log p(y_i)$$

False negatives are intrusive activities that are wrongly classified/identified as normal activities by the network security device.

False positive/alarm is any normal or expected or non-intrusive activity that is identified/classified as anomalous or malicious or intrusive by the network security device..

An **Intrusion Detection System** (**IDS**) is a passive device or software application that monitors, gathers and analyzes information from various areas within a computer or network system in order to identify possible malicious activity or policy violations, which may include both intrusions (attacks from outside the network) and misuse (attacks from within the network). It reports such activity(ies) to an administrator.

Intrusion Forecasting Systems (IFS)/Intrusion Prediction System (IPrS) are proactive ways of handling intrusion in networks unlike IDS which responds to Internet attacks after serious damage had been done or IPS which slows down the network in the case of heavy network traffic. They

deploy several techniques to predict/forecast the occurrence of intrusion in a system. Thus allowing for time to action against such occurrence.

An **Intrusion Prevention System (IPS)** is an active preemptive network security/threat prevention technology that examines network traffic flows to detect and prevent vulnerability exploits. Though its function is similar to that of an IDS, IPS has the additional ability to take immediate action, based on a set of rules established by the network administrator.

K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

Network intrusion/attack is any unauthorized activity on or against a network system, which might be from within or without the network.

Parsimonious system is a system that does not sacrifice comprehensiveness at the expense of simplicity and the goal of such a system is to create simple models that gives the highest performance.

Receiver Operating Characteristic (ROC) curve, in Statistics, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. It is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

Relative entropy, also known as the Kullback-Leibler divergence (KLD), between two probability distributions on a random variable is a measure of the distance between them. Formally, given two

probability distributions $p(x)$ and $q(x)$ over a discrete random variable X, the relative entropy given by D_{KL} $(p||q)$ is defined as follows: $D_{KL}(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$.

In this definition, $0 \log \frac{0}{0} = 0$, $\log \frac{0}{q} = 0$, and $p \log \frac{1}{0} = \infty$.

State-space representation is a mathematical model of a physical system as a set of input, output and state variables related by first-order differential equations. "State space" refers to the Euclidean space in which the variables on the axes are the state variables.

Traffic anomaly is deviation from the normal traffic pattern. It can be used to identify unknown attacks and Denial of Service (DoS) floods.

Trellis is a structure that records the probability of all initial subpaths of a Hidden Markov model that end in a certain state at a certain time.

True positives are anomalous, malicious or intrusive network activities that are correctly classified or identified as such.

True negatives are normal or innocuous network activities are correctly classified or identified as such.

Victim is a network device that is under attack.

1.9 List of Abbreviations and Acronyms

ACC: Prediction Accuracy

AFRL: Air Force Research Laboratory

CAIDA: Center for Applied Internet Data Analysis

CT: Computational Time

DARPA: The Defense Advanced Research Projects Agency

DBN: Dynamic Bayesian Network

DDoS: Distributed Denial of Service

DMZ: Demilitarized Zone

DoS: Denial of Service

ELBO: Evidence Lower Bound

EM: Expectation Maximisation

FN: False Negative

FM: F-Measure

FP: False Positive

HMM: Hidden Markov Model

ICMP: Internet Control Message Protocol

ID: Intrusion Detection

IDS: Intrusion Detection System

IFS: Intrusion Forecasting System

IoE: Internet of Everything

IoT: Internet of Things

IPrS: Intrusion Prediction System

IP: Internet Protocol

KDD: Knowledge Discovery in Databases

KLD: Kullback-Liebler Divergence

ML: Maximum Likelihood

PR: Precision

R2L: Remote to Local

ROC: Receiver Operator Characteristics

TCP: Transmission Control Protocol

TCP-SYN: Transmission Control Protocol SYNchronise packet

TN: True Negative

TNR: True Negative Rate

TP: True Positive

TPR: True Positive Rate

U2R: User to Root

UDP: User Datagram Protocol

VBI: Variational Bayesian Inference

CHAPTER TWO

LITERATURE REVIEW

This chapter presents a critical review of existing approaches to network attacks detection and prediction as well as nature and pattern of DDoS attacks. It also showcases major concepts used in this study.

2.1 Distributed Denial of Service (DDoS) Attack

Prasad *et al.* (2014) defined distributed denial of service (DDoS) attack as a large-scale, coordinated attack on the provision of services of a victim system or network resources, launched indirectly through a large number of compromised computer agents on the internet. Prior to launching an attack the attacker takes control of large number of vulnerable computer machines over the internet. The attacker exploits the weaknesses of these computers by inserting malicious code or some other hacking technique to get under his control. The compromised machines, which can run into thousands in numbers, are commonly referred to as ‘zombies.’ The group of zombies usually formed the ‘botnet.’ The magnitude of attack is dependent on the size of botnet. For larger botnet, attack is more severe and disastrous.

DDoS is a malicious act in which an attacker floods the victim with several requests that consume the system's resources so that the victim cannot satisfy legitimate requests. This class of attack aims at undermining the availability of the internet. According to Ankali *et al.* (2011), "availability" means guarantee of reliable access to the information system by authorized persons. Information systems and the internet only have values if the right people can access it as at when needed. Nowadays, denial of access to internet facilities through DDoS has become a very common attack.

The techniques for launching DDoS attack had been evolved over the years but the general attack model and process remain unchanged. In DDoS, the attacker first chooses as many handlers as possible that have security vulnerabilities, intrudes them by gaining access rights. In a likewise manner, agents or zombies are selected indirectly through the handlers. Both the handlers and agents are usually located in networks external to both the attacker and the victim. After the successful selection of agents and handlers, through scanning using ICMP, the attacker controls secured communications among the three systems to compromise attack. The message for exchange of information are usually encrypted. The agents generates specific types of DDoS attack traffic among ICMP, TCP, and UDP type. That is, in a DDoS attack, the victim system is seriously bombarded with specific types of traffic heading for it. The agents randomly generate the source IP addresses of attack packets in order to hide their real addresses. Depending on the attack type, the destination and source port numbers are also randomized.

According to Lin *et al.* (2004), two ways exist for paralyzing a network:

i) Sending a huge number of malicious packets (such as UDP and ICMP flood attacks) towards a victim
ii) Making use of the vulnerabilities of network protocol. e.g. TCP-SYN flooding attack uses the connection characteristic of three-way handshaking of TCP protocol

The general architecture of DDoS attack is shown in Figure 2.

From the foregoing, it can be seen that depending on the particular phase of the DDoS attack there are peculiar types of features of traffic as enumerated in Section 1.3.4 that can be observed, which can be used as parameters for the detection and prediction of the DDoS attack.

In this work, the entropy-based values of the features listed in Section 1.3.4 are used as observable states of an HMM.

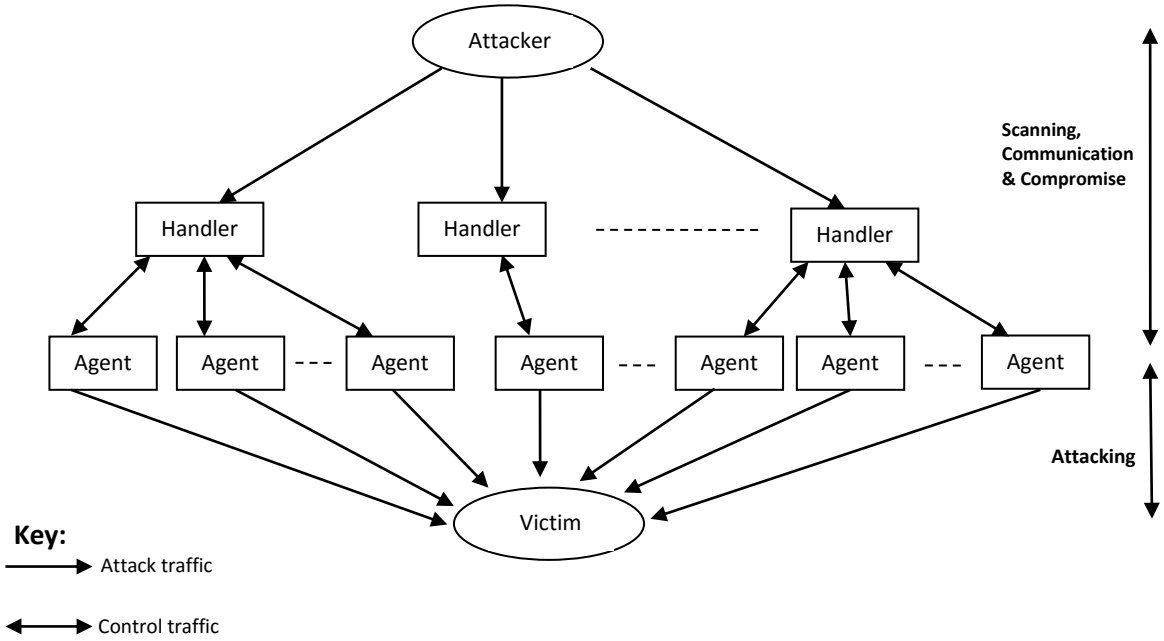


Figure 2: General architecture of DDoS attack (Source: Lee *et al.*, 2008)

As DDoS attacks are becoming one of the most threatening security issues, the need to predict and by so doing, guide against this type of attack is increasing. According to Alenezi *et al.*,(2012), DDoS is not just a "game" played for fun by some attackers, it has become an effective weapon for cyber war, international/political fight for supremacy and for so called "hacktivist" and/or terrorists groups

2.2 Review of Existing Works on Intrusion Prediction Systems

In order to guarantee the continuous availability of the resources of a network system, it is important to employ a proactive tool to handle DDoS attacks in such systems. Majority of current

tools are reactive in that they only detect such attacks after the attack had occurred and the damage done.

Intrusion prediction system has been an active area of both research and development efforts for years. A representative set of the major efforts toward intrusion prediction/forecasting systems are presented here, excluding the body of work for DDoS prediction which will be presented in Section 2.3.

In order to overcome the problem of detecting intrusions after they had taken place, Ramasubramanian and Kannan (2004) developed a framework for a statistical anomaly prediction system using ensemble Quickprop neural network forecasting model. The trained model predicts unauthorized invasions of user based on previous observations and takes further action to nip future intrusion against the system in the bud. The work focused on detecting significant changes of transaction intensity for intrusion pre-emption. A comparative evaluation of the two ensemble networks over the individual networks was carried out using mean absolute percentage error on a prediction dataset. Although the experimental results show a better prediction accuracy, the two neural network methods failed to provide high prediction confidence. They concluded that there is need to incorporate advanced intervention analysis to improve the prediction and reduce false negative alarm rates.

Chung *et al.* (2006) worked on the design of an online intrusion forecast system using a weather forecasting model, which allows network administrators to minimize the effects of damages in advance through an online intrusion prediction of the probable attacks. The information from the sensors of information security control systems and the profiles of the information system assets are used to analyze network vulnerabilities and also predict intrusion directions and the magnitude of damages. The probability of forecast was calculated based on the intensity of the threats and

occurrence frequency prediction. However, the work was unable to determine the intrusion probabilities threshold.

Jemili *et al.* (2006) worked on distributed intrusion detection and forecasting multiagent system using possibilistic network (DIDFAST.PN) which comprises of a system architecture of two interconnected layers of intelligent agents. The first layer is concerned with intrusion detection. On each host of a distributed computers system, an intelligent agent using possibilistic network is charged by detecting intrusion eventuality. The second layer is based upon one intelligent agent, which is charged by intrusion forecasting task based on possibilistic network prediction.

Agents of these two layers communicate using messages. When new intrusion is detected on the first layer, the agent responsible for this host informs the forecasting agent placed in the second layer. This latter computes conditional possibilities of intrusion appearance on each host of the distributed system, and informs the administrator of the concerned host about possible ultimate intrusion. It demonstrated a high performance when detecting intrusion but the forecasting performance was low.

Kannadiga *et al.* (2007) proposed E-NIPS (Event-based Network Intrusion Prediction System) for attack detection as well as prediction of probable attacks. The work utilized network penetration scenarios partitioned into several phases depending on the sequences followed during network penetrations. Each of the phases consists of attack classes which are precursors to attack classes of the next phase where an attack class comprises of a set of attacks with the same objectives that are categorized to generalise network penetration scenarios and reduce the burden on the prediction engine during intrusion alerts correlation and prediction exercise. Future attacks prediction are dependent on the attack classes detected in a penetration scenario's earlier phase. Experiments were

carried out using Defense Advanced Research Projects Agency (DARPA) 1999 dataset and most common occurrences of network penetration scenarios in that dataset. The experimental results showed that not only can the model predict attacks with reasonable accuracy, it was also able to estimate the penetration extent. The authors concluded that the model is not able to predict attacks that were not programmed as rules in the intrusion prediction engine. Also, the need for the development of a time-based engine to predict the likely occurrence time of attack events was emphasized.

Kim *et al.* (2008) proposed a hybrid intrusion forecasting system framework for an early warning system. The system utilizes three types of forecasting methods: time-series analysis, probabilistic modelling, and data mining method. Combining these methods makes it possible to take advantage of each forecasting technique while overcoming their drawbacks. Experimental results show that the hybrid intrusion forecasting method outperforms each of the three forecasting methods. The results also showed that among the three hybrid methods, the combination of the time-series analysis and the Markov chain method shows best performance in reducing the false alarm rate.

They concluded that there is need to develop new intrusion forecasting methods that provide improved accuracy of predictions with a lower false alarm rate as well as developing an alert correlation algorithm between each forecasting method in order to achieve earlier and more precise forecast of attacks.

Haslum *et al.* (2008) proposed an HMM for sensing intrusive activities in a distributed environment and make a one-step ahead prediction against possible intrusions. Activation of the prediction system is dependent on the predicted threat level and risk assessment of the resources. Intrusions attempts are blocked at the occurrence of any of the following: a serious attack that has already occurred, high packet flow rate, possible serious attack prediction and lastly, online risk assessment

of the possible assets available to the attacker. Experimental results from the systems show that the proposed framework is efficient for real-time distributed intrusion monitoring and prevention. However, the work only modelled integrity and confidentiality properties of network systems without any attempt to model availability. They believed and opined that availability is best modelled separately. They also concluded on the need to formulate better HMM with more states as well as incorporate parameter estimation based on real data.

Ahmadi (2009) introduced a new detection and prediction system using cooperative co-evolutionary immune system (CoCo-IDP) for distributed data networks based on genetic algorithm using grid computational technique in a distributed environment. The approach is an intelligent technique that integrated the immune system and cooperative co-evolutionary concept using an agent based grid computing in a distributed environment as a solution for tracking and predicting the unwanted security threats in a large scale data networks. The model detectors is able to discriminate existing incidents and predict new incidents in a distributed environment. The model was implemented using KDD dataset in a Jini platform running grid computing in distributed systems. The four classes of attack in KDD dataset was used in training the system. Evaluation results show that, the CoCo-IDP can adaptively converge for the best answer and can detect or predict the incidents in a selected boundary. Moreover, the system generates the flexible detectors with diversity in a variable threshold. Based on the experimental results, the proposed system in comparison with pure Immune System, has simpler rules, more powerful detection and prediction capabilities with high accuracy metric. The probability of detection and false accuracy rate compared in KDD database with several well known methods for proof and validation of the obtained results show the need to improve on the false positive rate of predictions.

Pointes *et al.* (2009) in an effort to overcome the post-mortem approaches of IDS examined the applicability of a cooperative architecture for forecasts of Unwanted Internet Traffic (UIT) on a more complex set-up, with hosts associated with sites that are geographically divided. The work formulated a collaborative architecture of IDS with prediction approaches that cover the gaps of the existing forecasting techniques as regards UIT by making use of five forecasting techniques: simple moving averages (SMA), exponential weighted moving averages (EWMA), combined SMA, combined EWMA and Fibonacci sequence. Experiments were carried out as a prototype that employs diverse sensors and forecasting techniques in a cooperative manner and able to predict UIT. A proof of concept of such architecture was presented, which allows concluding about the improvement in forecasts for IDS to cope with Unwanted Internet Traffic (UIT) and can therefore be employed as an early warning system. It was concluded that the value of the outcome of the system was questionable as it has 34.74% of warnings being reported late. The authors mentioned the need to improve the forecast precision through the use of other forecasting techniques such as Markov chains, HMM, neural networks.

Jemili, *et al.* (2009) proposed an intrusion detection and prediction system (HIDPAS) based on uncertain and imprecise inference networks and its implementation. The work presented an approach based on hybrid propagation that combines probability and possibility, through a Bayesian network that provides automatic learning from audit data, to identify attack plans and predict upcoming attacks. The model employed a supervised learning through the application of graph techniques based on Bayesian reasoning .The Bayesian network based system correlates attack scenarios based on their relationships. Inference was conducted to evaluate the likelihood of attack goal(s) and predict potential attacks based on the hybrid propagation of uncertainties. The application of the model in intrusion detection context helps in the detection of both normal and abnormal connections with very considerable rates. The use of Bayesian networks and the hybrid

propagation within Bayesian networks, which is especially useful when dealing with missing information system, resulted in demonstration of high performance in intrusions detection, correlation and prediction of attacks. The results of the experiments carried out using KDD'99 dataset show that the system could not recognise attack plans and could only work in a static environment. It was concluded that the system could further be improved by integrating an expert system which is able to provide recommendations based on attack scenarios prediction.

Khosronejad *et al.* (2013) worked on a hybrid approach of new techniques for modelling IDS. The work combined C5.0 and HMM to form a hierarchical hybrid intelligent system model. Though experimental results of the hybrid C5.0-HMM approach with KDD Cup 99 benchmark Intrusion data showed that this hybrid system provide more accurate intrusion detection compared to ordinary HMM approach, it could not achieve prediction.

Divya and Muniasamy (2015) proposed an hybrid framework, which combined two machine-learning techniques, HMM and genetic algorithm (GA) for predicting future intrusion attacks in network systems. The model comprises of two major components: the first component makes use of GA to derive efficient intrusion detection rules after which a precise attacks detection is made, the other component uses HMM in predicting the attacker's next attack class. Though the combination of these two gives a good intrusion prediction capability and reduced false positive rate, there was need to improve the false negative rate.

Thanthrige *et al.* (2016) proposed an HMM-based alert prediction framework. Alert clustering was employed to group selected alert attributes together. A given sequence of alerts is converted to a sequence of alert clusters and then HMM is used to predict future alert clusters based on the input. Experimental results show good performance. However, they identified the following challenges to be addressed in the proposed alert prediction framework: (1) increasing the prediction accuracy with

the increase of cluster size and predicting intrusion types that are not present in the training data set, and (2) identifying false alerts and misleading intrusion actions generated by the attacker in order to mislead intrusion detection systems.

2.2.1 Summary of the Review of Existing Works on Intrusion Prediction Systems

Table 1 presents the summary of related works on Intrusion Prediction Systems based on the approach employed, the strengths and the weaknesses inherent in the works

Table 1: Summary of Existing Works on Intrusion Prediction Systems

Author	**Approach**	**Strengths**	**Weaknesses**
Ramasubramanian and Kannan (2004)	Ensemble Quickprop neural network forecasting model	Average prediction accuracy	Low prediction confidence
Chung *et al.* (2006)	Online intrusion forecasting using a weather forecasting model	Threat intensity determines forecast probability and accuracy	Failure to determine intrusion probability threshold
Jemili *et al.* (2006)	two interconnected layers of intelligent agents for detection and forecasting of distributed intrusion	High performance in intrusion detection	Low forecasting performance.
Kannadiga *et al.* (2007)	Event-based network intrusion prediction system for attack detection and prediction using DARPA 1999 datasets	Reasonable prediction accuracy and penetration extent estimation	Only able to predict preprogrammed attacks
Kim *et al.* (2008)	Hybrid approach that combined time-series analysis, probabilistic modeling (Markov Chain model) and data mining method	Improved false alarm rate compared to each of the method combined	The forecasting precision was low
Haslum *et al.* (2008)	an HMM for sensing intrusive activities in a distributed environment and make a one-step ahead prediction against possible intrusions	efficient for real-time distributed intrusion monitoring and prevention	The work did not model the availability property of the system
Ahmadi (2009)	Distributed intrusion	High prediction	High false positive

	detection and prediction system using cooperative co-evolutionary immune system (CoCo-IDP) for distributed data networks based on genetic algorithm using grid computational technique.	capabilities	rate
Pointes *et al.* (2009)	use of five forecasting techniques: simple moving averages (SMA), exponential weighted moving averages (EWMA), combined SMA, combined EWMA and Fibonacci sequence	Useful tool as an early warning system able to cope with forecast of Unwanted Internet Traffic (UIT)	Low forecasting precision and tendency for late reporting of warning
Jemili, *et al.* (2009)	Hybrid propagation that combines probability and possibility, through a Bayesian network that provides automatic learning from audit data, to identify attack plans and predict upcoming attacks. Experiments with KDD'99 dataset	High performance in intrusion detection and prediction of attacks.	The model could only work in a static environment
Khosronejad *et al.* (2013)	Hybrid approach that combined C5.0 and HMM to form a hierarchical hybrid intelligent system for modelling IDS. Experiments was based on KDD Cup 99 dataset	More accurate intrusion detection	The model could not achieve intrusion prediction
Divya and Muniasamy (2015)	Hybrid framework, which combined HMM and genetic algorithm (GA) for predicting future intrusion attacks in network systems	Good prediction capability with reduced false positive rate	Need to improve the false negative rate
Thanthrige *et al.* (2016)	an HMM-based alert prediction framework	Good prediction performance	Unable to identify false alert and misleading intrusion actions

2.3 Review of Existing Works on DDoS Prediction Systems

In this section various existing DDoS prediction relevant works are reviewed to bring out the specific gaps in current researches.

In an effort to improve on existing methods of DDoS attack detection, Peng *et al.* (2004) proposed a simple robust scheme to proactively detect DoS attacks as well as DDoS attacks by monitoring the increase of new IP addresses. Unlike previous proposals for bandwidth attack detection schemes, which are based on monitoring the traffic volume that is not able to differentiate flash crowds from DDoS attacks, the scheme is very effective for highly distributed DoS attacks. The scheme exploits an inherent features of DDoS attacks that made it difficult for the attackers to counter the detection scheme by changing their attack signature. The model employed a sequential non-parametric change point detection technique (Cumulative Sum (CUSUM) algorithm) to improve the detection accuracy, as well as reduce false alarm rate, without need of a detailed model of normal and attack traffic. It was shown that through the combination of monitoring per flow speed, all types of DDoS attacks can be detected. The efficiency and robustness of the model was demonstrated through the use of trace-driven simulations. The simulation experiments results, in the Aukland traces, showed that the model is very effective in cases where there are multiple attack sources and highly distributed attack traffic. Also, the results demonstrated that high detection accuracy can be achieved on a range of different network packet traces. It was concluded that incorporation of other network traffic features into the model would yield better results.

Lee *et al.* (2008) also worked on a proactive detection of DDoS attacks by exploiting DDoS architecture, as described earlier in Section 2.1, using cluster analysis. They looked into the procedure of DDoS attack and thereby constructed nine features that could be used in analyzing the behaviour of the network during a DDoS attack. Quantifying the randomness and occurrence of

several fields in the TCP header made it possible for the nine constructed features to excellently reveal the abnormal pattern in the network data. Cluster analysis was then performed to proactively detect DDoS attacks. To evaluate the method, experiments were carried out using DARPA 2000 Intrusion Detection Scenario Specific dataset. Experimental results showed that the method was able to partition well the various DDoS attack phases and detect precursors of the DDoS attack as well as the attack itself. However, out of the five phases of the attack, the method could only detect three phases. Also, the entropy-based values of the constructed features were only used intuitively.

Rao *et al.* (2012) convinced that application layer based attacks make use of genuine HTTP requests to create devastating effect on networks resources which in turn results into DoS, applied HMM to monitor Application Layer DDoS attacks on web servers. The model was trained using complex algorithms of application of forward-backward algorithm to train HMM model thereby increasing the response time of the application. It was noticed that slight customisation of website design helps in minimising Application layer DDoS attacks. However, the work is limited to monitoring Application layer DDoS attacks.

Sendi *et al.* (2012) designed a proactive intrusion response system that can predict different kinds of DDoS attacks mostly minutes before it happens. The authors proposed an HMM architecture to predict intrusions and trigger good response strategies. HMM was used to extract the interactions between networks and attackers. A novel alert correlation was employed in decreasing false negatives in the prediction. Experimental results on the DARPA 2000 dataset showed that the model can predict DDoS attacks and has a potential to detect multi-step attacks missed by the detection component but the false positive rate was high.

Kwon *et al.* (2012) proposed a proactive DDoS attack forecasting system architecture, named Internet Intrusion Forecasting System Architecture (IIFSA), to overcome the limitations of the

reactive IDS systems. The IIFSA was designed by conducting analysis from physical, technical and informational perspectives. To obtain intrusion factors for DDoS attack forecasts, Honeynet was deployed and Hflow data gathered from Honeynet was analyzed. The Honeynets were deployed in Pohang University of Science and Technology, Pohang in Republic of Korea to collect the raw data necessary to forecast DDoS attacks. The Hflow data gathered from the Honeynets were analyzed as a first step to estimate intrusion factors. As the forecasting method, regression analysis based on its reported efficacy for the specific analysis (Schechter, 2005) was deployed as a single algorithm for the Statistical Analyzer of the Intrusion Forecasting Module. It was reported that in order to improve the accuracy of forecasts, there is need to combine several forecasting methods.

Shin *et al.* (2013) proposed a probabilistic approach to effectively forecast and detect network intrusions. The approach uses a Markov chain for probabilistic modelling of abnormal events in network systems. *K*-means clustering was performed to define the network states, with the introduction of the concept of an outlier factor. The performance of the approach was evaluated by experiments using the DARPA 2000 dataset. Experimental results showed that the model achieves some level of high detection performance However, although in the work, the various features of the DDoS attack and the concept of entropy were mentioned they were not incorporated into the model formulation. Also, it was concluded that there was need to improve the prediction accuracy through the use of combination of some probabilistic techniques.

Berezinski *et al.* (2015) using data mining techniques proved that an entropy-based approach is suitable to detect modern botnet-like malware based on anomalous patterns in the network. A dataset based on real and legitimate traffic with synthetic anomalies was developed to evaluate the model. The dataset, which was built from one day legitimate traffic profile representative of and suited to any regular working day in the network, contains labelled flows that are stored in a

relational database. Using results of features correlation and classification of different anomalies in tested scenarios, they also concluded that broad spectrum of network traffic feature is essential for detecting and classifying different kinds of anomalies.

2.3.1 Summary of the Review of Existing Works on DDoS Prediction Systems

Table 2 presents the summary of related works on DDoS Prediction Systems based on the approach employed, the strengths and the weaknesses inherent in the works

Table 2: Summary of Existing Works on on DDoS Prediction Systems

Author	**Approach**	**Strengths**	**Weaknesses**
Peng *et al.* (2004)	Cumulative Sum (CUSUM) algorithm was employed in monitoring the increase of new IP addresses in order to proactively detect DoS and DDoS attacks.	Effective in multiple attack sources and highly distributed attack traffic situations	Low performance due to incorporation of few network traffic features in the model
Lee *et al.* (2008)	Employed the concept of entropy and cluster analysis to detect DDoS using DARPA 2000 dataset for experiments	each phase of the attack scenario was well-partitioned and detection of precursors of DDoS attack was achieved	The work was only able to detect three phases of the attack and failed to build a model with the entropy-based values
Rao *et al.* (2012)	applied HMM to monitor Application Layer DDoS attacks on web servers	Increase in applications response times	limited to monitoring Application layer DDoS attacks only
Sendi *et al.* (2012)	Deployed HMM architecture in predicting intrusions using DARPA 2000 dataset as experimental data	Low false negative rate	High false positive rate
Kwon *et al.* (2012)	Deployed regression analysis and Honeynets in Pohang University of Science and Technology, Pohang in Republic of Korea to collect the raw data necessary to	Efficient forecasting	Low forecasting accuracy

	forecast DDoS attacks		
Shin *et al.* (2013)	Deployed Markov chain for probabilistic modeling of abnormal events in network systems using the DARPA 2000 dataset for experiments	achieved high detection performance while representing the level of attacks in stages	Non-usage of the various features of the DDoS attack and the entropy-based values as well as need to improve prediction accuracy.
Berezinski *et al.* (2015)	Deployed data mining techniques to demonstrate that an entropy-based approach is suitable to detect modern botnet-like malware. It compared the results for several popular classifiers.	The false positive rate was low	The spectrum of network traffic features used was narrow

From the afore discussions, it could be seen that earlier works have pointed out the need to model the availability property of network resources (Haslum, 2008). The work of Peng *et al.* (2004) established the need to incorporate more network traffic features statistics into DDoS attack prediction to achieve better accuracy. Also, though some of the existing works (Lee *et al.*, 2008; Shin *et al.*, 2013; Berezinski *et al.*, 2015) mentioned and even evaluated the entropy-based values of the network traffic features, these quantities were never used to build a model for predicting DDoS attack. To the best of the author's knowledge, the phases of the DDoS attack is another aspect that has never been incorporated into the formulation of a model for DDoS attack prediction. Finally, the existing methods, suffer from low prediction precision, high false positive and false negative rates as well as long computational time and the works of Kim *et al.* (2008), Shin *et al.* (2013), amongst others, pointed out that there is need combine and incorporate more probabilistic and time series techniques into DDoS prediction for more accurate results. This work, therefore, is aimed at bridging some of these gaps in literature.

2.4 Theories and Concepts Used in the Study

A prediction of network intrusion is generally viewed as a pattern recognition problem, and as such, it can be solved using one of two broad approaches: structural and empirical (Kumar and Ravi, 2007). The former derives the probability of a network for default based on its characteristics and dynamics, while the latter approach relies on previous knowledge and relationships in the area under study, learns from existing data or experience, and deploys statistical or intelligent methods to predict intrusion.

The most popular statistical techniques are linear discriminant analysis, multivariate discriminant analysis, quadratic discriminant analysis, logistic regression, and factor analysis. Whereas the most common intelligence techniques are neural networks, decision trees, case-based reasoning, evolutionary approaches, Bayesian techniques, etc.

According to Lacher *et al.* (1995), Vellido *et al.* (1999) and Gyanchandani *et al.* (2012), as cited in Kumar and Ravi, (2007), traditional statistical techniques have often been criticized because of their assumptions about linear separability of training data, multivariate normality, and independence of the predictive variables. Such constraints are incompatible with the complex nature, boundaries, and interrelationships of most of network activities used for learning and prediction. The intelligent techniques have shown themselves more appropriate for the task. For instance, neural networks do not rely on a-priori assumptions about the distribution of data and work well in unstructured, uncertain and noisy environments.

Over the years, several statistical techniques such as exponentially weighted moving average (EWMA), Fibonacci Sequence and Markov Chains, CUmulative SUM (CUSUM) algorithm, data mining, etc. have been employed in the development of intrusion forecasting systems.

The VBI-HMM that is employed in this work falls under intelligent (machine learning) methods.

2.4.1 Entropy

The definition of entropy as a measure of disorder or disorganisation is from thermodynamics and was proposed in the early 1850s by Clausius *et al*.,(1867) (as cited in Berezinski *et al.*, 2015*)*. Shannon (1948) adopted entropy to information theory as a measure of the uncertainty associated with a random variable. The more random the variable, the bigger the entropy and vice versa. For a probability distribution $p(Y = y_i)$ of a discrete random variable Y, the Shannon entropy is defined as:

$$H_s(Y) = -\sum_{i=1}^{n} p(y_i)\log_a p(y_i) \quad (1)$$

Y is the feature that can take values $\{y_1 \ldots y_n\}$ and $p(y_i)$ is the probability mass function of outcome y_i. The entropy of y can also be interpreted as the expected value of $\log_a p(y_i)$

where Y is drawn according to probability mass function p(y). Different units such as bits (a = 2), nats (a = e) or hurtleys (a = 10) can be used depending on the base of the logarithm. For network attack detection or prediction, typically sampled probabilities estimated from a number of occurrences of y_i in a time window t are used. The value of entropy depends on randomness and the value of n. To measure randomness only, normalised forms, as used in this study, have to be employed. For instance, an entropy value can be divided by n or by maximum entropy defined as $\log_a(n)$. The algorithm for calculating normalised entropy is as presented in Algorithm 1 in Section 3.2.3.

Aczel and Daroczy (1975) and Karmeshu (2003) showed that entropy as defined in equation (1) using functional analysis has the following properties:

i) Non-negativity: i.e. $\forall_{p(yi)\in[0,1]} H_s(Y) \geq 0$

ii) Symmetry: $H_s(p(y_1), p(y_2), ...) = H_s(p(y_2), p(y_1), ...)$

iii) Maximality: $H_s(p(y_1), ..., p(y_n)) \leq H_s(\frac{1}{n}, ..., \frac{1}{n}) = \log_a(n)$

iv) Additivity: $H_s(Y, Z) = H_s(Y) + H_s(Z)$, if Y and Z are independent variables.

2.4.2 Clustering

Clustering, which is considered as the most important unsupervised learning problem, deals with finding a structure in a collection of unlabelled data (MacQueen, 1967). It could be defined as the process of organising objects into groups such that members belonging to the same group are somehow similar. A *cluster* is thus a collection of objects which are similar among themselves and dissimilar to objects belonging to other clusters.

A simple graphical example is as in figure 3 in which four clusters into which the data could be divided is identified. In this example, the similarity criterion is distance i.e. two or more objects are belong to the same cluster if they are close according to a specified geometric distance. This is referred to as distance-based clustering.

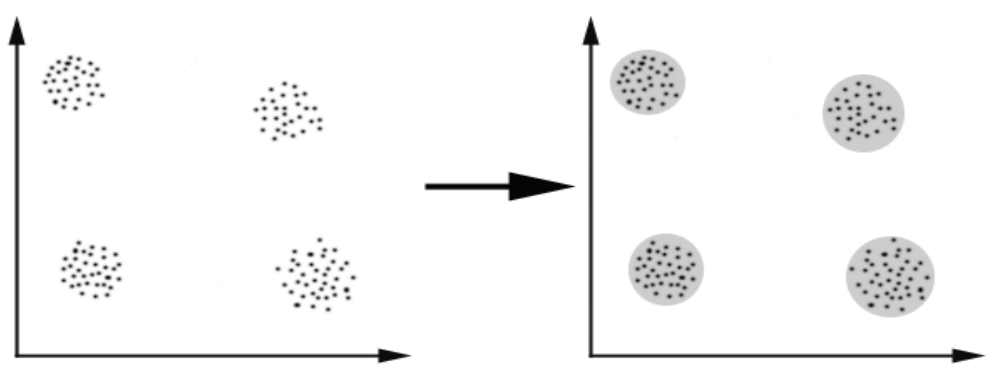


Figure 3: Clustering Example (MacQueen, 1967)

The similarity criterion could also be conceptual, in which case, two or more objects belong to the same cluster based on the definition of a common concept to all the objects, that is, objects grouping is done based on their fit to certain descriptive concepts and not simple similarity measures. This is referred to as conceptual clustering.

The goal of clustering is to determine the intrinsic grouping in a set of unlabelled data. A good clustering algorithm need to satisfy the following requirement.

i) Scalability;
ii) Dealing with different types of attributes;
iii) Discovering clusters with arbitrary shape;
iv) Minimal requirements for domain knowledge to determine input parameters;
v) Ability to deal with noise and outliers;
vi) Insensitivity to order of input records;
vii) High dimensionality;
viii) Interpretability and usability.

Clustering algorithms are classified into four viz.: Exclusive Clustering, Overlapping Clustering, Hierarchical Clustering and Probabilistic Clustering. The *K*-means clustering that is deployed in this study belongs to the exclusive clustering algorithm class. In it, data are grouped in an exclusive way, such that a certain datum belonging to a certain cluster cannot be included in another cluster.

2.4.2.1 *K*-Means Clustering

As earlier stated, *K*-Means is an exclusive clustering algorithm. It is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows an easy and simple way to classify a given data set through a certain number of clusters (assume *k* clusters) fixed a priori. The main idea in *K*-means clustering is to define *k* centroids, one for each cluster. These centroids are placed in a cunning way because different location causes different result. Therefore, the better choice is to place them as far away from each other as possible. Next, each point belonging to a given data set is taken and associated to the nearest centroid. In the absence of any pending point, the first step is completed and an early groupage is done. Then *k* new centroids re-calculate as barycenters of the clusters resulting from the previous step. After the re-calculation of these *k* new centroids, a new binding is done between the same dataset points and the nearest new centroid, which results in the generation of a loop. As a result of this loop it may be observed that the *k* centroids change their location step-by-step until no more changes are done. That is, centroids are not moving any more. The *K*-means algorithm aims at minimizing an *objective function*, in this case a squared error function. The objective function:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2 \qquad (2)$$

where $\left\| x_i^{(j)} - c_j \right\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centre c_j, is an indicator of the distance of the *n* data points from their respective cluster centres.

The algorithm for *K*-Means clustering is as presented in Algorithm 2 of chapter three.

2.4.3 Kullback-Liebler Divergence (KLD) or Relative Entropy

The KLD has numerous meanings in mathematics and its various applications areas in science is huge. Though it comes from the field of information theory, where it measures how much "information" is lost when coding a source using a different distribution other than the real one, it is used to compare distributions (Cover and Thomas, 2006). While, in statistics, it is commonly used as a measure of similarity between two density distributions Aczél and Daróezy (1975), in the context of information theory, it can be defined as the difference between entropies i.e. the joint entropy of *Q* and *P* and the entropy of *P*.

For discrete probability distributions *P* and *Q*, the KLD of *Q* from *P* is defined to be:

$$D_{KL}(P||Q) = \sum_{i}^{n} P(i) \ln \frac{P(i)}{Q(i)} \tag{3}$$

It can be described as the expectation of the logarithmic difference between the probabilities *P* and *Q*, where the expectation is taken using the probabilities *P*. The KLD is only defined if $Q(i) = 0 \Rightarrow P(i) = 0$, for all i (absolute continuity). If the quantity 0ln0 appears in the formula, it is interpreted as zero since $\lim_{x \to 0} x \ln(x) = 0$. It should be noted that the definition in equation (3) above is not symmetric, i.e. $D_{KL}(P||Q) \neq D_{KL}(Q||P)$ except when $P = Q$ (Cover and Thomas, 2006)

In summary, the KL divergence satisfies three divergence properties:

1. Self similarity: $D_{KL}(P||P) = 0$.
2. Self identification: $D_{KL}(P||Q) = 0$ only if $P = Q$
3. Positivity: $D_{KL}(P||Q) \geq 0$ for all P, Q.

KLD is also widely used in machine learning. Some of its application areas, amongst others, are in optimization by minimizing or maximizing the divergence between distributions, compression and Bayesian approximate inference. The work of Afolorunso *et al.* (2016) showed it can be deployed in

reducing the observable states of an HMM. As approximate inference is present in most machine learning researches, so also is KLD (Beal, 2003).

In this study, KLD is deployed in two different ways. First, it was employed in reducing the observable states of the HMM using algorithm 5 in Section 3.3. The concept is also embedded in the Variational Bayesian Inference (VBI) used in training the HMM. However, the KLD for variational inference is slightly different from the one defined above. It is as given below:

$$D_{KL}(q||p) = \mathrm{E}_q\left[\log \frac{q(Y)}{p(Y|x)}\right] \tag{4}$$

where $x = x_1, x_2, ..., x_n$ are the observations and $y = y_1, y_2, ..., y_m$ are the hidden variables

It is not easy to minimize the KLD as a function of variational distribution. However, this can be achieved by maximizing the evidence lower bound (ELBO) of the function. ELBO is obtained by applying Jensen inequality (f (E[X]) ≥ E[f(X)] when f is concave) on the log probability of the observations,

$$\log p(x) = \log \int_y p(x, y) \tag{5}$$

$$= \int_y p(x, y) \frac{q(y)}{q(y)} \tag{6}$$

$$= \log \left(\mathrm{E}_q \left[\frac{p(x,Y)}{q(y)}\right]\right) \tag{7}$$

$$\geq \mathrm{E}_q [\log p(x,Y)] - \mathrm{E}_q [\log q(Y)] \tag{8}$$

Equation (8) is the ELBO and it is the same bound used in deriving the EM algorithm (See Ibe, 2013). A family of variational distributions is chosen to make the expectations computable. In this study, Dirichlet distribution is chosen because it is the conjugate to the complete-data likelihood terms of the HMM (Beal, 2003). It can be shown that the difference between the ELBO and KLD is

the log normaliser, which is what the ELBO bounds (Beal, 2003). Hence, minimizing KLD is the same as maximizing the ELBO on the log marginal likelihood.

2.4.4 Hidden Markov Models (HMMs)

For modelling a large number of temporal sequences, HMM, which combines time series with probabilistic characteristics, has been identified as an excellent tool (Thanthrige, 2016). For this reason, it has been widely used for pattern matching in speech recognition (Rabiner, 1989), image identification (Bunke and Caelli, 2001), and for identification of network attacks (Cuppens, 2001). Warrender *et al.* (1999) introduced HMM into anomaly detection for the first time. If an attack is considered to be a pattern of an observed sequence, HMM will be appropriate for mapping those patterns to one of many attack states (Khanna, 2008). Several researchers, such as Khanna (2008), Sendi *et al.* (2012), Khosronejad *et al.* (2013), Saini *et al.* (2014), Sharma and Manoria (2015), have attempted using HMM in one form or the other to either detect or predict network attack.

HMM, the simplest form of Dynamic Bayesian Network (DBN) (Murphy, 2001; Ankali and Ashoka, 2011), is a doubly stochastic process: an unobservable (hidden) process S, which can only be observed through another (visible or observable) stochastic process O. Each state in Q (the set S of hidden states) has state-transition probabilities (which are not visible) and a probability distribution over the possible values of O. The basic assumption in HMM is that the current hidden state of the system is affected only by its previous state.

2.4.4.1 Components of HMM

HMM for network intrusion prediction, just like the basic HMM, is usually defined as a 5-tuple (S, O, A, B, π), (Ibe, 2013) where:

$S = \{s_1, s_2, \ldots, s_N\}$ is a finite set of N states

$O = \{o_1, o_2, \ldots, o_M\}$ is a finite set of M possible symbols

$A = \{A_{ij}\}$ is the set of state-transition probabilities where a_{ij} is the probability that the system goes from state s_i to s_j, such that $\sum_{j=1}^{n} a_{ij\,=1}$ for all i

$B = \{b_i(o_k)\}$ are the observation probabilities or likelihoods, where $b_i(o_k)$ is the probability that the symbol o_k is emitted when the system is in state s_i.

$\pi = \{\pi_i\}$ are the initial probabilities; i.e. π_i is the probability that the system starts in state s_i. Some states j may have $\pi_j = 0$, meaning that such states cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$.

Since the states and output sequence are understood, it is customary to compactly denote the parameters of an HMM by:

$$\lambda = (A, B, \pi) \tag{9}$$

HMM can be formally represented as in Figure 4 below where S_i are the hidden states that we would like to estimate and the O_i are the observation random variables from which the S_i are to be estimated. The letters C and E indicate the *Commencement* and *End* of the sequence of states.

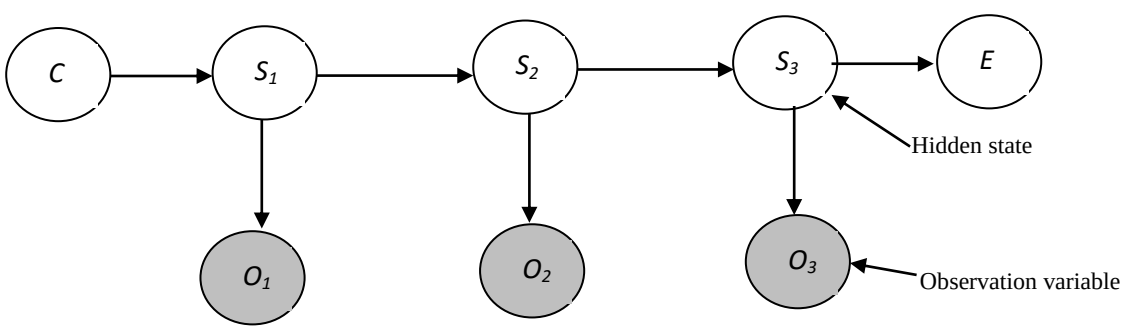


Figure 4: General Structure of an HMM

2.4.4.2 HMM-Based Prediction Model

In HMM-based prediction modelling there are three basic problems to be solved namely: Likelihood (or Classification or Evaluation), Decoding (or Inference) and Learning (or Estimation) problem

i) The Likelihood problem: this deals with how to efficiently compute the probability that a particular model $\lambda = (A, B, \pi)$ generated a given observation sequence $V = v_1, v_2, \ldots, v_T$ of length T where $v_i \varepsilon O$, i.e. What is $P[O| \lambda]$

ii) The Decoding problem: this is concerned with determining the most likely sequence of hidden states that could have generated a given observation sequence given a model $\lambda = (A, B, \pi)$, i.e. finding $Q^* = \arg\max_Q P[Q, O| \lambda]$

iii) The Learning problem: this deals with finding the model λ that best explains a given observation sequences. i.e. to estimate the most likely HMM parameters for a given observation sequence. This translates into finding the values of $P[O| \lambda]$, or $\lambda^* = \arg\max_\lambda P[Q, O| \lambda]$

Depending on the particular problem to be solved out of the three basic HMM problems, there are different solution methods (Ibe, 2013). The first problem (Likelihood problem) is usually solved by the forward algorithm and backward algorithm. In this study, the forward algorithm is discussed. The second HMM problem (decoding problem) is normally solved by the application of a specific type of the forward algorithm named Viterbi algorithm. The solution to this problem attempts to uncover the hidden part of the stochastic model. Lastly, the third problem (learning problem) is

usually solved by an iterative method called the Baum-Welch algorithm (sometimes called forward-backward algorithm), which is a special case of the Expectation Maximization (EM) method. This problem deals with how to adjust the HMM parameters so that the given set of observations, usually called the training set, is represented by the model in the best way for the intended application. It is an optimization problem that seeks to find the parameters of the HMM that maximise the probability of a given observation sequence.

2.4.4.2.1 Likelihood Computation - Forward Algorithm

Given an HMM, $\lambda = (A, B, \pi)$ and observation sequence$O$, determine the likelihood $P(O|\lambda)$. To achieve this, a forward probability variable $\alpha_t(i)$, which is the probability of being in state s_i at time t after observing the sequence $[o_1, o_2, \ldots, o_t]$ is defined as:

$$\alpha_t(i) = P[o_1, o_2, \ldots, o_t, q_t = s_i|\lambda] \qquad t = 1, \ldots, T;\ i = 1, \ldots, N \tag{10}$$

$$P(O|Q) = \prod_{i=1}^{T} P(o_i|q_i) \tag{11}$$

The forward algorithm computes the observation probability by summing over the probabilities of all possible hidden states paths that could generate the observation sequence. This is efficiently done by implicitly folding each of these paths into a single forward trellis.

$$\alpha_t(i) = P[o_1, o_2, \ldots, o_t, q_t = s_i|\lambda] \tag{12}$$

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(O_t) \tag{13}$$

2.4.4.2.2 Decoding - The Viterbi Algorithm

Given an observation sequence O and an HMM, λ = (A, B, π), seek the best (or optimal) hidden state sequence Q.

As earlier mentioned, the most popular decoding algorithms for HMM is the Viterbi algorithm. Though Viterbi algorithm (Ibe, 2013), originally designed for decoding convolutional codes, is applied in many other areas, in HMM, it is used to find the most likely state sequence Q* = $\{q_1^*$, q_2^*,…, $q_T^*\}$ for a given observation sequence $O = o_1, o_2, \ldots, o_T$. Each cell of the trellis, $v_t(j)$, represents the probability that the HMM is in state j after seeing the first t observations and passing through the most probable state sequence $q_0, q_1, q_{2,} \ldots \ldots . q_{t-1}$, given the automaton λ. The value of each cell $v_t(j)$is computed by recursively taking the most probable path that could lead us to this cell. Formally, each expresses the probability

$$v_t(j) = \max_{q_0, q_1, \ldots . q_{t-1},} P(q_0, q_1, \ldots . q_{t-1}, o_1, o_2, \ldots o_t, q_t = j|\lambda) \quad (14)$$

$$v_t(j) = \max_{i=1} v_{t-1}(i) a_{ij} b_j(o_t) \quad (15)$$

2.4.4.2.3 HMM Training: The Baum-Welch Algorithm

Given an observation sequence O and the set of states in the HMM, learn the HMM parameter $\lambda = (A, B, \pi)$ i.e., how do we adjust the model parameter to maximize $P(O|\lambda)$?

The Baum-Welch algorithm, which is a kind of forward-backward algorithm, is used to train both the transition probability A and the emission probabilities B of the HMM. The algorithm sets out setting the parameters A, B and π to initial values, which could be chosen from prior knowledge or some kind of uniform distribution. By using the current model, all probable paths for each training set are considered to arrive at new estimates $\hat{A}, \hat{B}, and\, \hat{\pi}$. This procedure is repeated till

insignificant changes in the current model parameters is achieved. The forward algorithm in Baum-Welch is the same as defined in equation (10) of section 2.4.4.2.1. The backward probability β is the probability of seeing the observations from time, $t + 1$ to the end, given that we are in state s_i at time t (and given the model λ):

$$\beta_t(i) = P[o_{t+1}, o_{t+2}, \dots, o_T | q_t = s_i | \lambda] \qquad t = 1, \dots, T;\ i = 1, \dots, N \qquad (16)$$

The forward and backward algorithm are used to compute the transition probability and observation or emission probability from the observation sequence. In other to re-estimate (iterative update and improvement) the HMM parameters, a new term ξ_t is defined.

$$\xi_t\,(i,j) = P\left[q_t = s_i, q_{t+1} = s_j | O, \lambda\right] \qquad (17)$$

To compute ξ_t, the term $\widetilde{\xi_t}$ is computed similar to ξ_t, but differs due the probability of the observation.

$$\widetilde{\xi_t}(i,j) = P\left[q_t = s_i, q_{t+1} = s_j, O | \lambda\right] \qquad (18)$$

$$\widetilde{\xi_t}(i,j) = \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \qquad (19)$$

Following the laws of probability and dividing by $P[O|\lambda]$ yields,

$$\xi_t\,(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P[O|\lambda]} \qquad (20)$$

$$\xi_t\,(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \qquad (21)$$

Thus, the forward probability of the whole network traffic (or alternatively, the backward probability of the whole network traffic), is computed with the aid of equation 22.

$$P(O|\lambda) = \alpha_T(N) = \beta_T(1) = \sum_{j=1}^{N} \alpha_t(j)\beta_t(j) \quad (22)$$

Therefore, equation 21 is expressed as

$$\xi_t\,(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\alpha_T(N)} \quad (23)$$

The expected number of transitions from state i to j is then the sum over all t of ξ.

$$\widehat{a_{ij}} = \frac{\sum_{t=1}^{T-1} \xi_t\,(i,j)}{\sum_{t=1}^{T-1}\sum_{j=1}^{N} \xi_t\,(i,j)} \quad (24)$$

Re-computing the observation probability, there is need to determine the probability of being in state j at time t, which is denoted by $\gamma_t(j)$:

$$\gamma_t(j) = P\left[q_t = s_j|O,\lambda\right] \quad (25)$$

Computing equation 25 to include the observation sequence in the probability yields,

$$\gamma_t(j) = \frac{P\left[q_t = s_j|O,\lambda\right]}{P[O|\lambda]} \quad (26)$$

Substituting equation (23) into equation (25) gives,

$$\hat{b}_j = \frac{\sum_{t=1,O_t=K}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \quad (27)$$

Equation 23 and 27 depicts the re-estimation of transition probability and emission probability.

2.4.5 Variational Bayesian Inference (VBI)

The maximum likelihood (ML) approach to learning models from data, a typical example of which is Baum-Welch algorithm, does not take into consideration model complexity. It is therefore susceptible to over-fitting the data. More complex models can usually give ever-increasing

likelihoods to the data. For an HMM, the complexity can be ascribed to several aspects such as the number of hidden states, n, in the model, the degree of connectivity in the state transition matrix, A, and the distribution of probabilities to the symbols by each state, as specified in the emission matrix, B. More generally the complexity is related to the richness of possible datasets that the model can produce. There are $n(n - 1)$ parameters in the transition matrix, and $n(m - 1)$ in the emission matrix, and so if there are many different observed symbols or if it anticipated that more than a few hidden states would be required then, aside from inference becoming very costly, the number of parameters to be fitted may begin to overwhelm the amount of data available (Beal, 2003). Traditionally, in order to avoid over-fitting, researchers have limited the complexity of their models in line with the amount of data they have available, and have also used sophisticated modifications to the basic HMM to reduce the number of free parameters (Beal, 2003). Such modifications include parameter-tying, enforcing sparsity constraints (for example limiting the number of candidates a state can transition to or symbols it can emit), or constraining the form of the hidden state transitions (for example employing a strict left-to-right ordering of the hidden states).

In machine learning, Variational Bayesian Inference, which can also be referred to as variational Bayes, is mostly used to infer the conditional distribution (also known as the posterior distribution) over the latent variables given the observations (and parameters). VBI for HMMs seeks to minimise the divergence between the true posterior and an approximation in which the parameters and hidden variables are assumed independent, which assumption allows for a very efficient iterative solution (MacKay, 1997; Beal, 2003 and Fox and Roberts, 2011). Since the aim of VBI is to approximate a conditional density of latent variables given observed variables, the paramount idea is to solve the problem with optimisation. This is achieved by picking a family of distributions over the latent variable with its own variational parameters ($q(Y_{1:m}|V)$) and then find the setting of

the parameters that makes q close to the posterior of interest (Gruhl and Sick, 2016). q is used with the fitted parameters as a proxy for the posterior. The closeness of the two distributions is measured with KLD as in section 2.4.3 above (Beal, 2003).

2.4.5.1 Choice of Distributions

For each state i, an independent Dirichlet prior distribution for the transition probabilities is assigned such that:

$$p(A) = \prod_{i=1}^{N} Dir(\pi_i|\alpha_i^{(0)}) \tag{28}$$

$$\alpha_i^{(0)} = \left\{\alpha_{i,1}^{(0)}, \dots, \alpha_{i,N}^{(0)})\right\} \tag{29}$$

The variational posterior distributions for the model parameters result in the following form:

$$q(A) = \prod_{i=1}^{N} Dir(\pi_i|\alpha_i) \tag{30}$$

$$\alpha_i = \left\{\alpha_{i,1}, \dots, \alpha_{i,N}\right\} \tag{31}$$

The means are assigned independent univariate Gaussian conjugate prior distributions, conditional on the precisions.

As stated earlier, Dirichlet distribution is chosen because it is the conjugate of the complete-data likelihood terms of the HMM.

CHAPTER THREE

RESEARCH METHODOLOGY

This chapter presents the research approaches employed in this study in order to achieve the specific objectives stated in Chapter One.

3.1 The Model Design

The methodology of the DDoS attack prediction system development was carried out based on the overall system design as illustrated in Figure 5. To the best knowledge of the author, no known work had combined all the methods used in this study for DDoS attack prediction. The proposed model is based on HMM algorithms. Once the network traffic is captured, a formatting process is carried out after which the desired network features are extracted from the data. The traffic is analysed at regular time intervals and the features listed in Section 1.3.4 were determined using the concept of normalised entropy. After which *K*-means clustering was deployed in partitioning the traffic into the distinguished DDoS phases. The HMM was formulated as a state-space representation, where these features are the observable states while the phases of the DDoS attack form the hidden states of the model. The initial probability, transition and emission (or output) matrices were derived from the data.

The entropy-based features to be used as the observable states of the HMM are many. Therefore, in order to reduce the computational time and achieve model parsimony, KLD is adapted to select the minimum number of the features that could represent the whole to achieve improved performance without loss of information. Due to the shortcomings of HMMs especially the traditional learning algorithm of the HMM (Baum-Welch algorithm), VBI is deployed in training the formulated model.

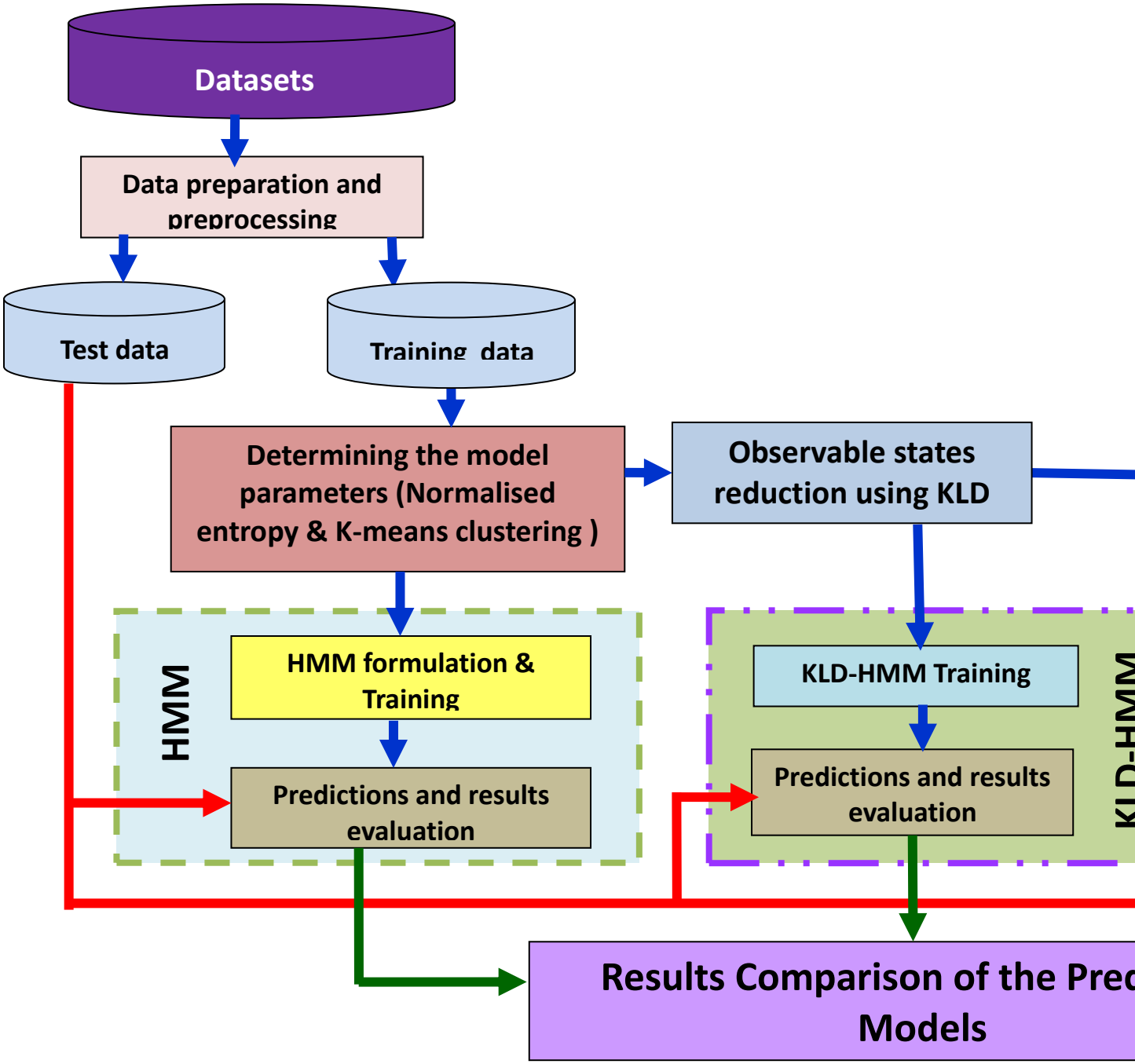


Figure 5: The Flowchart of the Model Architecture

The experimental procedure consists of five major steps. In the first step, the network states are defined by means of clustering the network traffic based on the evaluated entropy-based values of the network traffic features. In the second step, the initial probability distribution, the state transition

probability and the emission transition probability of the HMM are built based on the definitions got from the first step. In the third step, the HMM is trained using the DARPA 2000 intrusion dataset after which two sets of test data (DARPA 1999 (no attack) dataset and CAIDA 2007 simulated DDoS attack real-time dataset) are used to test the model and make predictions. In the fourth step, KLD was used to reduce the observables state space of the HMM using algorithm 5. The model formulated at this stage was also trained and tested as in step three. In the fifth step, the KLD-HMM model of step four was optimized using VBI. The optimized model was also tested and used for prediction. Finally, the results and computational efficiency of the three models were compared with an existing work.

3.2 HMM Construction

This section discusses the methodology for formulating the entropy-based state-space HMM for DDoS attack prediction.

3.2.1 Experimental Datasets Description

There are three different datasets that are used in this study. The various datasets are described in section 3.2.1.1 through section 3.2.1.3. Each dataset was pre-processed and transformed into the format that can be accommodate by the implemented platform.

3.2.1.1 Training Data - DARPA 2000 Intrusion Scenario Specific Datasets

To implement and train the model, the **DARPA 2000 Intrusion Scenario Specific Datasets** (MIT Lincoln Lab. 2000), which is available at http://www.ll.mit.edu/IST/ideval/data/2000/2000_data_index.html is used. Specifically, the LLDOS 2.0 - Scenario Two which presents an attack scenario datasets created for DARPA that includes a

DDoS attack. The decision to use this datasets was born out of the desire to achieve objectivity in results comparisons. This is due to the fact that the datasets is a benchmark data that have been used in several similar earlier works such as the works of Lee *et al.* (2008), Sendi *et al.* (2012), Shin *et al.* (2013), amongst several others.

The data were downloaded (see appendix A for a subset of the data in the extracted and Excel formats). About two hundred thousand network traffic records were contained in the data. After the downloading of the data, it was converted into the format that can be accommodated by the programming platform.

The downloaded dataset is the first attack scenario data set to be created for DARPA. It includes a distributed denial of service (DDoS) attack run by a novice attacker. It was reported that future versions of this and other example scenarios will contain more stealthy attack versions (MIT Lab. 2000).

This attack scenario is carried out over multiple networks and audit sessions. The sessions have been grouped into five attack phases, over the course of which the attacker probes the network, breaks in to a host by exploiting the Solaris sadmind vulnerability, installs Trojan mstream DDoS software, and launches a DDoS attack at an offsite server from the compromised host.

From the data, the parameters listed in section 1.3.4 will be estimated. To compute the entropies, first the probability of each quantity in the training data will be computed and plugged into equation (1).

3.2.1.2 Test Data I - 1999 DARPA Intrusion Detection Evaluation Dataset

The 1999 DARPA Intrusion Detection Evaluation Dataset, which is available at https://www.ll.mit.edu/ideval/data/, was created as an improvement over the 1998 DARPA Intrusion Detection (ID) Evaluation Dataset. The evaluation was of two parts: a real-time evaluation and an off-line evaluation.

In this dataset, intrusion detection systems testing were carried out in the off-line evaluation using audit logs and network traffic collected on a simulation network. This data was processed in batch mode by the systems, which attempted to identify attack sessions in the midst of normal activities.

For the real-time evaluation, IDSs were delivered to Air Force Research Laboratory (AFRL). These systems were inserted into the AFRL network test-bed and attempts were made to identify attack sessions in the midst of normal activities, in real-time. IDSs were tested as part of the real-time evaluation or the offline evaluation or both.

For the 1999 DARPA intrusion detection off-line evaluation, three weeks of training data were provided. The first and third weeks, which contained no attacks was provided to facilitate the training of anomaly detection systems.

For each day, several files such as outside sniffing data, and inside sniffing data (both in tcpdump format), BSM audit data, and so on are provide. In this study, the first week Tuesday no attack inside sniffing data, with over a million network traffic records, was used to test the efficiency, precision and false positive rate of the formulated model.

3.2.1.3 Test Data II - Center for Applied Internet Data Analysis (CAIDA) 2007 DDoS Attack Dataset

The data were downloaded from CAIDA website at https://data.caida.org/datasets/security/ddos-20070804/ on approval (See appendix B for the approval email) of a personal request to use the data for this study. In this dataset the network traffic from the victim and the network traffic to the victim were captured in separate files. But for the purpose of this study the two sets of network traffic, from the victim and to the victim were merged together and sorted by time like the original files. The merged file contained about one hundred and seventy thousand network traffic records.

3.2.2 Defining the Network States

This step extracts the desirable features of the temporal network data using the concept of entropy as discussed in Section 2.4.1 and illustrated in algorithm 1.

3.2.2.1 Algorithm 1: ***Calculating normalised entropy***

Input: Network Traffic

Output: Normalised entropy for each network feature

Loop: each interval of time until the traffic comes

Extract features from packet header

loop: for each packet in the time interval

Calculate frequency of all distinct *x* // where x = source IP, destination IP, source //port number, destination port number, packet type //(UDP, ICMP, TCP-SYN), packet size

End

Loop: for each distinct *x*

Calculate probability for each distinct *x*

$P_i = m_i/N$ // m_i = frequency of the ith x; N = total number of packets in //that time interval

Calculate entropy for each distinct x

$h_i = -\sum_{i=1}^{n} P_i \log P_i$

End

Normalise the entropy, in the time interval by

$H = -\sum_{i=1}^{n} h_i/\log(F)$ // F is total number of distinct x

END

3.2.3 Determining Model Parameters

In other to address the first problem of determining the hidden states and observable states of the HMM, the features listed in section 1.3.4 were dynamically estimated from the DARPA 2000 datasets using Algorithm 1, first at regular intervals of one second and then five seconds. It was discovered that the two intervals produced the same results. So, five seconds sampling interval was used. As earlier mentioned, to compute the entropies, the probabilities of each unique quantity in the training data is computed and plugged into equation (1).

After this, *K*-means clustering algorithm as presented in Algorithm 2 is applied to classify the network behaviour into states. The state of each observation is identified by the cluster it belongs to.

3.2.3.1 Algorithm 2: ***Clustering Algorithm***

Let n be the number of clusters wanted

Let S be the set of feature vectors ($|S|$ is the size of the set)

Let A be the set of associated clusters for each feature vector

Let sim(x,y) be the similarity function

Let $c[n]$ be the vectors for the clusters

Initialise:

```
    Let S' = S      //choose n random vectors to start the clusters
    For i=1 to n
        j = rand(|S'|)
        c[n] = S'[j]
        S' = S' - {c[n]}    //remove that vector from S' so it cannot be chosen again
    End for

    //assign initial clusters
    For i=1 to |S|
        A[i] = argmax(j = 1 to n) { sim(S[i], c[j]) }
    End for
```

Run:

```
    Let change = true
    While change
        change = false   //assume there is no change reassign feature vectors to clusters
    For i = 1 to |S|
        a = argmax(j = 1 to n) {sim(S[i], c[j])}
       If a!= A[i]
          A[i] = a
          change = true   //a vector changed affiliations - hence a need to re-compute
                          //the cluster vectors and re-run
        End if
```

```
End for
//recalculate cluster locations if a change occurred
If change
  For i = 1 to n
      mean, count = 0
      For j = 1 to |S|
          If A[j] == i
              mean = mean + S[j]
              count = count + 1
          End if
      End for
      c[i] = mean/count
  End for
End if
```

3.2.4 HMM Formulation, Training and Testing

Here, the HMM parameters, $\lambda = (A, B, \pi)$, is formulated based on the values got from the procedures in Section 3.2.3 above.

HMM is usually defined as stated in section 2.4.4.1

In this study, the five phases of DDoS resulting from Section 3.2.3 and an additional state N that represents the normal state when no malicious activity is going on in the system, form the set of hidden states from which the parameters A and π are derived. The first six consecutive entropy-

based features as listed in Section 1.3.4 forms the set of observables from which the emission matrix B is derived.

Next, the model so formulated was trained using the Baum-Welch algorithm (See Algorithm 3) until convergence. Then the two sets of test data as aforementioned were used to test the model and make predictions. The prediction module is implemented using Algorithm 4.

3.2.4.1 Algorithm 3: ***<u>Baum-Welch algorithm for training the HMM</u>***

Input: A set of observed sequences, O_1, O_2,…; Initial model parameter formulated ($\lambda = (A, B, \pi)$)

Initialize: $\lambda = (A, B, \pi)$

For i =1 to N **do***

$\pi_i =$ eexp($\gamma[\theta][i]$

End for

Run:

Do while ($\lambda \neq \lambda'$) **and**

(maximum iteration not reached) **and**

(likelihood tolerance not reached)

Subtract maximum iterations

If $\lambda' = \varphi$,

$\lambda \leftarrow \lambda'$

End if

Run Forward and Backward procedures

Calculate estimator variables ξ, γ, Component(γ), μ, Σ, c

Estimate π

Estimate A

For all states

For all components

Create Mixture distribution with estimated parameters and assign it to model component

End for

End for

Create estimated model and calculate likelihood

3.2.4.2 Algorithm 4: ***Viterbi Algorithm for prediction***

Set parameters π, A, B, O, T, N

//***Initialization***: //

For i = 1 to N **do**

$\delta_1(i) \leftarrow \pi_i * b_i(o_1)$

$\psi_1(i) = 0$

End for

//***Recursion***://

Repeat for j = 1 to N

Repeat for t = 2 to T

For all i,

Compute $\{a_j\} = \delta_{t-1}(i)\, a_{ij}$

End for

$\delta_t(j) \leftarrow \max\{a_j\} b_j(o_t)$

$\psi_t(j) \leftarrow \arg(\delta_t(j))$

End repeat

End repeat

//***Termination***://

$P^* \leftarrow \max_{1 \le i \le N}\{\delta_T(i)\}$

$q_T^* \leftarrow \arg P^*$

//***State Sequence*** (***Path) backtracking***//

For $t = T - 1$ to 1

$q_t^* = \psi_{t+1}(q_{t+1}^*)$

End for

$Q^* = \{q_t^*\}$

Return Q^*

3.3 Reduction of the Observable States Space of the Model

The section presents the method for reducing the parameter space of the HMM in order to achieve a computationally efficient model for predicting DDoS attacks.

To solve the second problem of long training time and also achieve model parsimony, the observable states of the model formulated in Section 3.2 was reduced using the KLD algorithm presented in Algorithm 5. The resultant model was also trained and tested using the same datasets.

3.3.1 Algorithm 5: ***Kullback-Liebler Divergence for reducing observable state space***

Step 1: Load all Datasets

Step 2: Load the results of Algorithm 1 in section 3.2.2.1 above

Step 3: Do while data

Compute: $D_{KL}(P||Q) = \sum_i^{\mathrm{n}} P(i) \ln \frac{P(i)}{Q(\mathrm{i})}$

Where:

For Case (a): P = Source IP address and Q = Source Port

For Case (b): P = Destination IP address and Q = Destination port

For Case (c): P = Occurrence rate of packets per unit time and Q = Packet type

If $D_{KL}(P||Q) \rightarrow 0$, (i.e. if it tends towards zero)

then either P or Q can be used in each case

Else use both P and Q

End if

3.4 VBI-HMM Formulation

This section presents the methodology for implementing a global convergent algorithm for training the formulated HMM in order to reduce its complexity.

To solve the third problem of convergence to local maxima of the traditional HHM training algorithm and in pursuance of good performance in overall computational time, the model derived in Section 3.3 was retrained using VBI algorithms presented in Algorithm 6. The resultant model was also tested using the same datasets.

3.4.1 Algorithm 6: ***Variation Bayesian Inference algorithm***

Let [net] = VBIHMM(*ds*, alphabet, *K*, *Y*, *t*, *net*)

Let K = number of states

Let Y = maximum number of iterations

Let t = termination tolerance

Let pm = previously learnt model

Let ds = cell array of strings

Let alphabet = strings of symbols

Let net.Pa = state transition dirichlet counts

Let net.Φb = observation emission Dirichlet counts

Let net.π = initial state prior Dirichlet counts

net.F = F learning curve

If nargin < 6

  Initfromprior ←1

Else

  Intfromprior ←0

Endif

If nargin < 5

  t ←0.0001

Endif

If nargin <4

   Y←100

Endif

L ← size(alphabet, 3)

N ← size(datastrings, 1)

```
For n = 1, 2, …, N
   Q(n) ← size(datastring{n}, 3)
   tmp ← zeros(1, Q(n))
        For j = 1, 2,…, L
       tmp(1, find (datastring{n} = alphabet(j))) = j
   Endfor

   data{n} ← tmp
Endfor
data{n} ← tmp
Endfor

αa ← 1
αb ← 1
απ ← 1


Let ua = ones(1, K) * (αa/K)
Let ub = ones(1, L) * (αb/L)
uπ = ones(, K) * (απ/K)
 pa = [ ], pb = [ ]
For k = 1, 2, …, K
      pa(k,:) = dirrnd(ua,1) * tot_len
      pb(k,:) = dirrnd(ub,1) * tot_len
end
pπ = dirrnd(uπ,1) * N
Fold = - lnf; ntol = tol * N
For  i  = 1, 2, …, Y
```

If (initfromprior = 0 and *i* = 1)

Pa ← hmm.*Pa*

Pb ← hmm.*Pb*

Pπ ← hmm.*Pπ*

Else

Pa ← pa + repmat(ua, [K 1])

Pa ← pb + repmat(ub, [K 1])

Pπ ← pπ + uπ

Endif

3.5 Evaluation Metrics

The results obtained from each of the above models were compared using standard metrics for intrusion prediction obtainable from the confusion matrix such as false positive (FP) rate, false negative (FN) rate, true positive (TP) rate, true negative (TN) rate, precision rate, prediction accuracy (ACC). Where:

1) **Prediction Accuracy** (ACC) is the fraction of the total number of predictions that were correct i.e. $ACC = \frac{\sum_{record=1}^{n} no.\ of\ \ correct\ classification}{\sum_{record=1}^{n} no.\ of\ \ total\ classification} = \frac{(TN + TP)}{(TN + TP + FN + FP)}$ (32)

2) **Precision (PR)** is the fraction of data instances predicted as positive that are actually positive.

$$PR = \frac{TP}{TP + FP} \quad (33)$$

3) **Confusion matrix** is a matrix that represents result of classification. It represents true and false classification results (Kumar, 2014). The possibilities to classify events are as enumerated below and depicted in Table 3.

- True positive (TP): Intrusions that are successfully predicted by the IPrS.
- False positive (FP): Normal or non-intrusive behaviour that is wrongly predicted as intrusive by the IPrS.
- True Negative (TN): Normal or non-intrusive behaviour that is successfully labelled as normal by the IPrS.
- False Negative (FN): Intrusions that are missed by the IPrS, and classified as normal.

Table 3: Confusion Matrix

Actual	Predicted	Predicted
	Intrusion	Normal
Intrusion	TP	FN
Normal	FP	TN

4) **True Positive Rate (TPR) (also referred to as Detection Rate or Sensitivity or Recall)** is the fraction of positive cases that were correctly predicted i.e.

$$TPR\ or\ Sensitivity = \frac{\sum_{record=1}^{n} no\ of\ \ predicted\ attacks}{\sum_{record=1}^{n} no\ of\ total\ attacks} = \frac{TP}{TP + FN} \tag{34}$$

5) **False Positive Rate** (FPR) or false alarm rate (FAR) is defined as the percentage ratio of incorrectly classified normal examples (false alarms) to the total number of normal examples

$$FPR = \frac{\sum_{record=1}^{n} no\ of\ \ false\ alarms}{\sum_{record=1}^{n} no\ of\ \ total\ normal\ records} = \frac{FP}{FP+TN} = 1 - specificity \tag{35}$$

6) **True Negative Rate (TNR) or Specificity** is the fraction of negative cases that were classified correctly i.e.

$$(TNR)\ or\ Specificity = \frac{(TN)}{(TN+FP)}\ ; \tag{36}$$

7) **False Negative Rate** (FNR) is the proportion of positive cases that were not predicted i.e.

$$(FNR) = \frac{\sum_{record=1}^{n} no\ of\ \ incorrectly\ classified\ \text{int}\ rusive\ records}{\sum_{record=1}^{n} no\ of\ \text{int}\ rusive\ records} = \frac{(\text{FN})}{(\text{FN}+\text{TP})} = 1 - Sensitivity \tag{37}$$

8) **F-measure (FM),** which is a measure of a test's accuracy, is the harmonic mean of the precision and recall. In most situations, there is a trade-off between precision and recall. If the classifier is optimised to increase one and disfavour the other, the harmonic mean quickly decreases. It is greatest however, when both precision and recall are equal. It is mathematically calculated as:

$$FM = \frac{2 \times (precision \times recall)}{(precision + recall)} \tag{38}$$

9) **Receiver Operating Characteristic (ROC) curves** illustrate the performance of each classifier. The ROC curve is a graphical metric that illustrates the performance of a classifier which in our case is an HMM that classifies packet sequence as "threat" or "normal traffic". It has been commonly used in the field of IDS and IFS in order to display trade-off between

detection/prediction rate and false alarm rate according to the change of internal thresholds (Wu and Banzh, 2010). It is very helpful in the choice of optimal value for a variable.

10) **Computational Time (CT)** is also another useful metric used in comparing the performance of the models. It is the total running time of the model, which comprises of the training time and the prediction time of the models.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Implementation Platform

The model architecture and the design methodology steps enumerated in Chapter Three were implemented in Matlab R2015c environment. As stated before, the model was tested using DARPA 1999 dataset and CAIDA 2007 simulated real time DDoS attack dataset.

Matlab is chosen as the implementation software because of its simulation capabilities and the ease with which major functions and concepts can be coded in that environment.

4.2 Results and Discussion of the Prediction Models

This section presents and discusses the experimental results of the models presented in Chapter Three.

In this section, the experimental results of the system is evaluated. As stated in chapter three, the training data and the two test data are respectively available at http://www.ll.mit.edu/IST/ideval/data/2000/2000_data_index.html, https://www.ll.mit.edu/ideval/data/, and https://data.caida.org/datasets/security/ddos-20070804/.

4.2.1 HMM Formulation

In order to determine the model parameters as stated in the first step of the design methodology in Section 3.1, some preliminary computations were carried out on the training dataset. To compute the normalised entropy values H of each network traffic features, first the probability of each distinct quantity in the training data was computed. Some of the sample results of this computation is as presented in Tables 4 through 6.

Table 4: Sample of Source IP and Destination IP Probabilities

S/N	Source IP	Probability	Destination IP	Probability
1.	128.11.47.66	4.06375E-05	128.11.47.66	4.06375E-05
2.	131.6.187.206	3.251E-05	131.6.187.206	4.06375E-05
3.	131.84.1.31	0.003714269	131.84.1.31	0.002665821
4.	132.24.126.14	0.001007811	132.24.126.14	0.000503905
5.	134.177.3.28	4.06375E-05	134.177.3.28	4.06375E-05
6.	134.177.3.35	0.004112517	134.177.3.35	0.003771162
7.	134.205.131.13	0.003332277	134.205.131.13	0.001942474
8.	134.205.165.120	0.003470444	134.205.165.120	0.002356976
9.	134.205.165.122	5.68925E-05	134.205.165.122	4.8765E-05
10.	135.13.216.191	0.015312218	135.13.216.191	0.009614838
11.	135.13.216.20	8.1275E-06	135.13.216.20	8.1275E-06
12.	135.8.60.182	0.027316542	135.8.60.182	0.020521948
13.	135.8.60.20	1.6255E-05	135.8.60.20	1.6255E-05
14.	136.149.63.132	0.002373231	136.149.63.132	0.001909964
15.	139.72.190.50	0.002040004	139.72.190.50	0.001706776
16.	140.140.138.108	0.001601118	140.140.138.108	0.001129723
17.	143.166.224.42	3.251E-05	143.166.224.42	4.06375E-05
18.	143.166.224.44	0.001527971	143.166.224.44	0.001422313
19.	143.166.224.45	4.06375E-05	143.166.224.45	4.06375E-05
20.	144.3.144.36	0.003007177	144.3.144.36	0.001666138

Table 5: Occurrence Rate of Packet Type and Packet Length

S/N	Packet Type	Occurrence Rate	Packet Length
1.	ARP	0.000707093	5220
2.	CDP	0.000601435	23680
3.	DNS	0.007160331	93230
4.	FINGER	0.000146295	1080
5.	FTP	0.002511399	25764
6.	FTP-DATA	0.001698648	276755
7.	HTTP	0.074220369	4257292
8.	ICMP	0.006152521	53416
9.	IMF	0.001511716	124764
10.	LLC	0.009622965	63936
11.	LOOP	0.003616739	26700
12.	NTP	0.000560798	6210
13.	POP	0.000406375	8157
14.	Portmap	3.251E-05	336
15.	SADMIND	1.6255E-05	2908
16.	SMTP	0.025577256	935022
17.	TCP	0.579751136	22070710
18.	TELNET	0.285706158	3481610

Table 6: Sample Probabilities of Source Port and Destination Port

S/N	Source Port	Probability	Destination Port	Probability
1	0	0.1333	0	0.1333
2	23	0.1583	23	0.3083
3	25	0.1833	25	0.1917
4	53	0.0083	53	0.0167
5	1489	0.0917	1489	0.0250
6	2072	0.2167	2072	0.1333
7	57205	0.0083	57224	0.0083
8	57224	0.0083	63330	0.0917
9	63330	0.0917	63331	0.0917

When determining the model parameters, six hidden states corresponding to the number of clusters were identified. Table 7 shows the values of each of the network features at the centroid of each cluster that correspond to the phases of the DDoS attack. These values represent the threshold values of each features at every phase of the DDoS attack. The states correspond to the phases of DDoS attack as listed in Section 1.3.4 (denoted by *I P*, *R*, *T* and *D* respectively) and an additional normal state (denoted by *N*) when there are no traces of malicious activity or any attempt to break into the system. So, $S_{i} = (s_1 = N; s_2 = I; s_3 = P; s_4 = R; s_5 = T; s_6 = D)$.

Table 7: Network Features Average for each phase of the DDoS Attack

Variable	**Cluster**					
	Normal	**Phase 1**	**Phase 2**	**Phase 3**	**Phase 4**	**Phase 5**
Entropy of Source IP	1.61	0.77	0.11	0.05	0.02	0.03
Entropy of Source Port	1.61	0.55	0.13	0.20	10.1	13.2
Entropy of Destination IP	1.57	5.01	0.09	0.10	11.1	12.6
Entropy of Destination Port	1.52	0.57	0.13	0.22	0.65	12.9
Entropy of Packet Type	1.23	0.51	0.06	0.09	0.10	0.01
Occurrence rate of TCP-SYN	0.02	0.01	0	0	0	0.01
Occurrence rate of UDP	0.01	0.02	0.96	0	0	0
Occurrence rate of ICMP	0.02	0.88	0	0	0	0
Number of Packets	36.87	41.35	1.21	2.10	1267	6225

The relationship among these states is as diagrammatically shown in Figure 6. For instance, at state *N*, the left to right edges show that the system can make a transition from state *N* to the next state *i* or remain at state *N*. The same holds for each of the states. The system can only transit a state at a time based on the HMM assumption. For the right to left edges, they show that the system can transit one state at a time to return to the normal state.

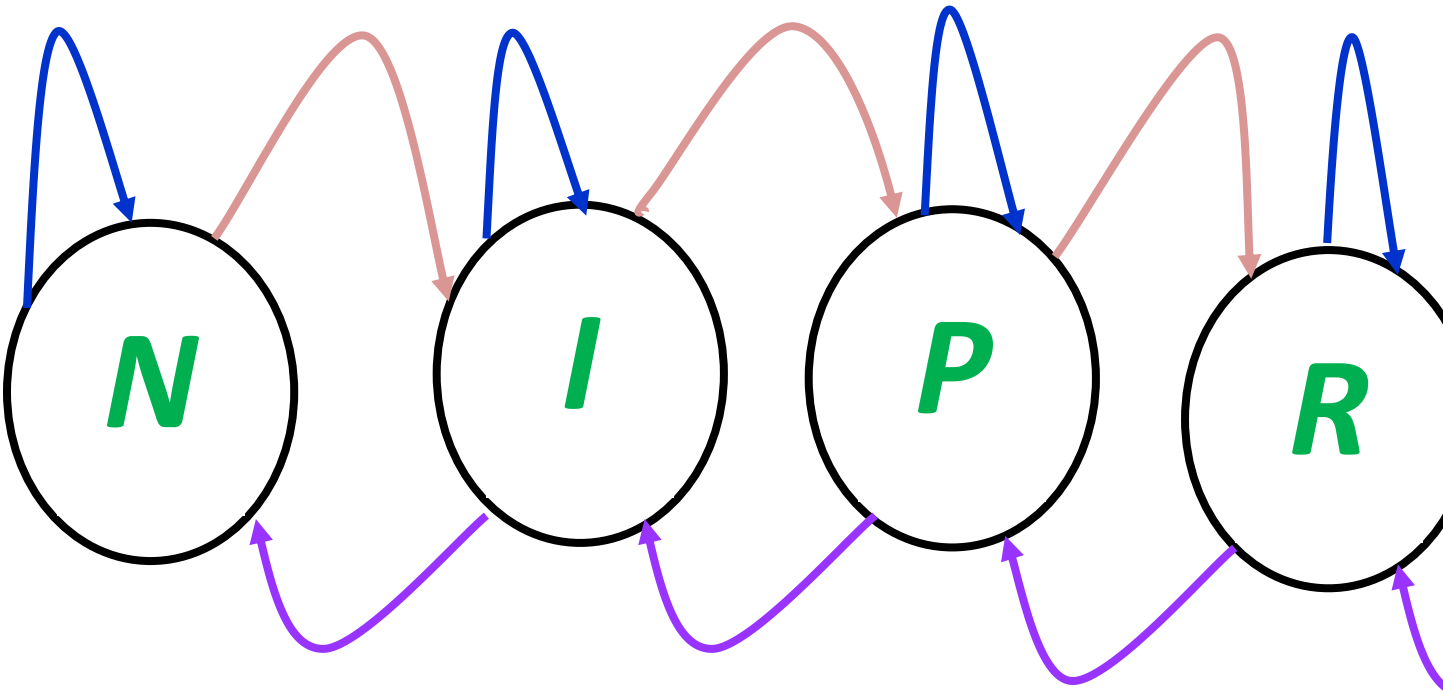


Figure 6: Hidden Markov Models states for prediction.

The finite set of M possible symbols ($O = \{o_1, o_2, o_3, \ldots, o_M\}$), in this study, are the first six of the features from the network traffic as listed in Section 1.3.4 within five seconds time interval. The observations are denoted by *IS, ID, PS, PD, PT, PO* respectively.

The State Transition Probability (A_{ij}), the Emission Transition Probability ($B_j(k)$) and the Initial State Probability (π_i) were obtained from the data and approximated to five decimal places. At system start-up, π = (0.97183, 0.02452, 0.00118, 0.00098, 0.00108, 0.00041), which implies that the system, has the probability of 0.97183 of being in state *N*; 0.02452 of being in state *I*; 0.00118 of being in *P*; 0.00098 of being in *R*; 0.00108 of being in *T*, and 0.00041 of being in state *D*. Next, the state transition probability (*A*), which is a 6 X 6 matrix and the emission probability matrix (*B*), also a 6 X 6 matrix was dynamically estimated from the temporal network as depicted below.

0.95880 0.00002 0.03819 0.00002 0.00216 0.00082

$$A = \begin{matrix} 0.03026 & 0.00001 & 0.96659 & 0.00311 & 0.00001 & 0.00001 \\ 0.00001 & 0.95527 & 0.04468 & 0.00001 & 0.00001 & 0.00001 \\ 0.00166 & 0.00404 & 0.00041 & 0.84611 & 0.14777 & 0.00002 \\ 0.00011 & 0.00011 & 0.00011 & 0.96471 & 0.00011 & 0.03484 \\ 0.00046 & 0.00001 & 0.00001 & 0.00151 & 0.00060 & 0.99741 \end{matrix}$$

$$B = \begin{matrix} 0.37212 & 0.01015 & 0.04177 & 0.06679 & 0.50579 & 0.00338 \\ 0.21831 & 0.00246 & 0.11835 & 0.61625 & 0.00905 & 0.03558 \\ 0.20108 & 0.00557 & 0.12168 & 0.62002 & 0.01369 & 0.03795 \\ 0.00960 & 0.00002 & 0.98823 & 0.00002 & 0.00002 & 0.00212 \\ 0.00011 & 0.00666 & 0.99289 & 0.00011 & 0.00011 & 0.00011 \\ 0.18821 & 0.60837 & 0.11356 & 0.00269 & 0.01843 & 0.06873 \end{matrix}$$

$$\pi = \begin{matrix} 0.97183 & 0.02452 & 0.00118 & 0.00098 & 0.00108 & 0.00041 \end{matrix}$$

The HMM, $\lambda = (A, B, \pi)$, was trained as earlier stated in Section 3.2, the model converged after 200 iterations in 714.90 seconds as shown in Table 8.

Two sets of test data, as earlier mentioned, were run through the model for prediction using algorithm 4. It was discovered that the model has FNR of 23%, FPR of 29%, TPR of 77%, TNR of 71% and the prediction accuracy is 74%.

Table 8: Relative Performance Analysis of various Models.

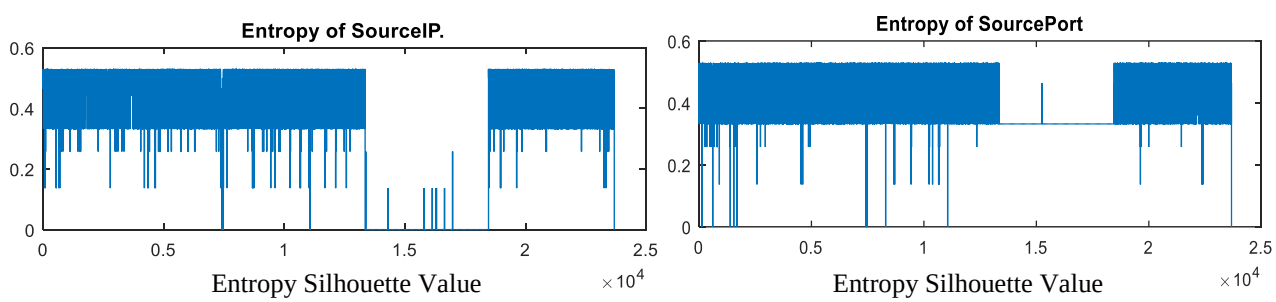
MODELS	Computational time in seconds	True Positive Rate (TPR)	False Negative Rate (FNR)	False Positive Rate (FPR)	True Negative Rate (TNR)	Prediction Accuracy (ACC)	F-Measure (FM)
APAN	30.73	0.83	0.17	0.22	0.78	0.81	0.819878
VBI-HMM	17.79	0.92	0.08	0.11	0.89	0.91	0.914973
KLD-HMM	59.53	0.84	0.16	0.21	0.79	0.82	0.82988
HMM	714.90	0.77	0.23	0.29	0.71	0.74	0.754702

4.2.2: HMM Parameter Space Reduction (KLD-HMM Formulation)

As stated in Section 3.3, to achieve the second objective of computational efficiency through HMM parameter space reduction, the KLD algorithm (see Algorithm 5) was applied on the formulated HMM. The dimensions of π, initial probability distribution, and the state transition probability (A) remained the same as in the original HMM. However, the dimension of the emission matrix was reduced to (6 X 3). The results obtained, as depicted in Figure 7, show the plots of the distribution

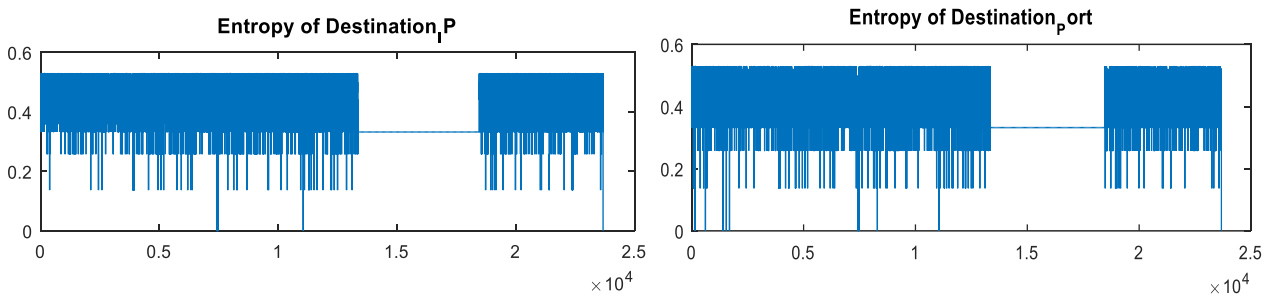
of the entropy values for the observable states being compared over time. It could be seen that the distribution of entropy of Source IP address (*IS*) and that of Source Port number (*PS*) are similar; the distribution graph of entropy of Destination IP address (*ID*) is similar to that of Destination Port number (*PD*) while that of Occurrence rate of Protocol (*PO*) is similar to that of entropy of Packet type (*PT*). This led to the choice of three of the observables namely: entropy of source IP (*SI*), entropy of destination IP (*DI*) and entropy of Packet type (*PT*) been used to represent the system. The resulting emission probability matrix ($B_{\text{KLD-HMM}}$) was a 6 X 3 matrix as depicted below:

$$B_{\text{KLD-HMM}} = \begin{matrix} 0.89179 & 0.10011 & 0.00810 \\ 0.58648 & 0.31794 & 0.09559 \\ 0.55745 & 0.33734 & 0.10521 \\ 0.00960 & 0.98828 & 0.00212 \\ 0.00011 & 0.99977 & 0.00011 \\ 0.50798 & 0.30650 & 0.18551 \end{matrix}$$

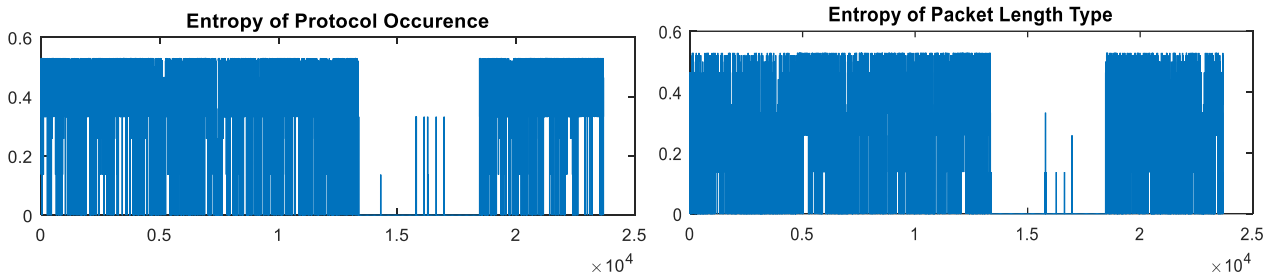


(a) Relative Entropy Distribution of Source IP and Source Port

Entropy Silhouette Value Entropy Silhouette Value



(b) Relative Entropy Distribution of Destination IP and Destination Port



(c) Relative Entropy Distribution of Protocol Occurrence Rate and Packet Type

The KLD-HMM model, like the original HMM, was likewise trained and used for prediction using the same sets of data. It was observed that the new model converges faster - after about 60 iterations in 59.53 seconds as against over 200 iterations in 714.90 seconds in the previous model. Also, as depicted in Table 8, there was considerable improvement in the values of FNR, FPR, TPR, TNR and prediction accuracy from 23%, 29% , 77%, 71% and 74% in the previous HMM to 16%, 21%, 84%, 79% and 82%, respectively.

Figure 8 through Figure 10 show the results of the comparison of the convergence rate of the loglikelihood, transition probabilities and the emission probabilities of the HMM and KLD-HMM constructed. Each graph shows that the convergence rate in the KLD-HMM model is better and faster than that of HMM for each variable considered.

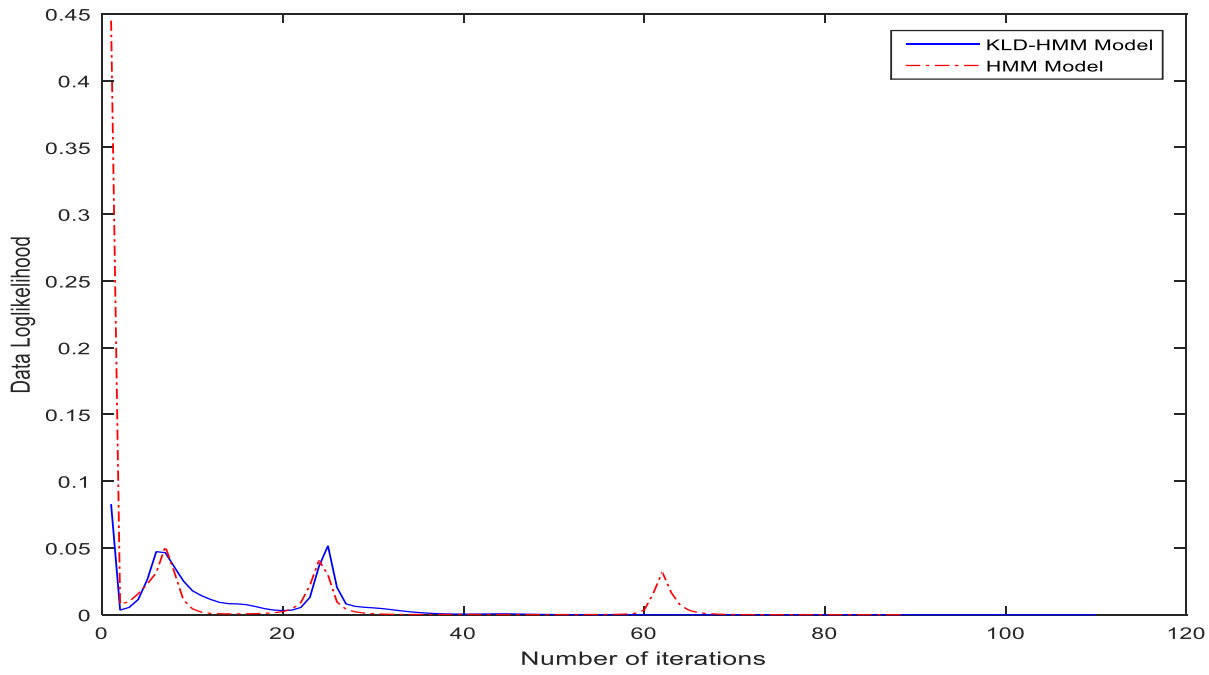


Figure 8: Convergence Rate of the Loglikelihood of the HMM and KLD-HMM

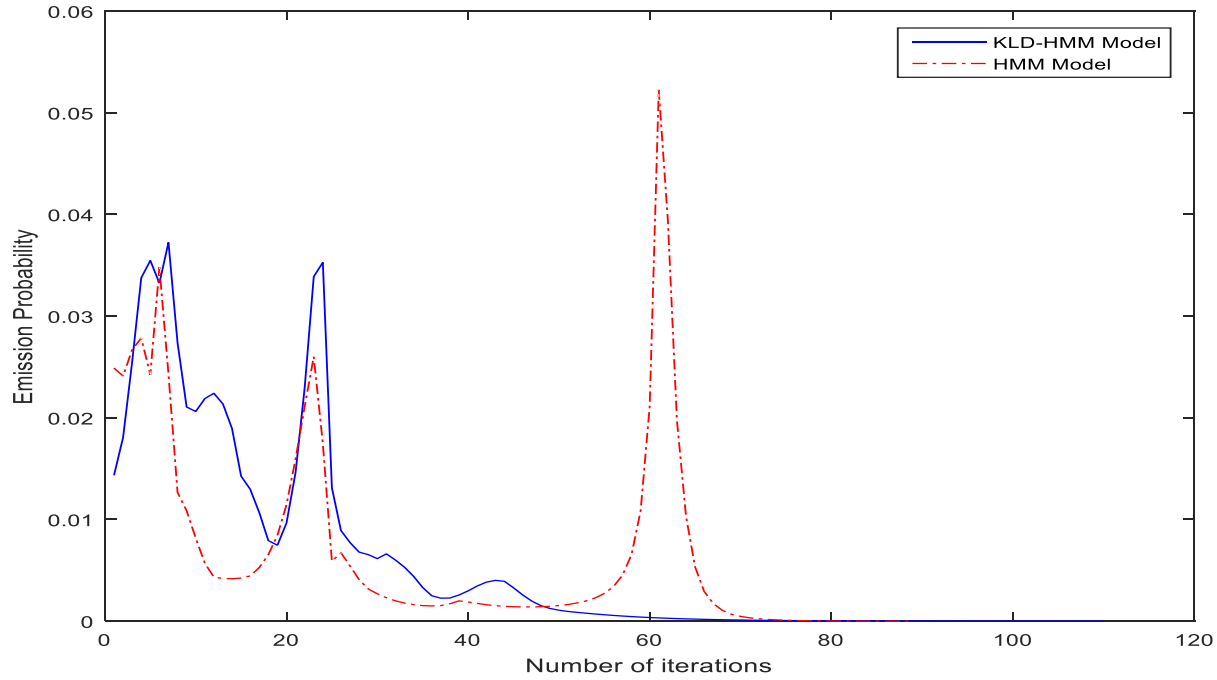


Figure 9: Convergence Rate of the Emission Probabilities of the HMM and KLD-HMM

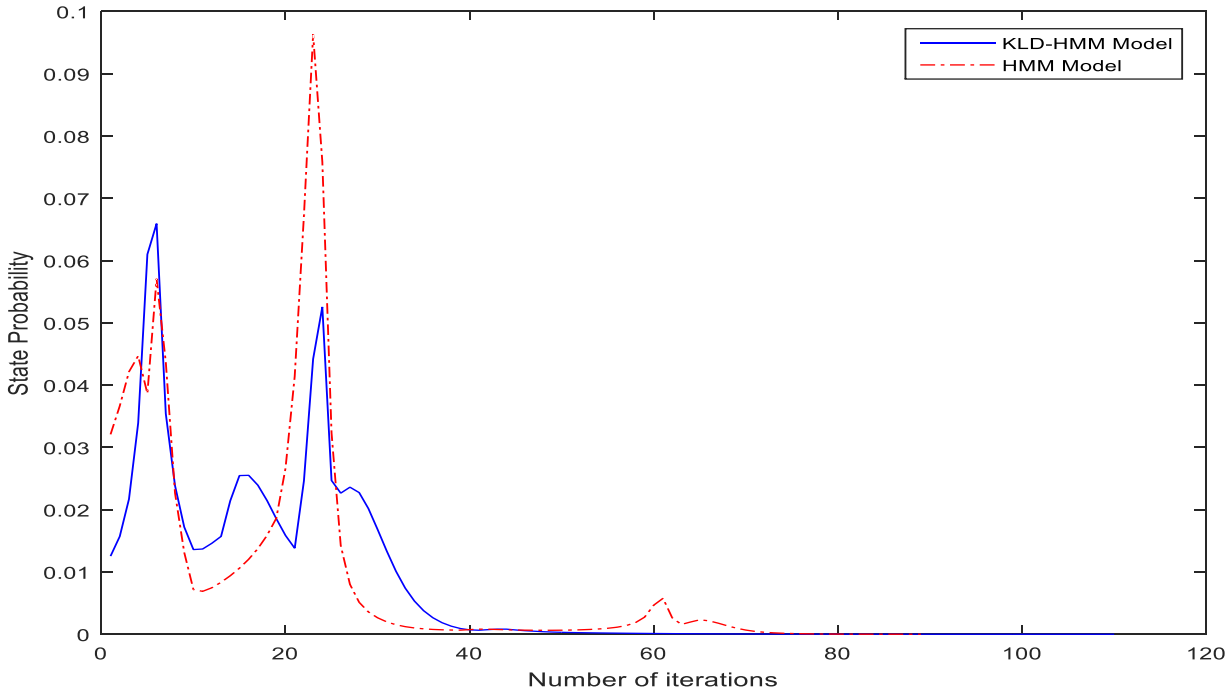


Figure 10: Convergence Rate of the Transition Probabilities of the HMM and KLD-HMM

Figure 11 depicts the comparison of the distribution of the occurrence frequency per state of HMM and KLD-HMM. The graph also shows that the KLD-HMM performance is better.

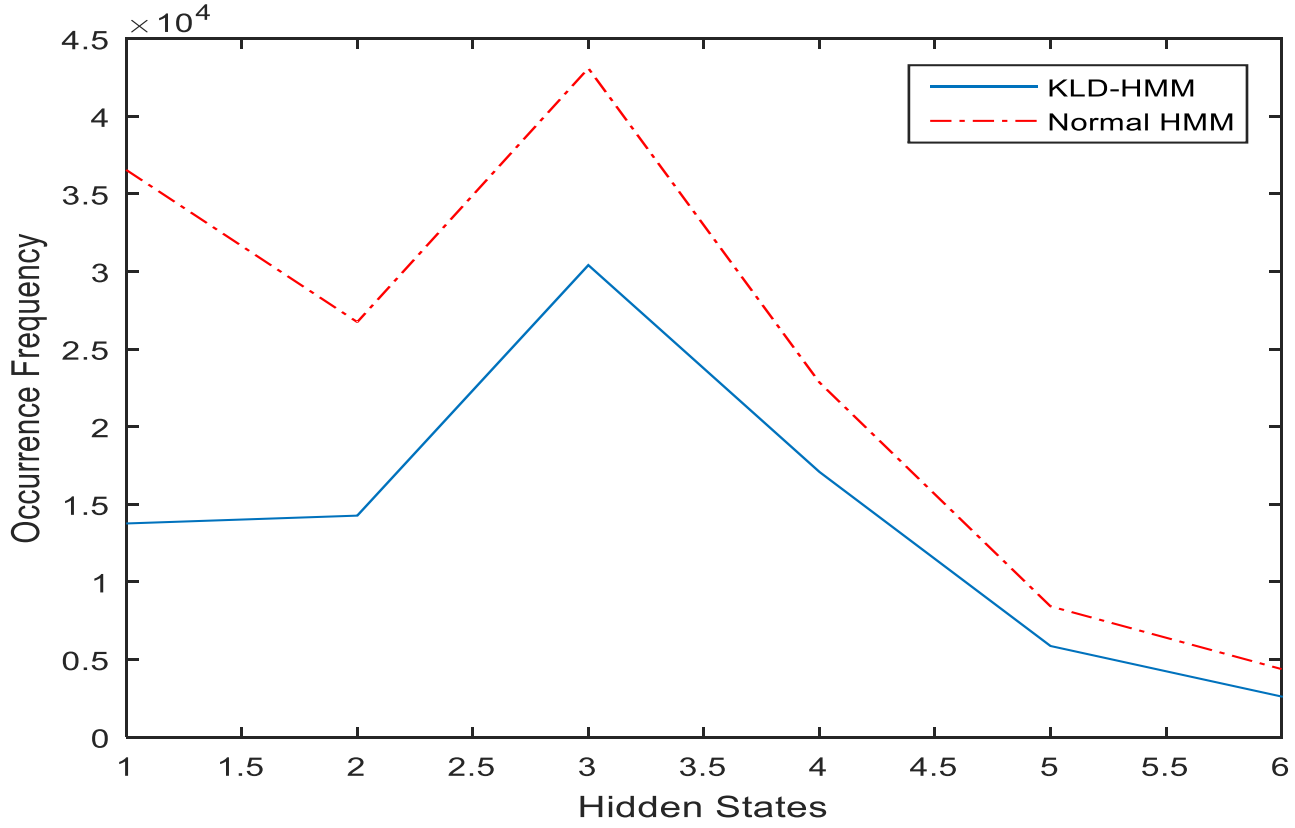


Figure 11: Frequency distribution per state of the HMM and KLD-HMM models

Figure 12 is the Receiver Operator Characteristics (ROC) curve of the test data, is a graphical metric that illustrates the performance of a classifier which in our case is an HMM that classifies packet sequence as "threat" or "normal traffic". The plot shows the rate of prediction as against false alarm rate. The curve with the continual variation in dotted line depicts the plot of the HMM model while the one in solid line is for the KLD-HMM and they both show the approximate variation between false and true classification of sequence packet data. It is observed from the two curves that the classification in the KLD-HMM is better that that of HMM.

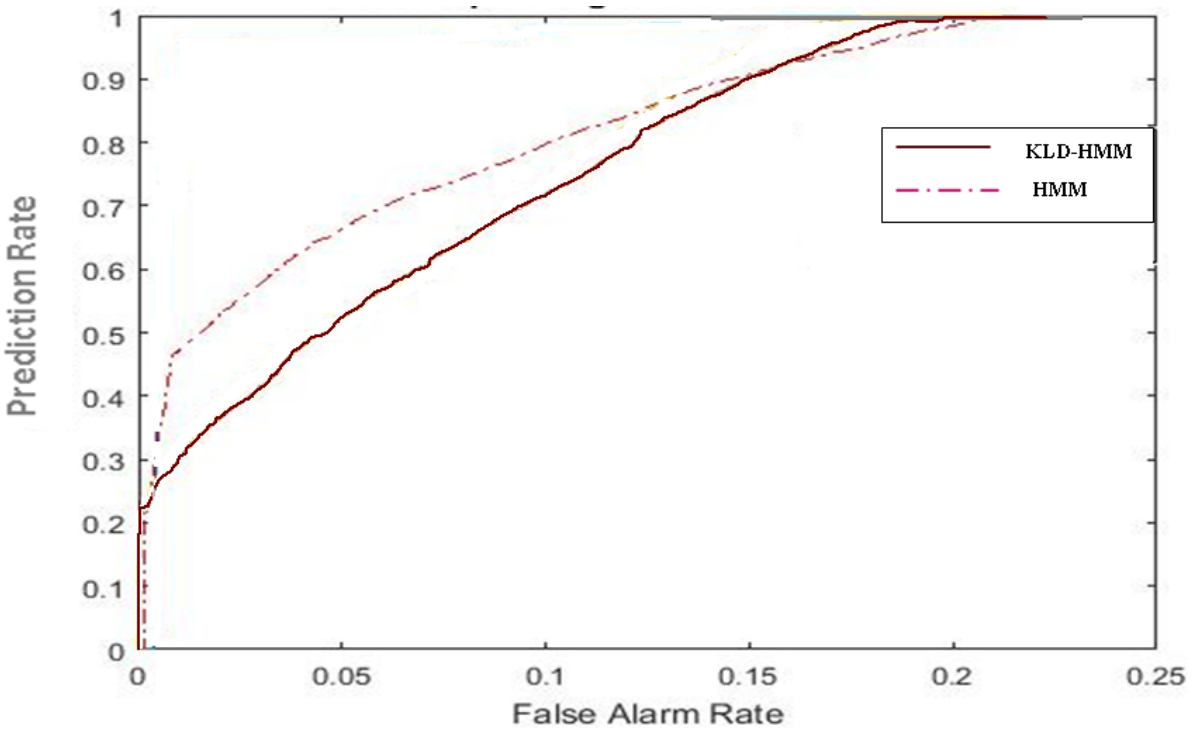


Figure 12: ROC curve of the performance of the HMM and KLD-HMM

4.2.3 VBI-HMM for DDoS Attacks Prediction

Here the resultant reduced parameters of arrived at in the previous section was re-trained with the VBI algorithm (see Algorithm 6). Here, first the Maximum Likelihood (ML) algorithm was run to convergence, and then the VBI algorithm was run from that point in parameter space to convergence. This was achieved by initialising each parameter's variational posterior distribution to be Dirichlet with the ML parameter as the mean. For the VBI algorithm, the prior over each parameter was a symmetric Dirichlet distribution.

Note that as depicted in Table 8, where it takes KLD-HMM about 60 iterations to converge to a local optimum, it takes only about 20 iterations for the VBI optimisation to converge to global optimum.

As shown in Table 8, the computation time was within 18 seconds; there was considerable improvement in the FNR, FPR, TPR, TNR and prediction accuracy to 8%, 11%, 92%, 89% and 91%, respectively. Compared to the previous models constructed in this study, it shows the best performance on all metrics used.

Figure 13 is the ROC curve of the three models. It could be seen that the curve representing the VBI-HMM model shows a less accurate detection rate initially until a threshold (around 0.005) is overcome where the performance of the model becomes excellent. In the development of such DDoS attack prediction system, this threshold value that translates to an improved performance should be taken into account.

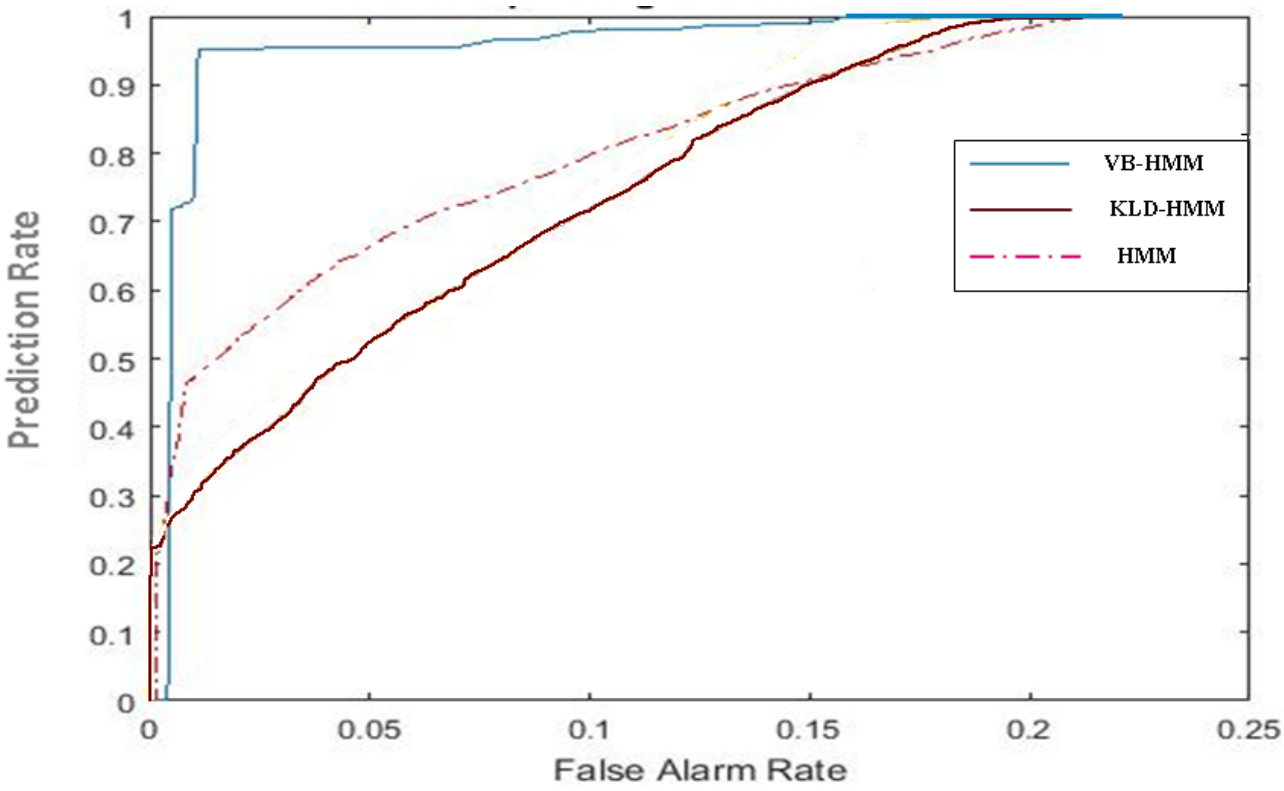


Figure 13: ROC curve of the performance of the HMM, KLD-HMM and VBI-HMM

4.3 Benchmarking of Experimental Results

For objectivity, the experimental results of this study was compared with a similar existing work: the work of Shin *et al.* (2013).

4.3.1 Comparison with *APAN* (Shin *et al.*, 2013)

The work of Shin *et al.* (2013), APAN, was implemented and experimental results from our models were benchmarked with their results. Figure 14 shows the ROC curve of the three models formulated in this study and that of APAN (which was implemented and tested with the same sets of data). The ROC curve shows that the VBI-HMM performs best out of all the models.

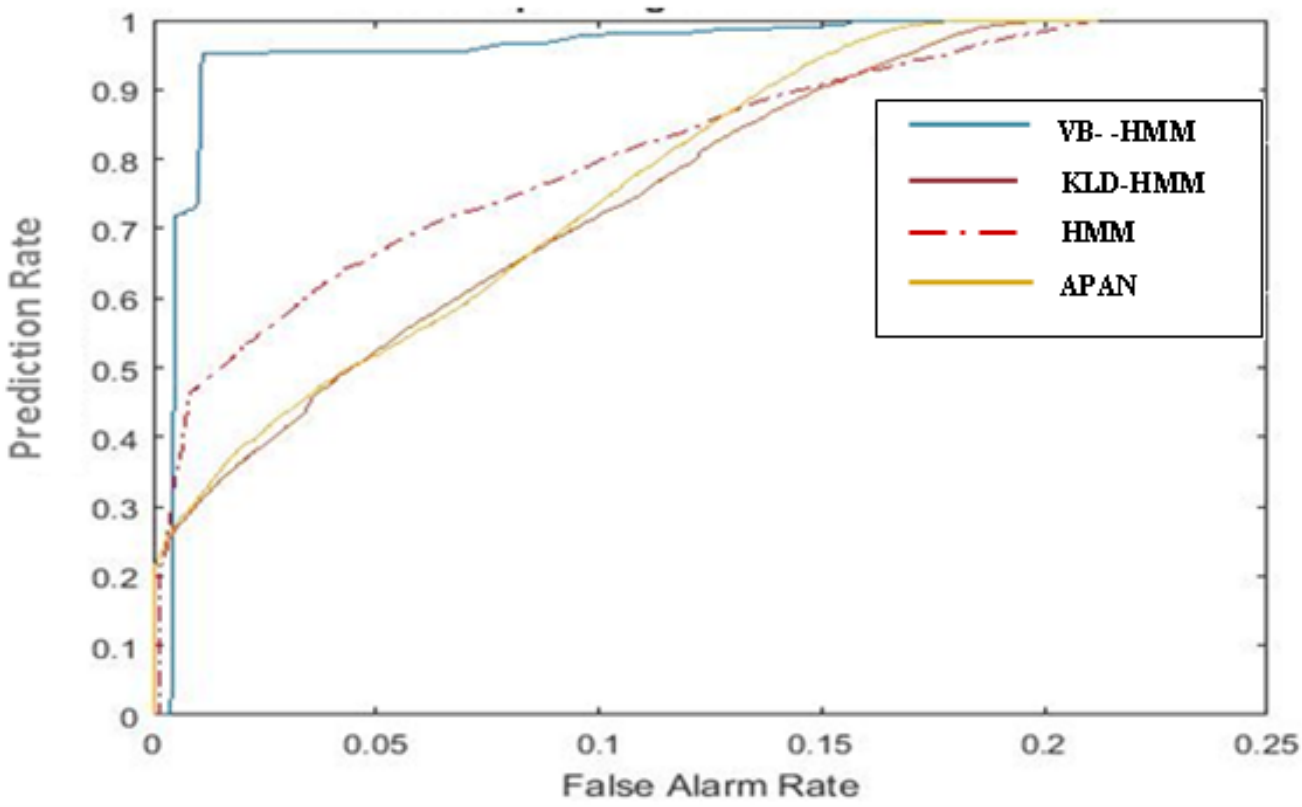


Figure 14: ROC curve of the performance of all the models

Figures 15, 16 and 17 are the graphical representation of the models comparison based on computational time, confusion matrices and prediction accuracy, respectively.

The True positive rate (TPR), False positive rate (FPR), True negative rate (TNR), False negative rate (TNR) and prediction accuracy were all calculated using the formulas in Section 3.5. The combination of the TPR, FPR, TNR and FNR form the confusion matrix for each of the models. The confusion matrices for the models are as in Table 9.

This information depicts the trade-off between each of the models but it can be concluded that the VBI-HMM model is more robust in terms of the prediction accuracy as depicted by its confusion matrix.

Table 9: Confusion Matrices of the Models

MODEL	**CONFUSION MATRIX**
APAN	$\begin{pmatrix} 0.83 & 0.17 \\ 0.22 & 0.78 \end{pmatrix}$
VBI-HMM	$\begin{pmatrix} 0.92 & 0.08 \\ 0.11 & 0.89 \end{pmatrix}$
KLD-HMM	$\begin{pmatrix} 0.84 & 0.16 \\ 0.21 & 0.79 \end{pmatrix}$
HMM	$\begin{pmatrix} 0.77 & 0.23 \\ 0.29 & 0.71 \end{pmatrix}$

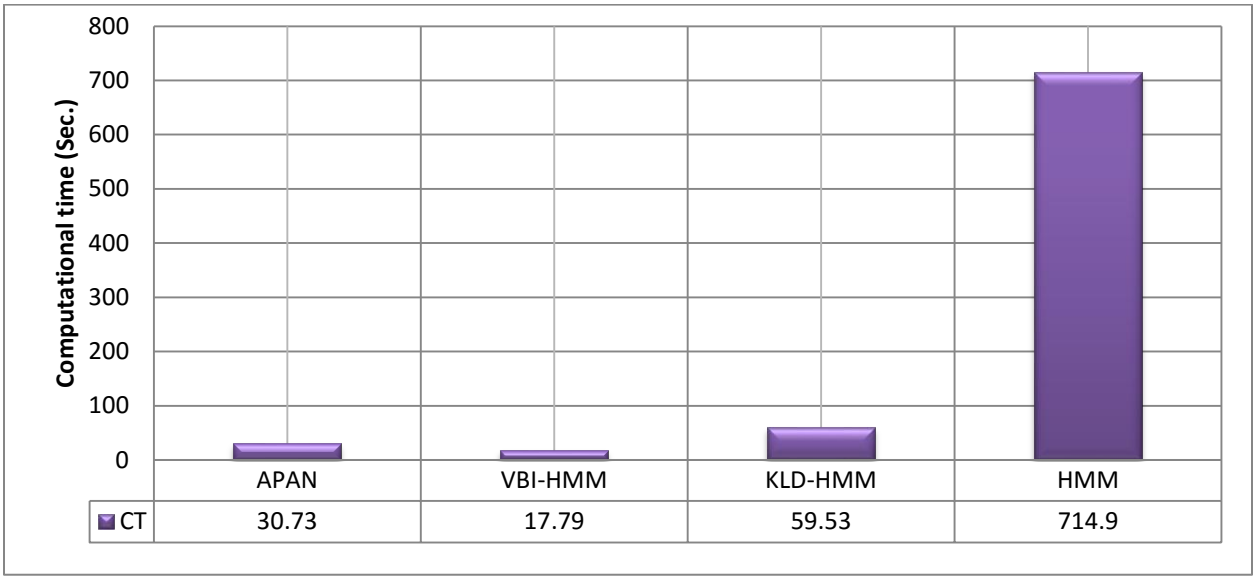


Figure 15: Computational time of the Models

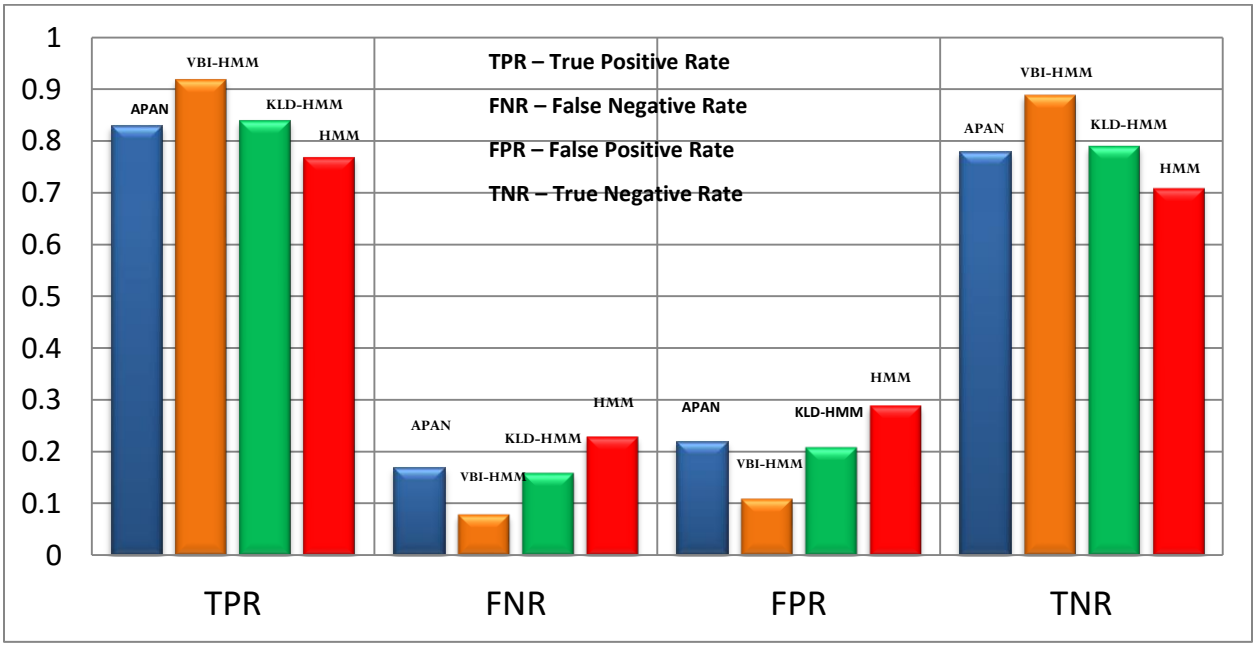


Figure 16: Confusion matrix Comparison of the Models

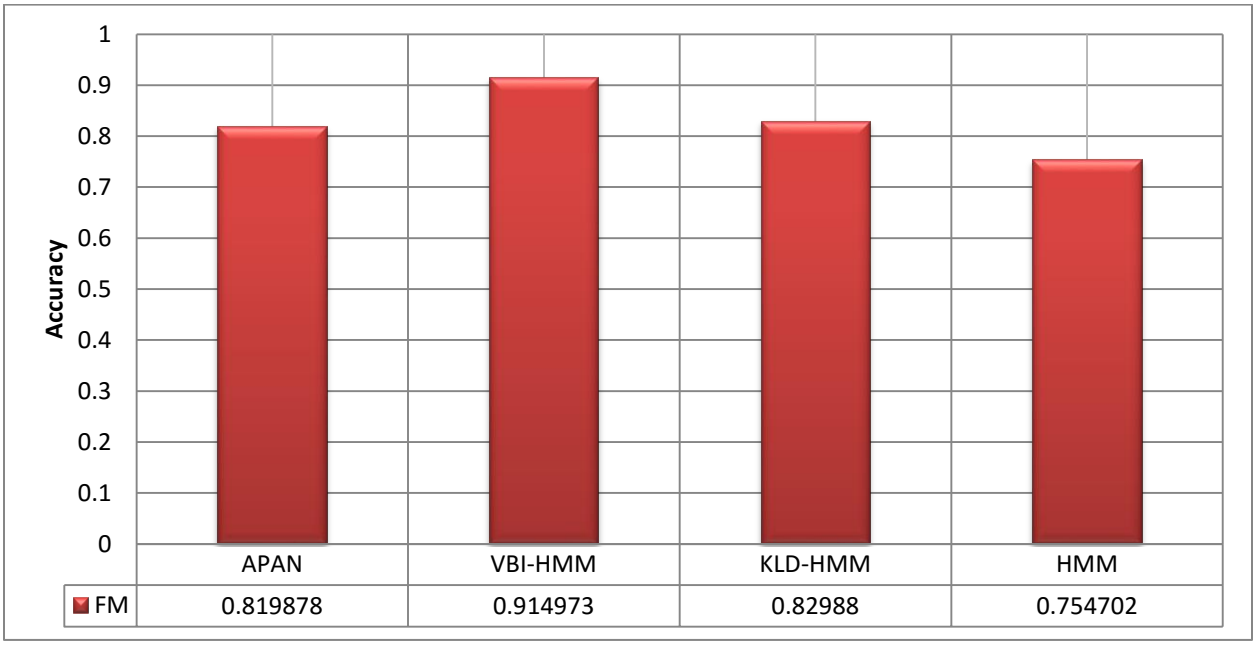


Figure 17: Prediction Accuracy of the Models

The VBI-HMM model performed best in terms of classification of packet sequence as either normal or attack prone. APAN's performance is relatively close to that of KLD-HMM model as depicted.

That is, the prediction rate and false alarm rates for the two are close as depicted by the confusion matrix (see Table 9) and the ROC curve (See Figure 14).

Computationally, VBI-HMM is most efficient as it reduces the propensity for over-fitting data due to model complexity which is not addressed by HMM.

Overall for real time application, the VBI-HMM is recommended for use since it can compute and predict traffic status in a relatively short time. It also ensures the efficiency of prediction over all other models.

CHAPTER FIVE

SUMMARY OF FINDINGS, CONTRIBUTIONS TO KNOWLEDGE CONCLUSION AND POSSIBLE EXTENSIONS

5.1 Summary of Findings

In this study, the problem of Distributed Denial of Service (DDoS) attack prediction has been presented. A Variational Bayesian Inference Hidden Markov Model (VBI-HMM) was developed for real-time prediction of DDoS attacks. The summary of findings is as presented in Table 10.

Table 10: Summary of Findings

S/N	Objectives	Findings
1.	To formulate an entropy-based state-space HMM for DDoS attack prediction.	HMM was formulated with the entropy-based values of the network features forming the observable states and the well-partitioned DDoS phases as hidden states. The model was well able to predict all the DDoS phases.
2.	To reduce the parameter space of the HMM in order to achieve a computationally efficient model for predicting DDoS attacks in network systems	Relative entropy (KLD) was adapted to reduce the observable state space of the original HMM. The training time and the computational time were greatly reduced (Compared to existing model (Shin *et al.*, 2013), 42% improvement was gained).
3.	To implement a global	The VBI-HMM developed was found to predict DDoS

	convergent algorithm for training the formulated HMM in order to reduce its complexity	attacks more efficiently (with 9% improvement). The implementation of the Variational Bayesian Inference algorithm enhanced the reliability of the developed model On evaluation, the prediction accuracy moved up from 74% and 81.5% with HMM and KLD-HMM metrics respectively to 90.5% for the VBI-HMM metrics. Based on the percentage of correctly predicted attack, obtained for VBI-HMM, KLD-HMM and HMM approach, the VBI-HMM outperforms the other HMM variants.

5.2 Conclusion

This study addresses the problem of real-time and accurate prediction of DDoS attack. Techniques for handling DDoS attacks with their various strengths and weaknesses were presented. A robust and efficient architecture for DDoS attack prediction was implemented via simulation experiments using different standard benchmark intrusion specific datasets. Experimental results on the DARPA and CAIDA datasets have shown that the develped model converges faster, which translates into computational efficiency, and shows good performance in predicting attacks compared to earlier models.

5.3 Contributions to Knowledge

The contributions of this thesis to the field of network security and cyber attacks pre-emption are summarised as follows:

1. This study has opened a new research direction to cyber attacks prediction model formulation by developing a novel VBI-HMM for fast and high precision prediction of Distributed Denial of Service (DDoS) attack.

2. This study has adapted Kullback-Liebler Divergence (KLD) concept to achieve a parsimonious model that minimises the entropy of states of HMM to accelerate DDoS prediction.

3. The developed VBI-HMM algorithm exhibits a novel ability of avoiding being trapped in a local maximum.
4. The hybrid VBI-HMM scheme developed in this study is more efficient than existing DDoS prediction tools.

5.4 Possible Extension

Some recommendations for further extension of DDoS attack prediction are as presented below:

1. The normalised entropy values of the network traffic features can be used to set calibrated alarm for warning and taking actions to pre-empt DDoS attacks.
2. A new learning paradigm called deep learning can be explored in training the model. This can be done either through software approach or hardware approach.
3. The model can further be extended to other types of Denial of Service (DoS) attacks and intrusion attacks in general.

4. Other forms of hidden Markov models, for instance, Hierarchical Hidden Markov Model (HHMM), can be considered for the model formulation.

REFERENCES

Abdullah, A., Pillai, T. R., Zheng, C. L., and Abaeian, V. (2015). Intrusion detection forecasting using time series for improving cyber defence. *International Journal of Intelligent Systems and Applications in Engineering*, *3*(1), 28-33

Aczél, J. and Daróezy, Z. (1975). *On measures of information and their characterizations*, New York-San Francisco-London. Academic Press. XII, 234 S., (Mathematics in Science and Engineering 115)

Afolorunso, A. A., Adewole, A. P., Abass, O. and Longe, H. O. D. (2016). Kullback-Liebler divergence for reducing the observable states space of hidden Markov model for predicting distributed denial of service attack. *11th Unilag Conference and Fair*, Lagos, Nigeria, 184-193.

Ahmadi, M. R. (2009). An Intrusion prediction technique based on co-evolutionary immune system for network security (CoCo-IDP). *International Journal of Network Security, 9(3),* 290–300.

Alenezi, M. and Reed, M. J (2012). Methodologies for detecting DoS/DDoS attacks against network servers. *The Seventh International Conference on Systems and Networks Communication* (*ICSNC 2012),* 92-98.

Ankali, S. B. and Ashoka, D. V. (2011). Detection Architecture of Application Layer DDoS Attack for Internet . *International Journal of Advanced Networking and Applications, 3*(01), 984-990.

Badajena, J. C. and Rout, C. (2012). Incorporating hidden Markov model into anomaly detection technique for network intrusion detection. *International Journal of Computer Applications*. *53*(11), 42-47.

Beal, M. (2003). *Variational algorithms for approximate bayesian inference*. Unpublished Ph.D. thesis, The Gatsby Computational Neuroscience Unit, University College London.

Berezinski, P., Jasiul, B. and Szpyrka, M. (2015). An entropy-based network anomaly detection method. *Entropy 2015*, *17*, 2367-2408. doi:10.3390/e17042367

Bunke, H. and Caelli, T. (Eds.). (2001). *Hidden Markov models: Applications in computer vision.* World Scientific, Series in Machine Perception and Artificial Intelligence, *45*.

Chen, S., Zuo, Z., Huang, Z. P. and Guo, X. J. (2016). A graphical feature generation approach for intrusion detection. *MATEC Web of Conferences*. 44, 02041. doi: 10.1051/matecconf/20164402041

Cheng, X. and Yangdan, N. (2012). The research on dynamic risk assessment based on hidden Markov models. *International Conference on Computer Science and Service System* (*CSSS)*, Nanjing, China, 1106-1109. doi:10.1109/CSSS.2012.280

Chung, Y., Kim, I., Lee, C., Im, E. G. and Won, D. (2006) Design of an on-line intrusion forecast system with a weather forecasting model. In: Gavrilova M.L. *et al.* (Eds.) *Computational Science and Its Applications - ICCSA 2006. ICCSA 2006*. Lecture Notes in Computer Science, 3983, 777-786. Springer, Berlin, Heidelberg. doi: 10.1007/11751632_84

Clausius, R.; Hirst, T. (1867). *The Mechanical Theory of Heat: With its applications to the steam-engine and to the physical properties of bodies*. J. van Voorst: London, UK.

Cover, T. and Thomas, J. (2006). *Elements of information theory*. Wiley: Hoboken, NJ, USA.

Cuppens, F. (2001). Managing alerts in a multi-intrusion detection environment. In ACSAC '01 *Proceedings of 17th Annual Computer Security Applications Conference.* Retrieved 12th November, 2016 from https://www.acsac.org/2001/papers/70.pdf

Debar, H., Dacier, M. and Wespi, A. (1999). Towards a taxonomy of intrusion detection systems. Computer Networks, *31*(8), 805-822

Devika, U. K. and Sunny, D. (2014). Comparison study among various anomaly detection techniques. *International Journal of Research in Computer and Communication Technology*, 121 - 124.

Divya T. and Muniasamy, K. (2015). Real-time intrusion prediction using hidden Markov model with genetic algorithm. In: Suresh, L., Dash, S. and Panigrahi B. (Eds.) *Artificial intelligence and evolutionary algorithms in engineering systems*. Advances in Intelligent Systems and Computing, *324*, 731-736. Springer, New Delhi. doi: 10.1007/978-81-322-2126-5_78

Fox, C. W. and Roberts, S. J. (2011). A tutorial on variational Bayesian inference. In: *Artificial Intelligence Review*. *38*(2), 85–95. doi:10.1007/s10462-011-9236-8

Govindu, S. K. (2005). Intrusion forecasting system. Retrieved 11th April, 2011 from http://www.securitydocs.com/library/3110

Grandison, T. and Terzi, E. (2007). Intrusion detection technology. *Encyclopedia of Database Systems*, 1568-1570. doi: 10.1007/978-0-387-39940-9_209.

Gruhl, C. and Sick, B. (2016). Variational Bayesian Inference for Hidden Markov Models With Multivariate Gaussian Output Distributions. Retrieved 4th December, 2016 from https://arxiv.org/pdf/1605.08618.pdf

Haq, N. F., Onik, A. R., Hridoy, M. A K., Rafni,, M. Shah, F. M., and Farid, D. M. (2015). Application of machine learning approaches in intrusion detection system: A survey. *International Journal of Advanced Research in Artificial Intelligence, 4*(3), 9 - 18.

Haslum, K., Moe, M. E. G. and Knapskog S. J. (2008). Real-time intrusion prevention and security analysis of networks using HMMs. *33rd IEEE Conference on Local Computer Networks*, Montreal, Canada, 927-934. **doi:** 10.1109/LCN.2008.4664305

Hilton, S. (2016). Dyn analysis summary of Friday October 21 attack [Blog post]. Retrieved 8th December, 2016 from http://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/

Ibe, O. C. (2013). *Markov processes for stochastic modelling* (2nd ed.). Burlington, MA: Elsevier Academic Press.

Jemili F., Zaghdoud, M. and Ahmed M. B. (2006). DIDFAST.BN: Distributed intrusion detection and forecasting multiagent system using Bayesian network. *2nd Information and Communication Technologies. ICTTA '06*. doi: 10.1109/ICTTA.2006.1684901

Jemili, F., Zaghdoud, M. and Ahmed, M. B. (2009*).* Hybrid intrusion detection and prediction multiagent system, HIDPAS. *(IJCSIS) International Journal of Computer Science and Information Security,5*(1), 62-71

Kang, M-J. and Kang, J-W. (2016). Intrusion detection system using deep neural network for in-vehicle network security. *PLoS One*, *11*(6). doi: 10.1371/journal.pone.0155781

Kannadiga, P., Zulkernine, M., Haque, A. (2007). E-NIPS: An event-based network intrusion prediction system. *In Proceedings of 10th Information Security Conference*, 37-52. doi: 10.1007/978-3-540-75496-1_3

Karmeshu, J. (2003). *Entropy measures, maximum entropy principle and emerging applications*. Springer: New York, NY, USA.

Karthikeyan, K. R. and Indra, A. (2010). Intrusion detection tools and techniques – A survey. *International Journal of Computer Theory and Engineering*, *2*(6), 901 - 906.

Kim, S., Shin, S., Kim, H., Kwon, K. and Han, Y. (2008). Hybrid intrusion forecasting framework for early warning system. *IEICE TRANS. INF. & SYST*., *E91–D*(5), 1234–1241

Khosronejad, M., Sharififar, E., Torshizi, H. A. and Jalali, M. (2013). Developing a hybrid method of hidden Markov models and C5.0 as a intrusion detection system, *International Journal of Database Theory and Application*, *6*(5), 165-174.

Kumar, G. (2014). Evaluation metrics for intrusion detection systems - A study. International *Journal of Computer Science and Mobile Applications*, *2*(11), 11-17

Kumar, P. and Ravi, V. (2007). Bankruptcy prediction in banks and firms via statistical and intelligent techniques. *European Journal of Operational Research*, *180*(1), 1-28

Kwon, D., Hong, J. W, and Ju, H. (2012). DDoS attack forecasting system architecture using Honeynet. *14th Asia-Pacific Network Operations and Management Symposium (APNOMS).* doi**:** 10.1109/APNOMS.2012.6356055

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

Lee S.Y., Low W.L., Wong P.Y. (2002). *Learning fingerprints for a database intrusion detection system*. In: Gollmann D., Karjoth G., Waidner M. (Eds.) Computer Security — ESORICS 2002. ESORICS 2002. Lecture Notes in Computer Science, *2502*, 264-280.. Springer, Berlin, Heidelberg. doi: 10.1007/3-540-45853-0_16

Lin, S. C. and Tseng S. S. (2004). Constructing detection knowledge for DDoS intrusion tolerance. *Expert Systems with Application. 27*(3), 379-390

Lundin, E., and Jonsson, E. (2002). Survey of research in the intrusion detection area, Technical report 02-04, Department of Computer Engineering, Chalmers University of Technology, Goteborg. Retrieved 15th May, 2015 from http://www.ce.chalmers.se/staff/emilie/papers/Lundin_survey02.pdf.

MacKay, D. J. C. (1997). Ensemble learning for hidden Markov models. Technical report, Cavendish Laboratory, University of Cambridge.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations, *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability.* Berkeley, University of California Press, *1*, 281-297

Maskat K., Mohd Shukran M., Khairuddin M. A. and Mohd Isa M, (2011). Mobile agents in intrusion detection system: Review and analysis. *Modern Applied Science, 5*(6), 218-231

MIT Lincoln Lab (2000). DARPA intrusion detection scenario specific datasets. Downloaded 5th July, 2014 from <http://www.ll.mit.edu/IST/ideval/data/2000/2000_data_index.html>.

MIT Lincoln Lab (1999). DARPA intrusion detection scenario specific datasets. Downloaded 5th July, 2014 from http://www.ll.mit.edu/IST/ideval/data/1999/1999_data_index.html.

Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., and Weaver, N. (2003). Inside the slammer worm. *IEEE Security & Privacy*, *99*(4), 33 - 39.

Moore, D., Shannon, C., and Brown, J. (2002). Code-Red: A case study on the spread and victims of an Internet worm. *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement*, Marseille, France, 273-284.

Murphy, K. P. (2002). *Dynamic Bayesian networks: Representation, inference and learning*. Ph.D. Thesis, University of California, Berkeley.

Pontes, E. (2012). Distributed multiagent intrusion forecasting system (DMIFS) based prediction models for cyber attacks detection system. In *Proceedings of the 7th European Symposium on Research in Computer Security*, Zurich, Switzerland, 264-280.

Pontes, E. and Guelfi, A. E. (2010). Cooperative architecture applied for distributed intrusion forecasting systems (DIFS). *Journal of Information Security Research*, *1*(1), 11-18

Prasad, K. M., Reddy, A. R. M. and Rao, K. V. (2014). DoS and DDoS Attacks: Defense, Detection and Traceback Mechanisms - A Survey. *Global Journal of Computer Science and Technology (E), 14*(7), 15-32.

Rabiner, L.R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286.

Ramasubramanian, P. and Kannan, A. (2004). Quickprop Neural Network Ensemble Forecasting Framework for a Database Intrusion Prediction System, *Neural Information Processing - Letters and Reviews. 5*(1), 9-18.

Rao, P. R. M., Reddy, K.V. and Hemanth, S. V. (2012). Minimizing application layer DDoS attacks using website customization. *International Journal of Computer Science and Technology*. *3*(4), 838-841.

Rodrıguez, L. J. and Torres, I. (2003). Comparative Study of the Baum-Welch and Viterbi Training Algorithms Applied to Read and Spontaneous Speech Recognition. In: Perales F.J., Campilho A.J.C., de la Blanca N.P., Sanfeliu A. (Eds.) *Pattern Recognition and Image Analysis*. IbPRIA 2003. Lecture Notes in Computer Science, vol. 2652. Springer, Berlin, Heidelberg. doi: 10.1007/978-3-540-44871-6_98

Saganowski Ł., Goncerzewicz M., Andrysiak T. (2013). Anomaly detection preprocessor for SNORT IDS system. In: Choraś R. (Eds.) *Image Processing and Communications Challenges 4*. Advances in Intelligent Systems and Computing, Springer, Berlin, Heidelberg. *184*, 225-232. doi: 10.1007/978-3-642-32384-3_28

Saini, P. and Godara, S. (2014). Modelling intrusion detection system using hidden Markov model: A review. *International Journal of Advanced Research in Computer Science and Software Engineering*, *4*(6), 542-547. (Available online at: www.ijarcsse.com)

Satpute K., Agrawal S., Agrawal J., and Sharma S. (2013). A Survey on Anomaly Detection in Network Intrusion Detection System Using Particle Swarm Optimization Based Machine Learning Techniques. In: Satapathy S., Udgata S., and Biswal B. (Eds.). *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications* (*FICTA)*. Advances in Intelligent Systems and Computing. Springer, Berlin, Heidelberg. *199*, 441-452

Schechter, S. E. (2005).“Toward econometric models of the security risk from remote attacks. *IEEE Security & Privacy*, *3*(1), 40–44.

Sendi, S., Dagenais, M. Jabbarifar, M. and Couture, M. (2012). Real time intrusion prediction based on optimized alerts with hidden Markov model. *Journal of Networks*, *7*(2), 311-321

Seng, J. L. and Chen, T. C. (2010). An analytic approach to select data mining for business decision. *Expert Systems with Applications*, *37*(12), 8042-8057.

Shah, A, A., Khiyal, M. S. H., Awan, M. D. (2015). Analysis of machine learning techniques for intrusion detection system: A review. *International Journal of Computer Applications*, *119*(3), 19 - 29.

Shannon, C. E. (1948). *A Mathematical Theory of Communication.* Bell Syst. Tech. J. *27*, 379–423.

Sharma, S. and Gupta, R. K. (2015). Intrusion detection system: A review. *International Journal of Security and Its Applications 9*(5), 69-76

Sharma, S. K. and Manoria, M. (2015). Intrusion detection using hidden Markov model. *International Journal of Computer Applications.* *115*(4), 35-38.

Shin, S., Lee, S., Kim, H. and Kim, S. (2013). Advanced probabilistic approach for network intrusion forecasting and detection. *Expert Systems with Applications*. *40*(1), 315-322

Sodiya, A. S., Longe, H. O. D. and Akinwale, A. T. (2004). A new two-tiered strategy to intrusion detection. *Information Management and Computer security*, *12*(1), 27-44.

Sodiya, A. S., Adeniran, O. and Ikuomola A. J. (2007). An expert system-based site security officer, *Journal of Computing and Information Technology - CIT 15*(3): 227–235.

Thanthrige, U. S. K. P. M., Samarabandu, J. and Wang, X. (2016). Intrusion alert prediction using a hidden Markov model. Retrieved 10th January, 2017 from https://arxiv.org/pdf/1610.07276

Warrender, C., Forrest, S. and Pearlmutter, B. (1999). Detection of intrusion using system calls: Alternative data models[C]. *IEEE Symposium on Security and Privacy*. Retrieved 16th July, 2015 from www.researchgate.net/publication/2448365_Detecting_Intrusions_Using_System_Calls_Alternative_Data_Models

Wu S. X. and Banzh F. W. (2010). The use of computational intelligence in intrusion detection systems: A review. *Applied Soft Computing Journal*, *10*(1), 1-35

Xiao, J., Zou, L. and Li, C. (2007). Optimization of hidden Markov model by a genetic algorithm for web information extraction. *International Journal of Computational Intelligence Systems*. doi: 10.2991/iske.2007.48

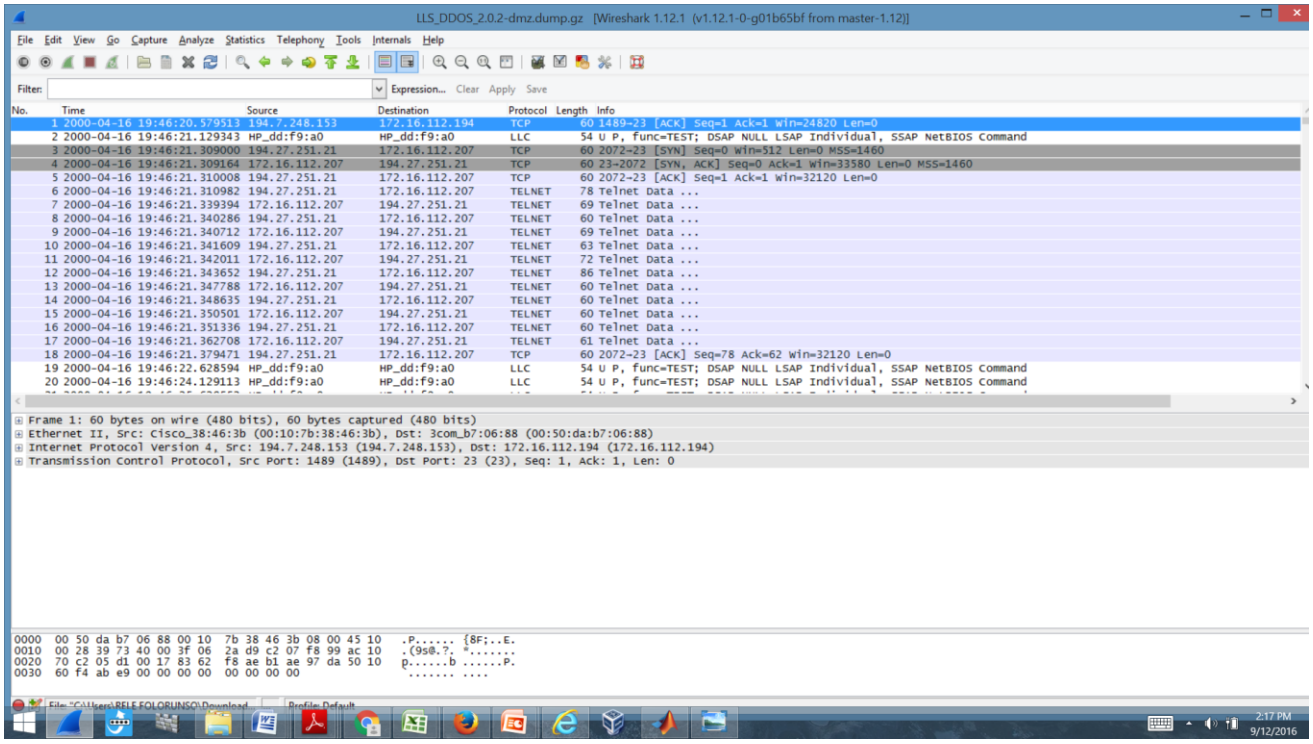
York, K. (2016). Dyn statement on 10/21/2016 DDoS attack [Blog post]. Retrieved 8th December, 2016 from http://dyn.com/blog/dyn-statement-on-10212016-ddos-attack/

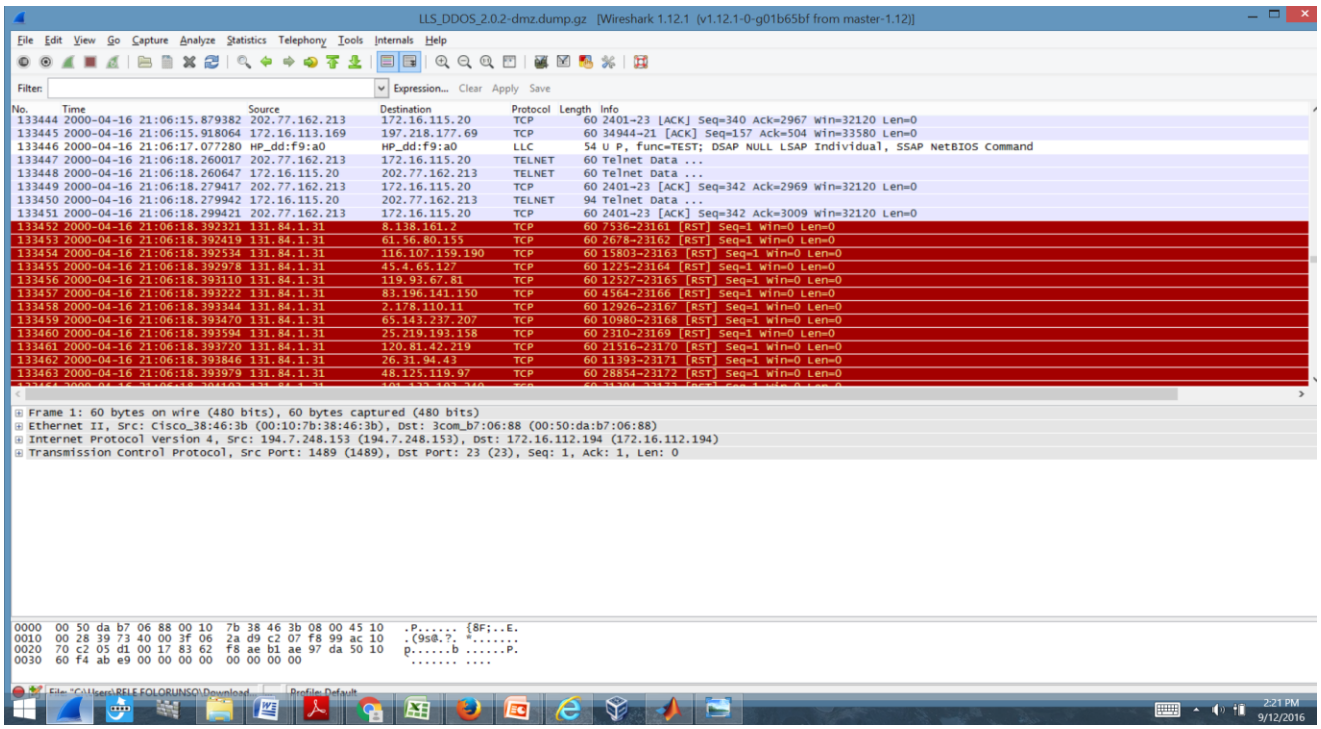
Zhang, C., Jiang, J., and Kamel, M. (2005). Intrusion detection using hierarchical neural networks. *Pattern Recognition letters*, *26*(6), 779-791.

Zhang, X., Jia, L., Shi, H., Tang, Z. and Wang, X. (2012). The application of machine learning methods to intrusion detection. *Congress on Engineering and Technology* (*S-CET)*, Spring, 1-4. doi**:** 10.1109/SCET.2012.6341943

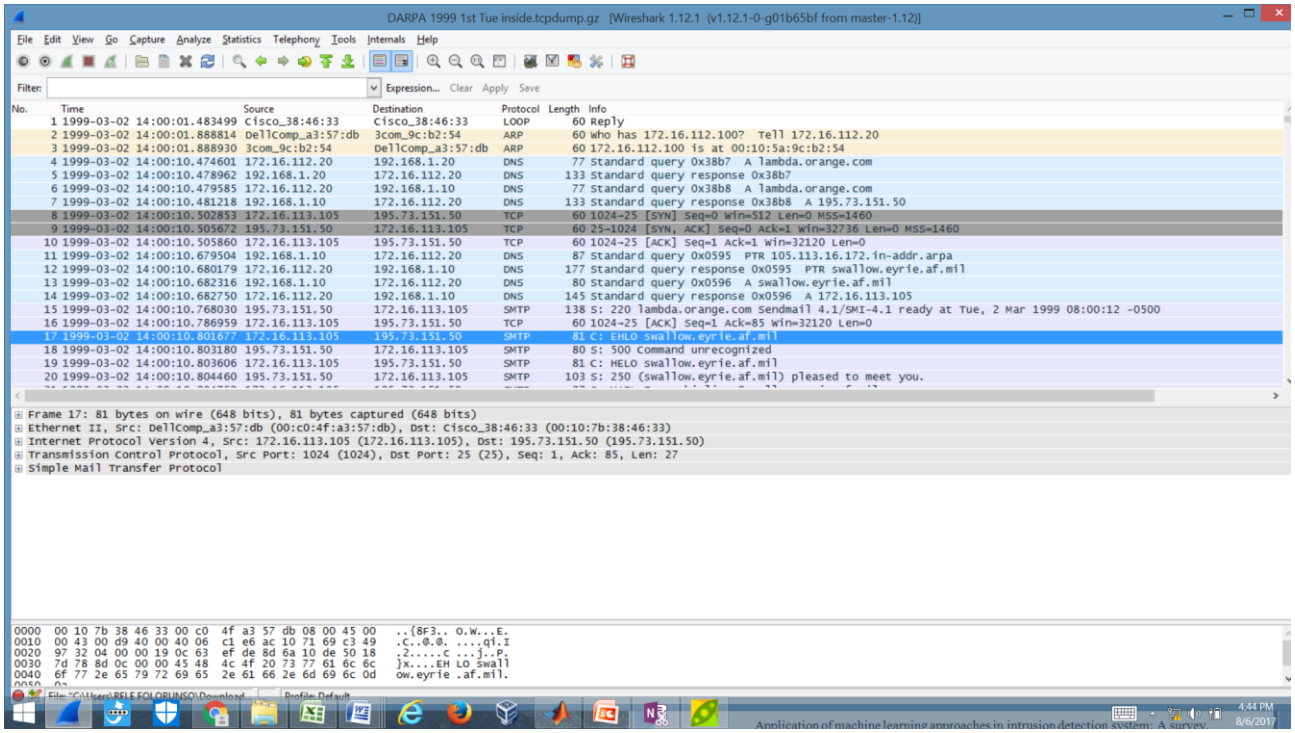
APPENDIX A: Experimental Datasets

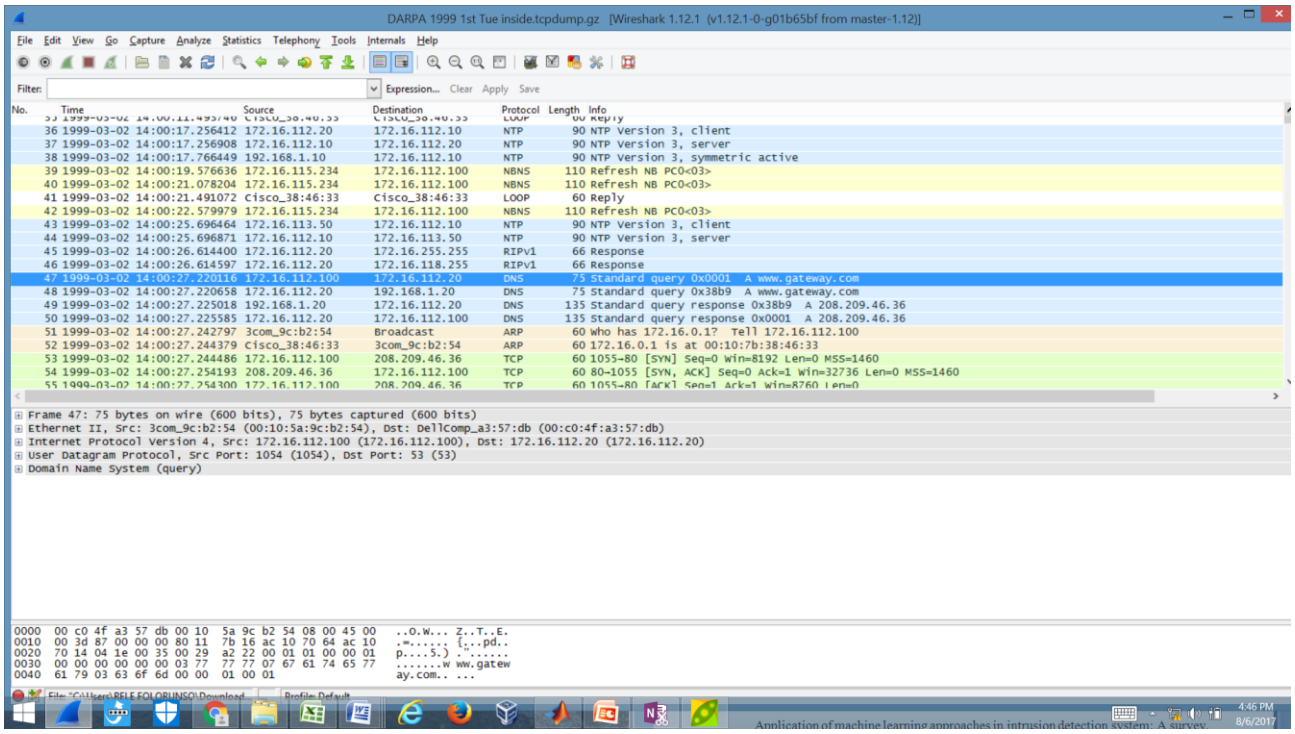
The Screenshots of the DARPA 2000 Dataset



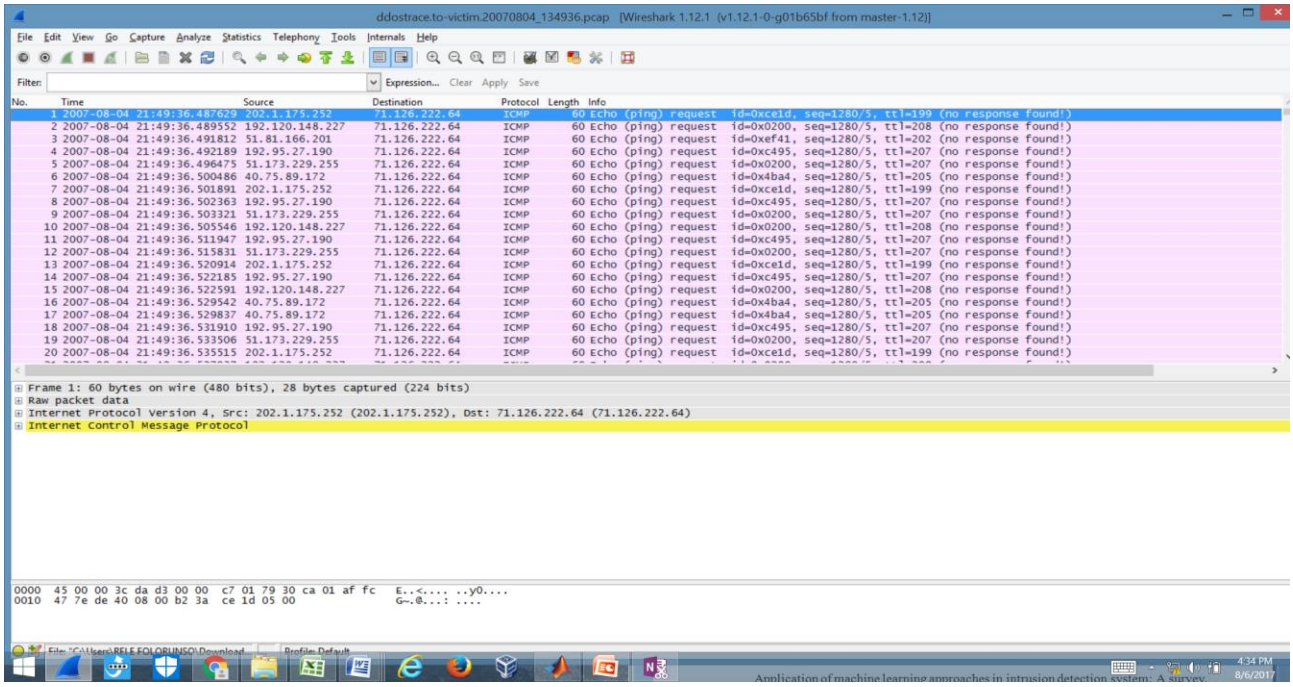


The Screenshots of the DARPA 1999 Dataset





The Screenshots of the CAIDA 2007 DDoS Dataset



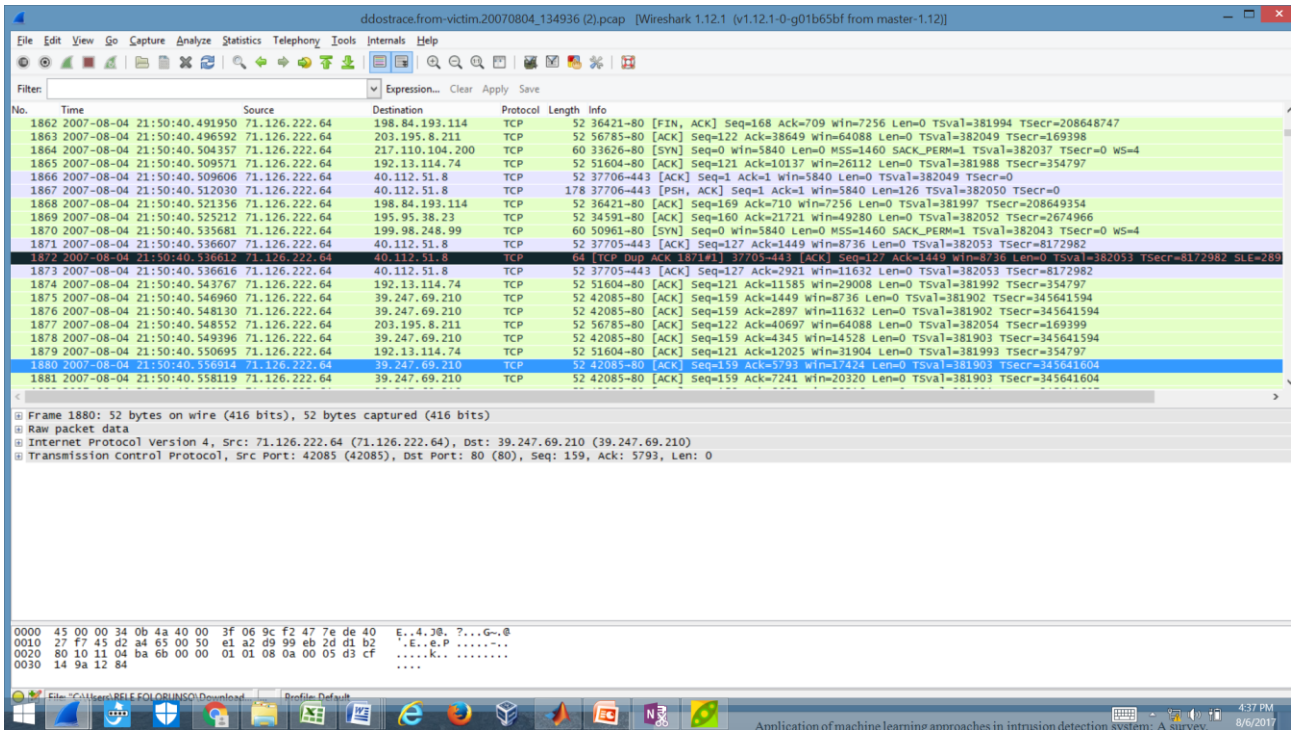


Table 8: Subset of the Experimental Data in Excel Format

Frame No	Time	Source	Destination	Protocol	Length	SrcIP	DstIP	pSrcPort	pDestPort	PacketType
1	19:46:21	194.7.248.153	172.16.112.194	TCP	60	194.7.248.153	172.16.112.194	1489	23	TCP
2	19:46:21	HP_dd:f9:a0	HP_dd:f9:a0	LLC	54	194.7.248.153	172.16.112.194	1489	23	TCP
3	19:46:21	194.27.251.21	172.16.112.207	TCP	60	194.27.251.21	172.16.112.207	2072	23	TCP
4	19:46:21	172.16.112.207	194.27.251.21	TCP	60	172.16.112.207	194.27.251.21	23	2072	TCP
5	19:46:21	194.27.251.21	172.16.112.207	TCP	60	194.27.251.21	172.16.112.207	2072	23	TCP
6	19:46:21	194.27.251.21	172.16.112.207	TELNET	78	194.27.251.21	172.16.112.207	2072	23	TCP
7	19:46:21	172.16.112.207	194.27.251.21	TELNET	69	172.16.112.207	194.27.251.21	23	2072	TCP
8	19:46:21	194.27.251.21	172.16.112.207	TELNET	60	194.27.251.21	172.16.112.207	2072	23	TCP
9	19:46:21	172.16.112.207	194.27.251.21	TELNET	69	172.16.112.207	194.27.251.21	23	2072	TCP
10	19:46:21	194.27.251.21	172.16.112.207	TELNET	63	194.27.251.21	172.16.112.207	2072	23	TCP
11	19:46:21	172.16.112.207	194.27.251.21	TELNET	72	172.16.112.207	194.27.251.21	23	2072	TCP
12	19:46:21	194.27.251.21	172.16.112.207	TELNET	86	194.27.251.21	172.16.112.207	2072	23	TCP
13	19:46:21	172.16.112.207	194.27.251.21	TELNET	60	172.16.112.207	194.27.251.21	23	2072	TCP
14	19:46:21	194.27.251.21	172.16.112.207	TELNET	60	194.27.251.21	172.16.112.207	2072	23	TCP
15	19:46:21	172.16.112.207	194.27.251.21	TELNET	60	172.16.112.207	194.27.251.21	23	2072	TCP
16	19:46:21	194.27.251.21	172.16.112.207	TELNET	60	194.27.251.21	172.16.112.207	2072	23	TCP
17	19:46:21	172.16.112.207	194.27.251.21	TELNET	61	172.16.112.207	194.27.251.21	23	2072	TCP
18	19:46:21	194.27.251.21	172.16.112.207	TCP	60	194.27.251.21	172.16.112.207	2072	23	TCP
19	19:46:23	HP_dd:f9:a0	HP_dd:f9:a0	LLC	54	194.27.251.21	172.16.112.207	2072	23	TCP
20	19:46:24	HP_dd:f9:a0	HP_dd:f9:a0	LLC	54	194.27.251.21	172.16.112.207	2072	23	TCP
21	19:46:26	HP_dd:f9:a0	HP_dd:f9:a0	LLC	54	194.27.251.21	172.16.112.207	2072	23	TCP
22	19:46:27	HP_dd:f9:a0	HP_dd:f9:a0	LLC	54	194.27.251.21	172.16.112.207	2072	23	TCP
23	19:46:29	HP_dd:f9:a0	HP_dd:f9:a0	LLC	54	194.27.251.21	172.16.112.207	2072	23	TCP
24	19:46:29	Cisco_38:46:3b	Cisco_38:46:3b	LOOP	60	194.27.251.21	172.16.112.207	2072	23	TCP
25	19:46:30	HP_dd:f9:a0	HP_dd:f9:a0	LLC	54	194.27.251.21	172.16.112.207	2072	23	TCP
26	19:46:31	172.16.115.20	192.168.1.10	DNS	76	172.16.115.20	192.168.1.10	57205	53	UDP
27	19:46:31	192.168.1.10	172.16.115.20	ICMP	104	192.168.1.10	172.16.115.20	25	63336	ICMP
28	19:46:32	HP_dd:f9:a0	HP_dd:f9:a0	LLC	54	192.168.1.10	172.16.115.20	25	63336	ICMP
29	19:46:33	HP_dd:f9:a0	HP_dd:f9:a0	LLC	54	192.168.1.10	172.16.115.20	25	63336	ICMP
30	19:46:34	172.16.115.20	195.115.218.20	DNS	76	172.16.115.20	195.115.218.20	57224	53	UDP
31	19:46:34	195.115.218.20	172.16.115.20	DNS	165	195.115.218.20	172.16.115.20	53	57224	UDP
32	19:46:34	172.16.112.50	195.115.218.108	TCP	74	172.16.112.50	195.115.218.108	63330	25	TCP
33	19:46:34	195.115.218.108	172.16.112.50	TCP	60	195.115.218.108	172.16.112.50	25	63330	TCP
34	19:46:34	172.16.112.50	195.115.218.108	TCP	60	172.16.112.50	195.115.218.108	63330	25	TCP
35	19:46:34	195.115.218.108	172.16.112.50	SMTP	138	195.115.218.108	172.16.112.50	25	63330	TCP
36	19:46:34	172.16.112.50	195.115.218.108	SMTP	68	172.16.112.50	195.115.218.108	63330	25	TCP
37	19:46:34	195.115.218.108	172.16.112.50	SMTP	90	195.115.218.108	172.16.112.50	25	63330	TCP
38	19:46:34	172.16.112.50	195.115.218.108	SMTP	83	172.16.112.50	195.115.218.108	63330	25	TCP
39	19:46:34	195.115.218.108	172.16.112.50	SMTP	90	195.115.218.108	172.16.112.50	25	63330	TCP
40	19:46:34	172.16.112.50	195.115.218.108	SMTP	87	172.16.112.50	195.115.218.108	63330	25	TCP

41	19:46:34	195.115.218.108	172.16.112.50	SMTP	86	195.115.218.108	172.16.112.50	25	63330	TCP
42	19:46:34	172.16.112.50	195.115.218.108	SMTP	60	172.16.112.50	195.115.218.108	63330	25	TCP
43	19:46:34	195.115.218.108	172.16.112.50	SMTP	104	195.115.218.108	172.16.112.50	25	63330	TCP
44	19:46:34	172.16.112.50	195.115.218.108	SMTP	801	172.16.112.50	195.115.218.108	63330	25	TCP
45	19:46:34	195.115.218.108	172.16.112.50	TCP	60	195.115.218.108	172.16.112.50	25	63330	TCP
46	19:46:34	172.16.112.50	195.115.218.108	IMF	60	172.16.112.50	195.115.218.108	63330	25	TCP
47	19:46:34	195.115.218.108	172.16.112.50	SMTP	73	195.115.218.108	172.16.112.50	25	63330	TCP
48	19:46:34	172.16.112.50	195.115.218.108	SMTP	60	172.16.112.50	195.115.218.108	63330	25	TCP
49	19:46:34	195.115.218.108	172.16.112.50	SMTP	78	195.115.218.108	172.16.112.50	25	63330	TCP
50	19:46:34	195.115.218.108	172.16.112.50	TCP	60	195.115.218.108	172.16.112.50	25	63330	TCP
51	19:46:34	172.16.112.50	195.115.218.108	TCP	60	172.16.112.50	195.115.218.108	63330	25	TCP
52	19:46:34	172.16.112.50	195.115.218.108	TCP	60	172.16.112.50	195.115.218.108	63330	25	TCP
53	19:46:34	195.115.218.108	172.16.112.50	TCP	60	195.115.218.108	172.16.112.50	25	63330	TCP
54	19:46:35	HP_dd:f9:a0	HP_dd:f9:a0	LLC	54	195.115.218.108	172.16.112.50	25	63330	TCP
55	19:46:37	HP_dd:f9:a0	HP_dd:f9:a0	LLC	54	195.115.218.108	172.16.112.50	25	63330	TCP
56	19:46:37	194.27.251.21	172.16.112.207	TELNET	60	194.27.251.21	172.16.112.207	2072	23	TCP
57	19:46:37	172.16.112.207	194.27.251.21	TELNET	60	172.16.112.207	194.27.251.21	23	2072	TCP
58	19:46:37	194.27.251.21	172.16.112.207	TCP	60	194.27.251.21	172.16.112.207	2072	23	TCP
59	19:46:37	194.27.251.21	172.16.112.207	TELNET	60	194.27.251.21	172.16.112.207	2072	23	TCP
60	19:46:37	172.16.112.207	194.27.251.21	TELNET	60	172.16.112.207	194.27.251.21	23	2072	TCP
61	19:46:37	194.27.251.21	172.16.112.207	TCP	60	194.27.251.21	172.16.112.207	2072	23	TCP
62	19:46:37	194.27.251.21	172.16.112.207	TELNET	60	194.27.251.21	172.16.112.207	2072	23	TCP
63	19:46:37	172.16.112.207	194.27.251.21	TELNET	60	172.16.112.207	194.27.251.21	23	2072	TCP
64	19:46:37	194.27.251.21	172.16.112.207	TCP	60	194.27.251.21	172.16.112.207	2072	23	TCP
65	19:46:37	194.27.251.21	172.16.112.207	TELNET	60	194.27.251.21	172.16.112.207	2072	23	TCP
66	19:46:37	172.16.112.207	194.27.251.21	TELNET	60	172.16.112.207	194.27.251.21	23	2072	TCP
67	19:46:37	194.27.251.21	172.16.112.207	TCP	60	194.27.251.21	172.16.112.207	2072	23	TCP
68	19:46:37	194.27.251.21	172.16.112.207	TELNET	60	194.27.251.21	172.16.112.207	2072	23	TCP
69	19:46:37	172.16.112.207	194.27.251.21	TELNET	60	172.16.112.207	194.27.251.21	23	2072	TCP
70	19:46:37	194.27.251.21	172.16.112.207	TCP	60	194.27.251.21	172.16.112.207	2072	23	TCP
71	19:46:38	194.27.251.21	172.16.112.207	TELNET	60	194.27.251.21	172.16.112.207	2072	23	TCP
72	19:46:38	172.16.112.207	194.27.251.21	TELNET	60	172.16.112.207	194.27.251.21	23	2072	TCP
73	19:46:38	194.27.251.21	172.16.112.207	TCP	60	194.27.251.21	172.16.112.207	2072	23	TCP
74	19:46:38	172.16.113.204	194.7.248.153	TCP	74	172.16.113.204	194.7.248.153	63331	25	TCP
75	19:46:38	194.7.248.153	172.16.113.204	TCP	60	194.7.248.153	172.16.113.204	25	63331	TCP
76	19:46:38	172.16.113.204	194.7.248.153	TCP	60	172.16.113.204	194.7.248.153	63331	25	TCP
77	19:46:38	194.7.248.153	172.16.113.204	SMTP	137	194.7.248.153	172.16.113.204	25	63331	TCP
78	19:46:38	172.16.113.204	194.7.248.153	TCP	60	172.16.113.204	194.7.248.153	63331	25	TCP
79	19:46:38	194.27.251.21	172.16.112.207	TELNET	60	194.27.251.21	172.16.112.207	2072	23	TCP
80	19:46:38	172.16.112.207	194.27.251.21	TELNET	60	172.16.112.207	194.27.251.21	23	2072	TCP
81	19:46:38	194.27.251.21	172.16.112.207	TCP	60	194.27.251.21	172.16.112.207	2072	23	TCP
82	19:46:38	172.16.113.204	194.7.248.153	SMTP	79	172.16.113.204	194.7.248.153	63331	25	TCP
83	19:46:38	194.7.248.153	172.16.113.204	SMTP	80	194.7.248.153	172.16.113.204	25	63331	TCP
84	19:46:38	172.16.113.204	194.7.248.153	SMTP	79	172.16.113.204	194.7.248.153	63331	25	TCP
85	19:46:38	194.7.248.153	172.16.113.204	SMTP	101	194.7.248.153	172.16.113.204	25	63331	TCP

86	19:46:38	172.16.113.204	194.7.248.153	SMTP	94	172.16.113.204	194.7.248.153	63331	25	TCP
87	19:46:38	194.7.248.153	172.16.113.204	SMTP	101	194.7.248.153	172.16.113.204	25	63331	TCP
88	19:46:38	172.16.113.204	194.7.248.153	SMTP	90	172.16.113.204	194.7.248.153	63331	25	TCP
89	19:46:38	194.7.248.153	172.16.113.204	SMTP	89	194.7.248.153	172.16.113.204	25	63331	TCP
90	19:46:38	172.16.113.204	194.7.248.153	SMTP	60	172.16.113.204	194.7.248.153	63331	25	TCP
91	19:46:38	194.7.248.153	172.16.113.204	SMTP	104	194.7.248.153	172.16.113.204	25	63331	TCP
92	19:46:38	172.16.113.204	194.7.248.153	IMF	1028	172.16.113.204	194.7.248.153	63331	25	TCP
93	19:46:38	194.7.248.153	172.16.113.204	SMTP	73	194.7.248.153	172.16.113.204	25	63331	TCP
94	19:46:38	172.16.113.204	194.7.248.153	SMTP	60	172.16.113.204	194.7.248.153	63331	25	TCP
95	19:46:38	194.7.248.153	172.16.113.204	SMTP	78	194.7.248.153	172.16.113.204	25	63331	TCP
96	19:46:38	194.7.248.153	172.16.113.204	TCP	60	194.7.248.153	172.16.113.204	25	63331	TCP
97	19:46:38	172.16.113.204	194.7.248.153	TCP	60	172.16.113.204	194.7.248.153	63331	25	TCP
98	19:46:38	172.16.113.204	194.7.248.153	TCP	60	172.16.113.204	194.7.248.153	63331	25	TCP
99	19:46:38	194.7.248.153	172.16.113.204	TCP	60	194.7.248.153	172.16.113.204	25	63331	TCP
100	19:46:38	194.27.251.21	172.16.112.207	TELNET	60	194.27.251.21	172.16.112.207	2072	23	TCP
101	19:46:38	172.16.112.207	194.27.251.21	TELNET	60	172.16.112.207	194.27.251.21	23	2072	TCP
102	19:46:38	194.27.251.21	172.16.112.207	TCP	60	194.27.251.21	172.16.112.207	2072	23	TCP
103	19:46:38	172.16.112.207	194.27.251.21	TELNET	63	172.16.112.207	194.27.251.21	23	2072	TCP
104	19:46:38	194.27.251.21	172.16.112.207	TCP	60	194.27.251.21	172.16.112.207	2072	23	TCP
105	19:46:39	Cisco_38:46:3b	Cisco_38:46:3b	LOOP	60	194.27.251.21	172.16.112.207	2072	23	TCP
106	19:46:40	HP_dd:f9:a0	HP_dd:f9:a0	LLC	54	194.27.251.21	172.16.112.207	2072	23	TCP
107	19:46:41	HP_dd:f9:a0	HP_dd:f9:a0	LLC	54	194.27.251.21	172.16.112.207	2072	23	TCP
108	19:46:41	194.7.248.153	172.16.112.194	TELNET	60	194.7.248.153	172.16.112.194	1489	23	TCP
109	19:46:41	172.16.112.194	194.7.248.153	TELNET	60	172.16.112.194	194.7.248.153	23	1489	TCP
110	19:46:41	194.7.248.153	172.16.112.194	TCP	60	194.7.248.153	172.16.112.194	1489	23	TCP
111	19:46:42	194.7.248.153	172.16.112.194	TELNET	60	194.7.248.153	172.16.112.194	1489	23	TCP
112	19:46:42	172.16.112.194	194.7.248.153	TELNET	60	172.16.112.194	194.7.248.153	23	1489	TCP
113	19:46:42	194.7.248.153	172.16.112.194	TCP	60	194.7.248.153	172.16.112.194	1489	23	TCP
114	19:46:42	194.7.248.153	172.16.112.194	TELNET	60	194.7.248.153	172.16.112.194	1489	23	TCP
115	19:46:42	172.16.112.194	194.7.248.153	TELNET	60	172.16.112.194	194.7.248.153	23	1489	TCP
116	19:46:42	194.7.248.153	172.16.112.194	TCP	60	194.7.248.153	172.16.112.194	1489	23	TCP
117	19:46:42	194.7.248.153	172.16.112.194	TELNET	60	194.7.248.153	172.16.112.194	1489	23	TCP
118	19:46:42	172.16.112.194	194.7.248.153	TELNET	60	172.16.112.194	194.7.248.153	23	1489	TCP
119	19:46:42	194.7.248.153	172.16.112.194	TCP	60	194.7.248.153	172.16.112.194	1489	23	TCP
120	19:46:42	194.7.248.153	172.16.112.194	TELNET	60	194.7.248.153	172.16.112.194	1489	23	TCP
121	19:46:42	172.16.112.194	194.7.248.153	TELNET	60	172.16.112.194	194.7.248.153	23	1489	TCP
122	19:46:42	194.7.248.153	172.16.112.194	TCP	60	194.7.248.153	172.16.112.194	1489	23	TCP
123	19:46:42	194.7.248.153	172.16.112.194	TELNET	60	194.7.248.153	172.16.112.194	1489	23	TCP
124	19:46:42	172.16.112.194	194.7.248.153	TELNET	60	172.16.112.194	194.7.248.153	23	1489	TCP
125	19:46:42	194.7.248.153	172.16.112.194	TCP	60	194.7.248.153	172.16.112.194	1489	23	TCP
126	19:46:42	WesternD_17:79:5	Cisco_38:46:3b	ARP	60	194.7.248.153	172.16.112.194	1489	23	TCP
127	19:46:42	Cisco_38:46:3b	WesternD_17:79:5	ARP	60	194.7.248.153	172.16.112.194	1489	23	TCP
128	19:46:42	194.7.248.153	172.16.112.194	TELNET	60	194.7.248.153	172.16.112.194	1489	23	TCP
129	19:46:42	172.16.112.194	194.7.248.153	TELNET	60	172.16.112.194	194.7.248.153	23	1489	TCP

130	19:46:42	194.7.248.153	172.16.112.194	TCP	60	194.7.248.153	172.16.112.194	1489	23	TCP
131	19:46:42	194.7.248.153	172.16.112.194	TELNET	60	194.7.248.153	172.16.112.194	1489	23	TCP
132	19:46:42	172.16.112.194	194.7.248.153	TELNET	60	172.16.112.194	194.7.248.153	23	1489	TCP
133	19:46:42	194.7.248.153	172.16.112.194	TCP	60	194.7.248.153	172.16.112.194	1489	23	TCP
134	19:46:42	194.7.248.153	172.16.112.194	TELNET	60	194.7.248.153	172.16.112.194	1489	23	TCP
135	19:46:42	172.16.112.194	194.7.248.153	TELNET	60	172.16.112.194	194.7.248.153	23	1489	TCP
136	19:46:42	194.7.248.153	172.16.112.194	TCP	60	194.7.248.153	172.16.112.194	1489	23	TCP
137	19:46:43	194.7.248.153	172.16.112.194	TELNET	60	194.7.248.153	172.16.112.194	1489	23	TCP
138	19:46:43	172.16.112.194	194.7.248.153	TELNET	60	172.16.112.194	194.7.248.153	23	1489	TCP
139	19:46:43	194.7.248.153	172.16.112.194	TCP	60	194.7.248.153	172.16.112.194	1489	23	TCP
140	19:46:43	194.7.248.153	172.16.112.194	TELNET	60	194.7.248.153	172.16.112.194	1489	23	TCP
141	19:46:43	172.16.112.194	194.7.248.153	TELNET	60	172.16.112.194	194.7.248.153	23	1489	TCP
142	19:46:43	194.7.248.153	172.16.112.194	TCP	60	194.7.248.153	172.16.112.194	1489	23	TCP
143	19:46:43	194.27.251.21	172.16.112.207	TELNET	60	194.27.251.21	172.16.112.207	2072	23	TCP
144	19:46:43	194.7.248.153	172.16.112.194	TELNET	60	194.7.248.153	172.16.112.194	1489	23	TCP
145	19:46:43	172.16.112.194	194.7.248.153	TELNET	60	172.16.112.194	194.7.248.153	23	1489	TCP
146	19:46:43	172.16.112.207	194.27.251.21	TCP	60	172.16.112.207	194.27.251.21	23	2072	TCP
147	19:46:43	194.7.248.153	172.16.112.194	TCP	60	194.7.248.153	172.16.112.194	1489	23	TCP
148	19:46:43	194.7.248.153	172.16.112.194	TELNET	60	194.7.248.153	172.16.112.194	1489	23	TCP
149	19:46:43	172.16.112.194	194.7.248.153	TELNET	60	172.16.112.194	194.7.248.153	23	1489	TCP
150	19:46:43	194.7.248.153	172.16.112.194	TCP	60	194.7.248.153	172.16.112.194	1489	23	TCP

APPENDIX B: APPROVAL EMAIL TO USE CAIDA 2007 DDoS DATASET

Fwd: [ddos-data-access] DDoS Data Request: Adenrele Afolorunso (National Open University of Nigeria)

Inbox x

Adenrele AFOLORUNSO <aafolorunsho@noun.edu.ng> 12/19/16
to me

---------- Forwarded message ----------
From: **Paul Hick** <pphick@caida.org>
Date: Wed, Dec 14, 2016 at 10:20 PM
Subject: Re: [ddos-data-access] DDoS Data Request: Adenrele Afolorunso (National Open University of Nigeria)
To: aafolorunsho@noun.edu.ng
Cc: ddos-data-access@caida.org

Hello Adenrele Afolorunso,

Thank you for requesting CAIDA's 2007 DDoS Attack Dataset.

Your request has been approved

A username and temporary password has been assigned to you.

Your username is your email-address: aafolorunsho@noun.edu.ng
Your temporary password is : ujeiquae

The procedure for accessing the CAIDA data is as follows:

1.) You can use the temporary password to log in to
https://data.caida.org/cgi-bin/chpw
You will be prompted to change your password to a permanent one at that time.
2.) You can use your username and new, permanent password to download datasets from

https://data.caida.org/datasets/security/ddos-20070804/

We've had several people experience difficulties retrieving the data using web browsers. Our guess is that the browsers themselves have difficulty handling the large trace files. We recommend using a recent version (> 1.10.2) of the program 'wget' (http://www.gnu.org/software/wget/) for downloading data.

For more information on usage of CAIDA data see the CAIDA data usage FAQ:
http://www.caida.org/data/data-usage-faq.xml

We rely on you to comply with the Acceptable Use Policies for this dataset, and report all publications (papers, presentations, class projects, websites etc.) to us. We'll use your input to update:
http://www.caida.org/data/publications/bydataset/index.xml
unless you ask that your publication not be included on our website.

We urge researchers to consider the data carefully and be sure that their use in research is consistent with the nature and the limitations of the data. Please contact us at data-info@caida.org if you have questions about the data.

Best Regards,
Paul Hick

On Tue, 2016-12-13 at 15:00 -0800, aafolorunsho@noun.edu.ng wrote:
> **********
> This message was generated by the CAIDA formhandler in response to a
> submission, detailed below.
> **********
>
> A DDoS Data Request form was submitted by
> aafolorunsho@noun.edu.ng on Tuesday, December 13, 2016 at 15:00:14
>
> Here is a summary of the submission. Event logged as follows:
> ---
> [1401] 15:00:14 12/13/2016 :: aafolorunsho@noun.edu.ng submitted
> from Adenrele Afolorunso (aafolorunsho@noun.edu.ng) from National Open University of Nigeria
> ---
> DETAILS:
> Status: academic researcher/faculty advisor/student,
> First Name: Adenrele
> Last Name: Afolorunso
> Institution: National Open University of Nigeria
> Address: Computer Science Department,
> Faculty of Science,
> National Open University of Nigeria
> 91, Cadastral zone, Nnamdi Azikwe Expressway,
> Jabi, Abuja.
> Nigeria
> Phone Number: +2348033288154
> Email: aafolorunsho@noun.edu.ng
> Position: student,
> Advisor's Name (if student): Prof. Olayide Abass
> Project URL:
> Usage: To develop a novel parsimonious and computationally efficient model for predicting DDoS attacks in network systems and introduce a global convergent method for training the HMM in order to reduce the complexity.
>
> Datasets: ddos-attack-2007
> PhD students:
> MS students:
> Other students:
> AUP: A. A.
> Temporary password: (removed)
> Already have password:
> Subscribe to data-announce: yes
> ---
>
> Errors that occured during submission:
> + No errors detected.
>
>_____
> ddos-data-access mailing list
> ddos-data-access@caida.org
> https://mailman.caida.org/mailman/listinfo/ddos-data-access
>
--
Paul Hick

Center for Applied Internet Data Analysis
San Diego Supercomputer Center
University of California San Diego
9500 Gilman Dr., La Jolla, CA 92093-0505

Phone: (858) 822-3674
URL : http://www.caida.org/~pphick
Email: pphick@caida.org
Skype: pphick

APPENDIX C: PROGRAM LISTING

```
clc
clear

% Read the raw data
[~,~,D]=xlsread('ThesisData.xlsx');

%  Find the probability of the source IP, Destination IP and protocol type,
%  source port, destination port and length of protocol

D2=D(2:end,5:12);

%Remove NAN
%[c1,c2] = isnan(D2);

%separate the groups

Pr= D2(:,1); %Protocol
L =  D2(:,2); %Packet length
Sip = D2(:,3);  %Source ip
Dip = D2(:,4);  %Destination ip
Ptype= D2(:,7);
Sport=D2(:,5);
Dsport=D2(:,6);

% Source ip %%%%%%%%%%%%%%%%%%%%
%identify unique parameters
n1=length(Sip);
Sip2 = unique(Sip);

for i=1:length(Sip2)
   x(i)=sum(cell2mat(strfind(Sip,Sip2{i})));

end

%Probability of Source ip

P1= x./n1;
Sip3= {Sip2;x;P1};

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Destination ip %%%%%%%%%%%%%%%%%%%%
%identify unique parameters

n2=length(Dip);
Dip2 = unique(Dip);

for i=1:length(Dip2)
   x2(i)=sum(cell2mat(strfind(Dip,Dip2{i})));

end
```

```
%Probability of Destination ip

P2= x2./n2;
Dip3= {Dip2;x2;P2};
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Packet type %%%%%%%%%%%%%%%%%%%%
%identify unique parameters
n3=length(Ptype);
Ptype2 = unique(Ptype);

for i=1:length(Ptype2)
   x3(i)=sum(cell2mat(strfind(Ptype,Ptype2{i})));
end

%Probability of Packet type

P3= x3./n3;
Ptype3= {Ptype2;x3;P3};
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Packet Length
%identify unique parameters
n4=length(L);
%L2 = unique(L);

% for i=1:length(L)
%    x4(i)=sum(cell2mat(strfind(L,L2{i})));
%
 x4 =sum(cell2mat(L));
% end

%Probability of Packet length

P4= x4./n4;
L3= {x4;P4};


%Protocol
%identify unique parameters
n5=length(Pr);
Pr2 = unique(Pr);

for i=1:length(Pr2)
   x5(i)=sum(cell2mat(strfind(Pr,Pr2{i})));

end

%Probability of Packet length

P5= x5./n5;
Pr3= {Pr2;x5;P5};
```

```
disp(Ptype3)
disp(Dip3)
disp(Sip3)
disp(L3)
disp(Pr3)

xlswrite('Results.xlsx',{'Type','Entropy','Probability'},'PackType','A1')
xlswrite('Results.xlsx',Ptype3{1},'PackType','A2')
xlswrite('Results.xlsx',[x3' P3'],'PackType','B2')

xlswrite('Results.xlsx',{'Type','Entropy','Probability'},'DestinationIP','A1')
xlswrite('Results.xlsx',Dip3{1},'DestinationIP','A2')
xlswrite('Results.xlsx',[x2' P2'],'DestinationIP','B2')

xlswrite('Results.xlsx',{'Type','Entropy','Probability'},'SourceIP','A1')
xlswrite('Results.xlsx',Sip3{1},'SourceIP','A2')
xlswrite('Results.xlsx',[x' P1'],'SourceIP','B2')

xlswrite('Results.xlsx',{'Type','Entropy','Probability'},'Length','A1')
xlswrite('Results.xlsx',L3{1},'Length','A2')
xlswrite('Results.xlsx',[x4' P4'],'Length','B2')

xlswrite('Results.xlsx',{'Type','Entropy','Probability'},'Potocol','A1')
xlswrite('Results.xlsx',Pr3{1},'Potocol','A2')
xlswrite('Results.xlsx',[x5' P5'],'Potocol','B2')


%Different datasets are scored in this Section
%A classifier is then used to train based on prescribed labels.

load VBHMM.mat
load Clustered.mat
load ROC_Values.mat
Feat = 6;
nFeat = 3;

%Score each sequence that translates to a minute.


for v = 1:nFeat
startPos = 1;
lp = 1;
element = 1;
   while lp >= 1
      [startPos, tDats,lp] = sample(nS_Index(v,:),startPos,6);
      nTdats = db2cell_String(transpose(tDats));
      sF_Score(v,element) = vbhmm_cF(nTdats,Cluster_Set,model);
      element = element + 1;
   end
end
VBscored = (sum(sF_Score))/3;

%Score Normal data
for v1 = 1:nFeat
startPos1 = 1;
lp1 = 1;
element1 = 1;
```

```
   while lp1 >= 1
      [startPos1, tDats1,lp1] = sample(nTS_Index(v1,:),startPos1,6);
      nTdats1 = db2cell_String(transpose(tDats1));
      sF_Score1(v1,element1) = vbhmm_cF(nTdats1,Cluster_Set,model1);
      element1 = element1 + 1;
   end
end

nVBscored = (sum(sF_Score1))/3;

vScore = [VBscored nVBscored];
vScore = transpose(vScore);

%Compute Receiver Operating Characteristics for scores generated by HMM.
ROC_label = transpose(ROC_label);
[vX,vY,vT,vAUC] = perfcurve(ROC_label,vScore,'Threat');

%figure
vs = 0.31827053 * (0.0001:0.0001:0.6661);
save('VBROC.mat','vs', 'vY' ,'vX');


%Computation of P_APAN
%State determination and Transistion Probability computation.

load Clustered.mat
span = 1;
states = 6;
%Determination of Outlying states
for j = 1:numel(GridVector)
   [d, dm] = OF(GridVector(j), Centroid);
   if dm >= d
       ot{span} = j;
       span = span + 1;
   end
end
ot = cell2mat(ot);

%Compute initial Probability Distribution of training dataset
for j = 1:6
    take = S_Index(j,:);
    [~,~,new] = unique(take);
    Freq(j,:) = accumarray(new,1,[],@sum); %Determine Frequency for each
cluster.
    FreqN = sum(Freq);  %Compute overall frequency of each state.
end
for b = 1:6
    Q_apan{b} = FreqN(b)/sum(FreqN);
end
Q_apan = cell2mat(Q_apan);


%Compute State Transition Probabilities.

for w=1:states
    for q=1:states
        TransP(w,q) = count(w,q)/FreqN(w);
```

```
    end
end

%Finally Determination of APAN

%P_Apan = [TransP, 1e-14; Q_apan, 1e-23];
save('P_Apan.mat', 'Q_apan', 'TransP');


classdef (ConstructOnLoad) VBIHMM %#ok<*MCSUP,*PROP>
%VBIHMM    Construct Variational Bayesian hidden Markov model
%
%   VBIHMM constructs a Variational Bayesian hidden Markov model with a given
%   number of hidden states, symbols and real-valued features.
%
%   Usage:   Obj=VBIHMM(M,K,D)
%
%   Inputs:  M   - Number of states
%            K   - Number of symbols
%            D   - Number of features
%
%   Outputs: Obj - Variational Bayesian hidden Markov model
%
%   Input arguments M, K and D should be positive integer scalars.
%
%
%   Copyright (c) 2016 Afolorunso A. A.

    % Number of states, symbols and features.
    properties (SetAccess=private)
        NumState
        NumSym
        NumFeat
    end

    % Hyper-parameters.
    properties
        TransWeight
        TransStren
        EmissWeight
        EmissStren
        CompLoc
        CompScale
        CompDisp
        CompPrec
    end

    % Constructor.
    methods
        function Obj=VBIHMM(NumState,NumSym,NumFeat)
            if nargin()>0

                % Check number of arguments.
                if nargin()<3
                    error('VBIHMM:NotEnoughInputs', ...
                        'Not enough inputs.')
                end
```

```
            if nargin()>3
                error('VBIHMM:TooManyInputs', ...
                    'Too many inputs.')
            end
            if nargout()>1
                error('VBIHMM:TooManyOutputs', ...
                    'Too many outputs.')
            end

            % Check input arguments.
            CheckArg(NumState,NumSym,NumFeat)

            % Set number of states, symbols and features.
            Obj.NumState=NumState;
            Obj.NumSym=NumSym;
            Obj.NumFeat=NumFeat;

            % Set default hyper-parameters.
            Obj.TransWeight=ones(NumState,NumState)/NumState;
            Obj.TransStren=ones(NumState,1);
            Obj.EmissWeight=ones(NumSym,NumState)/NumSym;
            Obj.EmissStren=ones(NumState,1);
            Obj.CompLoc=zeros(NumFeat,NumSym);
            Obj.CompScale=ones(NumSym,1);
            Obj.CompDisp=RepVal(eye(NumFeat),3,NumSym);
            Obj.CompPrec=NumFeat*ones(NumSym,1);

        end
    end
end

% Access methods.
methods
    function Obj=set.TransWeight(Obj,TransWeight)

        % Store number of states.
        NumState=Obj.NumState;

        % Check transition weights.
        if ~isnumeric(TransWeight)
            error('VBIHMM:BadInputClass', ...
                'Input must be numeric.')
        end
        if ~isreal(TransWeight)
            error('VBIHMM:BadInputClass', ...
                'Input must be real.')
        end
        if isempty(TransWeight)
            error('VBIHMM:BadInputSize', ...
                'Input must be non-empty.')
        end
        if ndims(TransWeight)>2
            error('VBIHMM:BadInputSize', ...
                'Input must be a matrix.')
        end
        if size(TransWeight,1)~=NumState
            error('VBIHMM:BadInputSize', ...
```

```
            'Input must have %d row(s).',NumState)
        end
        if size(TransWeight,2)~=NumState
            error('VBIHMM:BadInputSize', ...
                'Input must have %d column(s).',NumState)
        end
        if any(isnan(TransWeight(:))|isinf(TransWeight(:)))
            error('VBIHMM:BadInputValue', ...
                'Input must contain finite numbers.')
        end
        for i=1:NumState
            if any(TransWeight(:,i)<=0|abs(sum(TransWeight(:,i))-1)>...
                    eps()*NumState)
                error('VBIHMM:BadInputValue', ...
                    ['Column %d of input must contain numbers ',...
                        'on the unit simplex.'],i)
            end
        end

        % Set transition weights.
        Obj.TransWeight=TransWeight;

    end
    function Obj=set.TransStren(Obj,TransStren)

        % Store number of states.
        NumState=Obj.NumState;

        % Check transition strengths.
        if ~isnumeric(TransStren)
            error('VBIHMM:BadInputClass', ...
                'Input must be numeric.')
        end
        if ~isreal(TransStren)
            error('VBIHMM:BadInputClass', ...
                'Input must be real.')
        end
        if isempty(TransStren)
            error('VBIHMM:BadInputSize', ...
                'Input must be non-empty.')
        end
        if ndims(TransStren)>2||min(size(TransStren))>1
            error('VBIHMM:BadInputSize', ...
                'Input must be a vector.')
        end
        if numel(TransStren)~=NumState
            error('VBIHMM:BadInputSize', ...
                'Input must have %d element(s).',NumState)
        end
        if any(isnan(TransStren)|isinf(TransStren)|TransStren<=0)
            error('VBIHMM:BadInputValue', ...
                'Input must contain positive finite numbers.')
        end

        % Set transition strengths.
        Obj.TransStren=TransStren;

    end
```

```
function Obj=set.EmissWeight(Obj,EmissWeight)

    % Store number of states and symbols.
    NumState=Obj.NumState;
    NumSym=Obj.NumSym;

    % Check emission weights.
    if ~isnumeric(EmissWeight)
        error('VBIHMM:BadInputClass', ...
            'Input must be numeric.')
    end
    if ~isreal(EmissWeight)
        error('VBIHMM:BadInputClass', ...
            'Input must be real.')
    end
    if isempty(EmissWeight)
        error('VBIHMM:BadInputSize', ...
            'Input must be non-empty.')
    end
    if ndims(EmissWeight)>2
        error('VBIHMM:BadInputSize', ...
            'Input must be a matrix.')
    end
    if size(EmissWeight,1)~=NumSym
        error('VBIHMM:BadInputSize', ...
            'Input must have %d row(s).',NumSym)
    end
    if size(EmissWeight,2)~=NumState
        error('VBIHMM:BadInputSize', ...
            'Input must have %d column(s).',NumState)
    end
    if any(isnan(EmissWeight(:))|isinf(EmissWeight(:)))
        error('VBIHMM:BadInputValue', ...
            'Input must contain finite numbers.')
    end
    for i=1:NumState
        if any(EmissWeight(:,i)<=0|abs(sum(EmissWeight(:,i))-1)>...
                eps()*NumSym)
            error('VBIHMM:BadInputValue', ...
                ['Column %d of input must contain numbers ',...
                    'on the unit simplex.'],i)
        end
    end

    % Set emission weights.
    Obj.EmissWeight=EmissWeight;

end
function Obj=set.EmissStren(Obj,EmissStren)

    % Store number of states.
    NumState=Obj.NumState;

    % Check emission strengths.
    if ~isnumeric(EmissStren)
        error('VBIHMM:BadInputClass', ...
            'Input must be numeric.')
    end
```

```
        if ~isreal(EmissStren)
            error('VBIHMM:BadInputClass', ...
                'Input must be real.')
        end
        if isempty(EmissStren)
            error('VBIHMM:BadInputSize', ...
                'Input must be non-empty.')
        end
        if ndims(EmissStren)>2||min(size(EmissStren))>1
            error('VBIHMM:BadInputSize', ...
                'Input must be a vector.')
        end
        if numel(EmissStren)~=NumState
            error('VBIHMM:BadInputSize', ...
                'Input must have %d element(s).',NumState)
        end
        if any(isnan(EmissStren)|isinf(EmissStren)|EmissStren<=0)
            error('VBIHMM:BadInputValue', ...
                'Input must contain positive finite numbers.')
        end

        % Set emission strengths.
        Obj.EmissStren=EmissStren;

    end
    function Obj=set.CompLoc(Obj,CompLoc)

        % Store number of symbols and features.
        NumSym=Obj.NumSym;
        NumFeat=Obj.NumFeat;

        % Check component locations.
        if ~isnumeric(CompLoc)
            error('VBIHMM:BadInputClass', ...
                'Input must be numeric.')
        end
        if ~isreal(CompLoc)
            error('VBIHMM:BadInputClass', ...
                'Input must be real.')
        end
        if isempty(CompLoc)
            error('VBIHMM:BadInputSize', ...
                'Input must be non-empty.')
        end
        if ndims(CompLoc)>2
            error('VBIHMM:BadInputSize', ...
                'Input must be a matrix.')
        end
        if size(CompLoc,1)~=NumFeat
            error('VBIHMM:BadInputSize', ...
                'Input must have %d row(s).',NumFeat)
        end
        if size(CompLoc,2)~=NumSym
            error('VBIHMM:BadInputSize', ...
                'Input must have %d column(s).',NumSym)
        end
        if any(isnan(CompLoc(:))|isinf(CompLoc(:)))
            error('VBIHMM:BadInputValue', ...
```

```
            'Input must contain finite numbers.')
    end

    % Set component locations.
    Obj.CompLoc=CompLoc;

end
function Obj=set.CompScale(Obj,CompScale)

    % Store number of symbols.
    NumSym=Obj.NumSym;

    % Check component scales.
    if ~isnumeric(CompScale)
        error('VBIHMM:BadInputClass', ...
            'Input must be numeric.')
    end
    if ~isreal(CompScale)
        error('VBIHMM:BadInputClass', ...
            'Input must be real.')
    end
    if isempty(CompScale)
        error('VBIHMM:BadInputSize', ...
            'Input must be non-empty.')
    end
    if ndims(CompScale)>2||min(size(CompScale))>1
        error('VBIHMM:BadInputSize', ...
            'Input must be a vector.')
    end
    if numel(CompScale)~=NumSym
        error('VBIHMM:BadInputSize', ...
            'Input must have %d element(s).',NumSym)
    end
    if any(isnan(CompScale)|isinf(CompScale)|CompScale<=0)
        error('VBIHMM:BadInputValue', ...
            'Input must contain positive finite numbers.')
    end

    % Set component scales.
    Obj.CompScale=CompScale;

end
function Obj=set.CompDisp(Obj,CompDisp)

    % Store number of symbols and features.
    NumSym=Obj.NumSym;
    NumFeat=Obj.NumFeat;

    % Check component dispersions.
    if ~isnumeric(CompDisp)
        error('VBIHMM:BadInputClass', ...
            'Input must be numeric.')
    end
    if ~isreal(CompDisp)
        error('VBIHMM:BadInputClass', ...
            'Input must be real.')
    end
```

```
        if isempty(CompDisp)
            error('VBIHMM:BadInputSize', ...
                'Input must be non-empty.')
        end
        if ndims(CompDisp)>3
            error('VBIHMM:BadInputSize', ...
                'Input must be a cube.')
        end
        if size(CompDisp,1)~=NumFeat
            error('VBIHMM:BadInputSize', ...
                'Input must have %d row(s).',NumFeat)
        end
        if size(CompDisp,2)~=NumFeat
            error('VBIHMM:BadInputSize', ...
                'Input must have %d column(s).',NumFeat)
        end
        if size(CompDisp,3)~=NumSym
            error('VBIHMM:BadInputSize', ...
                'Input must have %d page(s).',NumSym)
        end
        if any(isnan(CompDisp(:))|isinf(CompDisp(:)))
            error('VBIHMM:BadInputValue', ...
                'Input must contain finite numbers.')
        end
        for i=1:NumSym
            Asymm=abs(CompDisp(:,:,i)-CompDisp(:,:,i)');
            if any(Asymm(:)>eps()*NumFeat)
                error('VBIHMM:BadInputValue', ...
                    'Input must contain symmetric matrices.')
            end
            [~,NumSing]=chol(CompDisp(:,:,i));
            if NumSing>0
                error('VBIHMM:BadInputValue', ...
                    'Input must contain positive-definite matrices.')
            end
        end

        % Set component dispersions.
        Obj.CompDisp=CompDisp;

    end
    function Obj=set.CompPrec(Obj,CompPrec)

        % Store number of symbols and features.
        NumSym=Obj.NumSym;
        NumFeat=Obj.NumFeat;

        % Check component precisions.
        if ~isnumeric(CompPrec)
            error('VBIHMM:BadInputClass', ...
                'Input must be numeric.')
        end
        if ~isreal(CompPrec)
            error('VBIHMM:BadInputClass', ...
                'Input must be real.')
        end
        if isempty(CompPrec)
            error('VBIHMM:BadInputSize', ...
```

```
                'Input must be non-empty.')
            end
            if ndims(CompPrec)>2||min(size(CompPrec))>1
                error('VBIHMM:BadInputSize', ...
                    'Input must be a vector.')
            end
            if numel(CompPrec)~=NumSym
                error('VBIHMM:BadInputSize', ...
                    'Input must have %d element(s).',NumSym)
            end
            if any(isnan(CompPrec)|isinf(CompPrec))
                error('VBIHMM:BadInputValue', ...
                    'Input must contain finite numbers.')
            end
            if any(CompPrec<=NumFeat-1)
                error('VBIHMM:BadInputValue', ...
                    'Input must contain numbers greater than %d.',NumFeat-1)
            end

            % Set component precisions.
            Obj.CompPrec=CompPrec;

        end
    end

    % Main methods.
    methods
        [State,Sym,Weight,Feat]=Sim(Obj,NumPoint,varargin)
        [Obj,Bound,varargout]=Estim(Obj,Feat,varargin)
    end

end

 hg = transpose(normParameter);
fr = transpose(Kullb_0Param);

figure

plot(1:length(hg),hg(1,:),'b',1:length(fr),fr(1,:),'-.r')
title ('RATE OF CONVERGENCE FOR LOGLIKEHOOD')
legend('KLD-HMM Model','HMM Model')
figure
plot(1:length(hg),hg(2,:),'b',1:length(fr),fr(2,:),'-.r')
title ('RATE OF CONVERGENCE FOR TRANSITION PARAMETER')
legend('KLD-HMM Model','HMM Model')
figure

plot(1:length(hg),hg(3,:),'b',1:length(fr),fr(3,:),'-.r')
title ('RATE OF CONVERGENCE FOR EMISSION PROBABILITY')
legend('KLD-HMM Model','HMM Model')
load PacketFreq.mat

figure;plot(1:length(nFreqN),nFreqN,1:length(FreqN),FreqN,'-.r')
title 'FREQUENCY DISTRIBUTION PER STATE'
xlabel 'Hidden States'
ylabel 'Occurrence Frequency'
legend('KLD HMM Model','HMM Model')
```

```
function CheckArg(varargin)

% Check input arguments.
for i=1:numel(varargin)
    if ~isnumeric(varargin{i})
        error('VBIHMM:BadInputClass', ...
            'Input %d must be numeric.',i)
    end
    if ~isreal(varargin{i})
        error('VBIHMM:BadInputClass', ...
            'Input %d must be real.',i)
    end
    if isempty(varargin{i})
        error('VBIHMM:BadInputSize', ...
            'Input %d must be non-empty.',i)
    end
    if ndims(varargin{i})>2||numel(varargin{i})>1
        error('VBIHMM:BadInputSize', ...
            'Input %d must be a scalar.',i)
    end
    if isnan(varargin{i})||isinf(varargin{i})
        error('VBIHMM:BadInputValue', ...
            'Input %d must contain a finite number.',i)
    end
    if round(varargin{i})~=varargin{i}||varargin{i}<=0
        error('VBIHMM:BadInputValue', ...
            'Input %d must contain a positive integer.',i)
    end
end

end


function TestVBIHMM()
%
%
clc,clear all,close all;
% Set number of states, symbols and features.
NumState=2;
NumSym=3;
NumFeat=5;

% Set number of sequences, points per sequence and missing values.
NumSeq=2;
NumPoint=100;
NumMiss=20;

% Set parameter generation options.
TransParam=1/5;
EmissParam=1/5;
LocParam=2;
DispParam=5;

% Set sampling options.
NumDeg=5;
```

```
NumObs=1000;

% Print header and display status.
fprintf('\n')
fprintf('Sampling data ... ')

% Generate parameters for sampling.
[Trans,Emiss,Loc,Disp]=GenParam(NumState,NumSym,NumFeat,...
    TransParam,EmissParam,LocParam,DispParam);
fprintf('\n')
disp(Emiss);
disp(Trans);
% Create model for sampling.
Obj=VBIHMM(NumState,NumSym,NumFeat);

% Set hyper-parameters.
Obj.TransWeight=Trans;
Obj.TransStren(:)=NumObs;
Obj.EmissWeight=Emiss;
Obj.EmissStren(:)=NumObs;
Obj.CompLoc=Loc;
Obj.CompScale(:)=NumObs;
Obj.CompDisp=Disp;
Obj.CompPrec(:)=max(NumObs,NumFeat);

% Sample data and remove values at random.
[~,~,~,Feat]=Obj.Sim(NumPoint(ones(NumSeq,1)),'NumDeg',NumDeg);
for i=1:NumSeq
    Feat{i}(randi(NumFeat*NumPoint,NumMiss,1))=nan();
end
disp(Feat);
% Update status.
fprintf('Done\n')
fprintf('Estimating model ... ')

% Create model for estimation.
Obj=VBIHMM(NumState,NumSym,NumFeat);

% Constrain transition parameters.
Obj.TransWeight=Trans;
Obj.TransStren(:)=NumObs;

% Estimate model and state probabilities.
[Obj,Bound,Prob]=Obj.Estim(Feat,'NumDeg',NumDeg);

% Update status.
fprintf('Done\n')
fprintf('Plotting results ... ')

% Close any existing figure.
close('all')

% Plot results.
PlotBound(Bound)
PlotSeg([Feat{:}],[Prob{:}])

% Update status and print footer.
```

```
fprintf('Done\n')
fprintf('\n')

end



function [Trans,Emiss,Loc,Disp]=GenParam(NumState,NumSym,NumFeat,...
    TransParam,EmissParam,LocParam,DispParam)

% Sample transition parameters.
Trans=rand(NumState,NumState);
Trans=bsxfun(@rdivide,Trans,sum(Trans,1));
Trans=(1-TransParam)*eye(NumState)+TransParam^2*Trans+...
    (1-TransParam)*TransParam/NumState;

% Sample emission parameters.
Emiss=rand(NumSym,NumState);
Emiss=bsxfun(@rdivide,Emiss,sum(Emiss,1));
Emiss=(1-EmissParam)*eye(NumSym,NumState)+EmissParam^2*Emiss+...
    (1-EmissParam)*EmissParam/NumSym;

% Sample location parameters.
Loc=LocParam*randn(NumFeat,NumSym);

% Allocate space.
Disp=zeros(NumFeat,NumFeat,NumSym);

% Sample dispersion parameters.
for i=1:NumSym
    Fact=randn(NumFeat,NumFeat+DispParam)/sqrt(NumFeat+DispParam);
    Disp(:,:,i)=Fact*Fact';
end

end



function PlotBound(Bound)

% Set options.
FontName='times';
FontSize=20;
LineColor=[0,0,1];
LineWidth=2;
MarkerSize=20;

% Create figure.
Fig=figure(...
    'NumberTitle','off',...
    'Name','Variational Lower Bound on the Model Evidence');

% Create axes.
Axes=axes(...
    'Parent',Fig,...
    'NextPlot','add',...
```

```
    'Box','on',...
    'Layer','top',...
    'FontName',FontName,...
    'FontSize',FontSize);

% Annotate axes.
set(get(Axes,'XLabel'),...
    'String','Iteration',...
    'FontName',FontName,...
    'FontSize',FontSize)
set(get(Axes,'YLabel'),...
    'String','Lower bound',...
    'FontName',FontName,...
    'FontSize',FontSize)
set(get(Axes,'Title'),...
    'String','Variational lower bound on the model evidence',...
    'FontName',FontName,...
    'FontSize',FontSize)

% Plot lower bound.
line(...
    'Parent',Axes,...
    'XData',1:numel(Bound),...
    'YData',Bound,...
    'Color',LineColor,...
    'LineWidth',LineWidth,...
    'Marker','.',...
    'MarkerSize',MarkerSize)

% Adjust axes.
set(Axes,...
    'XLim',[0,numel(Bound)+1])

end



function PlotSeg(Feat,Prob)

% Set options.
FontName='times';
FontSize=20;
ColorDilut=1/2;
LineWidth=2;
MarkerSize=20;
Margin=3/20;

% Store number of states, features and points.
[NumFeat,NumPoint]=size(Feat);
[NumState,~]=size(Prob);

% Build color map.
ColorMap=hsv(NumState);

% Create figure.
Fig=figure(...
    'NumberTitle','off',...
```

```
    'Name','Variational Bayesian Sequence Segmentation');

% Create invisible axes.
Axes=axes(...
    'Parent',Fig,...
    'Position',[Margin/2,Margin/2,1-Margin,1-Margin],...
    'Visible','off');

% Annotate axes.
set(get(Axes,'Title'),...
    'String','Variational Bayesian hidden Markov model',...
    'FontName',FontName,...
    'FontSize',FontSize,...
    'Visible','on')
set(get(Axes,'YLabel'),...
    'String','Feature values',...
    'FontName',FontName,...
    'FontSize',FontSize,...
    'Visible','on')

% Allocate space for axis handles.
Axes=zeros(NumFeat,1);

% Plot features.
for i=1:NumFeat

    % Set position.
    Pos(1)=Margin/2;
    Pos(2)=Margin/2+(1-Margin)*(1-i/NumFeat);
    Pos(3)=1-Margin;
    Pos(4)=1/NumFeat-Margin/NumFeat;

    % Create axes.
    Axes(i)=axes(...
        'Parent',Fig,...
        'Position',Pos,...
        'NextPlot','add',...
        'Box','on',...
        'Layer','top',...
        'FontName',FontName,...
        'FontSize',FontSize);

    % Plot lines. """
    line(...
        'Parent',Axes(i),...
        'XData',1:NumPoint,...
        'YData',Feat(i,:),...
        'Color',ColorDilut*get(Axes(i),'Color'),...
        'LineStyle',':',...
        'LineWidth',LineWidth)

    % Plot features.
    for j=1:NumPoint
        line(...
            'Parent',Axes(i),...
            'XData',j,...
            'YData',Feat(i,j),...
```

```
            'Color',(1-ColorDilut)*Prob(:,j)'*ColorMap+...
                ColorDilut*get(Axes(i),'Color'),...
            'LineStyle','none',...
            'Marker','.',...
            'MarkerSize',MarkerSize)
    end

    % Adjust axes.
    if i<NumFeat
        set(Axes(i),...
            'XTickLabel',{})
    end
    set(Axes(i),...
        'XLim',[0,NumPoint+1])

end

% Allocate space for line handles.
Line=zeros(NumState,1);

% Plot pure lines.
for i=1:NumState
    Line(i)=line(...
        'Parent',Axes(1),...
        'XData',[],...
        'YData',[],...
        'Color',ColorMap(i,:),...
        'Marker','.',...
        'MarkerSize',MarkerSize);
end

% Allocate space for labels.
Label=cell(NumState,1);

% Create labels.
for i=1:NumState
    Label{i}=sprintf('State %d: %2.1f%%',i,100*(sum(Prob(i,:))/NumPoint));
end

% Annotate plot.
set(legend(Line,Label{:}),...
    'FontName',FontName,...
    'FontSize',FontSize,...
    'Location','northeast')

% Adjust axis limits.
set(Axes,...
    'XLim',[0,NumPoint+1])

% Link axes.
linkaxes(Axes,'x')

end
```