Improvement of DDoS Attack Detection and Web Access Anonymity

by

Theerasak Thapngam
B.Eng (Prince of Songkla University)
M.I.T (Deakin University)

Submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

Deakin University

September, 2011
I am the author of the thesis entitled

**Improvement of DDoS Attack Detection and Web Access Anonymity**

submitted for the degree of

**Doctor of Philosophy**

This thesis may be made available for consultation, loan and limited copying in accordance with the Copyright Act 1968.

'I certify that I am the student named below and that the information provided in the form is correct'

Full Name  Theerasak Thapngam

Signed [Signature Redacted by Library]

Date  24/03/2012
DEAKIN UNIVERSITY

CANDIDATE DECLARATION

I certify that the thesis entitled

Improvement of DDoS Attack Detection and Web Access Anonymity

submitted for the degree of

Doctor of Philosophy

is the result of my own work and that where reference is made to the
work of others, due acknowledgment is given.

I also certify that any material in the thesis which has been accepted for a
degree or diploma by any university or institution is identified in the text.

'I certify that I am the student named below and that the information
provided in the form is correct'

Full Name  Theerasak Thapngam

Signed  [Signature Redacted by Library]

Date  24/03/2019
ABSTRACT

This thesis presents an efficient set of methods to improve security on service availability and information privacy in a computer network environment with a focus on two important issues: the detection of distributed denial of service (DDoS) attacks and the anonymity of web access. The proposed methods for DDoS attack detection allow service providers in cyber-space to strengthen and protect their valuable online service from numerous DDoS attacks. For web access anonymity, the proposed methods allow web service providers to achieve perfect anonymity of web access to protect the personally identifiable information of legitimate users. Unique methods with several sets of mathematical models are proposed in this thesis to enhance both DDoS attack detection and web access anonymity. A number of publicly available datasets were implemented in our experiments to evaluate the performance of these methods. The results show a significant improvement in terms of detection performance and level of anonymity.

DDoS attack detection is a key component in a DDoS defence system. Our detection methods allow for detection of DDoS attacks plus the ability to discriminate attacks from legitimate traffic flows in particular flash crowds (FCs). We measure the different levels of similarity, dependency and predictability among DDoS attacks and FCs. With nominated thresholds, our detection methods discriminate suspicious traffic flows and improve security of availability for online services.

Packet padding strategy is a key approach to achieving perfect anonymity of web access. Web access anonymity allows legitimate users to protect their privacy information and remain anonymous to a third party. In this thesis, we propose pre-fetched web objects as cover traffic instead of the commonly used dummy packet padding strategy in order to improve the level of anonymity. We formally establish several mathematical models for satisfying the requirements of this solution. As a result of our experiment, the proposed strategy can reduce a significant amount of delay and cost compared to the dummy packet padding strategy.

Finally, we summarise the methodology, the contribution of this thesis, and discuss future work. We point out the conflict between the deployment of traffic analysis for DDoS attack detection and the requirement of a high level of anonymity for web access. We believe this study provides an original solution to the compromises of these two contradictory topics and leads to the achievement of high security among service availability and increased information privacy.
ACKNOWLEDGEMENTS

I would sincerely like to thank my supervisor, Professor Wanlei Zhou for his tireless efforts in helping to review this thesis and for providing guidance throughout its preparation. I appreciate his ongoing support and patience. He generously gave me his time and encouragement to resolve the matter. I feel honoured that I had his supervision, guidance and leadership during my Ph.D. candidature.

I am also thankful to my associate supervisor, Dr. Shui Yu for guiding the direction of my research. He also helped to revise and review research papers and other material written for publication. I appreciated his assistance and support during my Ph.D. candidature.

I offer my sincere thanks and appreciation to Dr. Xiang Yang and Dr. Robin Doss, for our frequent discussions on the direction of research on DDoS attacks.

I would like to thank my Ph.D. and research colleagues. They are Dr. Rafiq Islam, Dr. Leanne Ngo, Dr. Ashley Chonka, Dr. Ferial Khaddage, Dr. Ke Li, Dr. Ping Li, Yini Wang, Alessio Bonti and my other colleagues.

I would also like to thank the many staff from the School of Information Technology at Deakin University, for their assistance and support during the preparation of this thesis and throughout my studies.

I also appreciate the generous financial support provided by Deakin University through Deakin University Postgraduate Research Scholarship.

Finally, to all my family and friends, I would like to thank my parents, Janya and Udomsin Thapngam, my sister, Wilailak Thapngam, my wife, Yupaporn Thapngam, my brother-in-law, Kampol, all my grandparents, uncles and aunts, as well as my many cousins and best friends around the world.

A special thank to my newborn son, Theetouch (Alpha) Thapngam for becoming an inspiration to my work during the time of my candidature.
PUBLICATIONS

During my Ph.D. candidature, the following research papers were published in fully refereed international conference proceedings and journals, or have been accepted for publication in, or submitted to Journals. Excerpts from some of these papers, which were the contribution of the author, have been reproduced in part or in full within the body of this thesis.


# TABLE OF CONTENTS

Abstract .................................................................................................................................. i  
Acknowledgement ........................................................................................................ ii  
Publications ................................................................................................................ iii  
Table of Contents ........................................................................................................ iv  
List of Tables ................................................................................................................ viii  
List of Figures ............................................................................................................... ix  
Acronyms ..................................................................................................................... xii  

Chapter 1 ........................................................................................................................ 1  
Introduction ............................................................................................................... 1  
1.1 Motivation and Rationale ...................................................................................... 1  
1.2 Thesis Overview .................................................................................................... 3  
1.3 Thesis Structure ................................................................................................... 5  

Chapter 2 ........................................................................................................................ 8  
Related Literature ...................................................................................................... 8  
2.1 DDoS Attack ......................................................................................................... 8  
2.2 DDoS Network Models ....................................................................................... 10  
2.2.1 Client-Server DoS Network Model ................................................................ 10  
2.2.2 Typical DDoS Network Model ...................................................................... 13  
2.2.3 Distributed Reflector DoS Network Model ................................................. 16  
2.2.4 Evolution of the DDoS Network Architecture ........................................... 17  
2.3 DDoS Attack Sources ....................................................................................... 18  
2.3.1 DDoS Attack Tools ....................................................................................... 19  
2.3.2 DDoS Attack Worms ................................................................................... 21  
2.3.3 DDoS Attack Botnets .................................................................................. 22  
2.3.4 Evolution of DDoS Attack Sources ............................................................. 24  
2.4 DDoS Reactive Defence Mechanisms ................................................................. 27  
2.5 DDoS Detection Strategies .............................................................................. 29  
2.5.1 Signature-based Detection .......................................................................... 30  
2.5.2 Anomaly Detection ...................................................................................... 31  
2.5.3 Hybrid Detection System .......................................................................... 32  
2.5.4 Third-Party Detection ................................................................................. 33  
2.6 Traffic Analysis Attacks .................................................................................... 33  
2.7 Research Challenges ......................................................................................... 35  
2.8 Chapter Summary and Conclusion ..................................................................... 38  

Chapter 3 ................................................................................................................................ 39  
Discrimination of DDoS Attack and Flash Crowd ...................................................... 39  
3.1 Introduction .......................................................................................................... 39
8.1 Methodology and Thesis Overview......................................................154
8.2 Thesis Contribution........................................................................155
8.3 Future Directions............................................................................160

References..........................................................................................165
LIST OF TABLES

Table 1 A sample of victim responses to typical attacks..............................................11
Table 2 Measurement of detection systems..................................................................30
Table 3 Results of similarity measurement................................................................52
Table 4 List of initial variables......................................................................................83
Table 5 Comparison of the detection results.................................................................85
Table 6 Measurement of a sample flow.......................................................................106
Table 7 List of initial variables...................................................................................107
Table 8 List of accuracy measurement.......................................................................109
Table 9 Result of training...........................................................................................114
Table 10 Result of classification................................................................................114
Table 11 List of LDA scores.......................................................................................114
LIST OF FIGURES

Figure 1 Hierarchical diagram of Client-Server DoS network model.........................11
Figure 2 Hierarchical diagram of Typical DDoS network model..........................14
Figure 3 Hierarchical diagram of Distributed Reflector DoS network model.............16
Figure 4 Network diagram of DDoS attack tools....................................................19
Figure 5 Network diagram of DDoS attack botnets..................................................23
Figure 6 DDoS defence strategies with time-line diagram........................................27
Figure 7 A sample network with two traffic flows..................................................45
Figure 8 The discrimination algorithm.................................................................48
Figure 9 The measurements of two normal flows \( \mu = 10, \sigma = 1 \)...................50
Figure 10 The measurements of two Poisson flows \( \lambda = 10 \)...............................50
Figure 11 The metric sensitivity of normal flows \( \mu = 10 \) against standard deviation..........................................................51
Figure 12 The metric sensitivity of Poisson flows against arrival rate........................51
Figure 13 Similarity measure using the Hellinger distance.....................................53
Figure 14 Similarity measure using the Jeffrey distance.......................................53
Figure 15 Similarity measure using the Sibson distance.......................................54
Figure 16 A sample of a server environment.........................................................66
Figure 17 Accumulative arrival rate \( \lambda \) (packet/time interval) from k source IP address(es)...............................................................................................66
Figure 18 Algorithm of Method 1..........................................................................68
Figure 19 Algorithm of Method 2..........................................................................69
Figure 20 Threshold of Decision Making............................................................70
Figure 21 Algorithm of Decision Making by thresholds.................................72
Figure 22 Experiment on generated dataset 1(exponential line) with method 1......77
Figure 23 Experiment on generated dataset 1 (exponential line) with method 2........78
Figure 24 Experiment on generated dataset 2 (peak) with method 1..................79
Figure 25 Experiment on generated dataset 2 (peak) with method 2...............80
Figure 26 Experiment on sample dataset 1 (WC55) with method 1...............86
Figure 27 Experiment on sample dataset 1 (WC55) with method 2...............87
Figure 28 Experiment on sample dataset 2 (MIT17060) with method 1.........88
Figure 29 Experiment on sample dataset 2 (MIT17060) with method 2.........89
Figure 30 Experiment on sample dataset 3 (WC1387) with method 1..........90
Figure 31 Experiment on sample dataset 3 (WC1387) with method 2..........91
Figure 32 Architecture of a traffic discrimination system..........................106
Figure 33 Threshold of Decision Making..................................................109
Figure 34 Correlation plots of the datasets for training and classification phases.111
Figure 35 Entropy plots of the datasets for training and classification phases.....113
Figure 36 Scatter plot of original data for training........................................115
Figure 37 Scatter plot with threshold from LDA training process.................115
Figure 38 Scatter plot with threshold from LDA training process with zoom in from
Figure 37..................................................................................................116
Figure 39 LDA score from training process.................................................116
Figure 40 Scatter plot of original data for classification................................118
Figure 41 Scatter plot with threshold from LDA classification process..........118
Figure 42 Scatter plot with threshold from LDA classification process with zoom
from Figure 41.........................................................................................119
Figure 43 LDA score from classification process........................................119
Figure 44 A packet padding system for anonymous communication...........131
Figure 45 Fingerprints of network sessions...............................................136
Figure 46 Anonymity level against anonymisation progress on web objects............136
Figure 47 Anonymity cost coefficient (ACC) against anonymity level..................137
Figure 48 A packet padding system for anonymous web browsing.......................147
Figure 49 Cost coefficient of perfect anonymity versus sessions with a different
missing rate..............................................................................................................151
Figure 50 Cost coefficient of perfect anonymity against length of session per day for
the dummy packet padding strategy.................................................................152
ACRONYMS

AI - Artificial Intelligence
AIMD - Additive Increase and Multiplicative Decrease
AM - Access Matrix
App-DDoS - Application-layer Distributed Denial of Service
CERT - Computer Emergency Response Team
DCOM - Distributed Component Object Model
DDoS - Distributed Denial of Service
DGSOT - Dynamically Growing Self-Organising Tree
DLL - Dynamic-Link Library
DoS - Denial of Service
DR - Detection Rate
DRDoS - Distributed Reflector Denial of Service
DWARD - DDoS Network Attack Recognition and Defense
FNR - False Negative Rate
FP - False Positive Rate
FTP - File Transfer Protocol
Gbps - Giga-bit per second
GUI - Graphical User Interface
HBDS - Heuristic-based detection system
HsMM - Hidden semi-Markov Model
HTML - Hypertext Mark-up Language
HTTP - Hypertext Transfer Protocol
ICA - Independent Component Analysis
ICMP - Internet Control Message Protocol
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDS</td>
<td>Intrusion Detection</td>
</tr>
<tr>
<td>IGMP</td>
<td>Internet Group Management Protocol</td>
</tr>
<tr>
<td>IIS</td>
<td>Internet Information Services</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>IPS</td>
<td>Intrusion Prevention System</td>
</tr>
<tr>
<td>IRC</td>
<td>Internet Relay Chat</td>
</tr>
<tr>
<td>ISP</td>
<td>Internet service provider</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>LDA</td>
<td>Linear Discriminant Analysis</td>
</tr>
<tr>
<td>LLDOS</td>
<td>Lincoln Laboratory Scenario DDoS</td>
</tr>
<tr>
<td>LSASS</td>
<td>Local Security Authority Subsystem Service</td>
</tr>
<tr>
<td>NIDS</td>
<td>Network Intrusion Detection System</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>P2P</td>
<td>Peer-to-Peer</td>
</tr>
<tr>
<td>PAN</td>
<td>Personal Area Network</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis</td>
</tr>
<tr>
<td>PDoS</td>
<td>Pulsing Denial of Service</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RFC</td>
<td>Request for Comments</td>
</tr>
<tr>
<td>RoQ</td>
<td>Reduction of Quality</td>
</tr>
<tr>
<td>RPC</td>
<td>Remote Procedure Call</td>
</tr>
<tr>
<td>RTT</td>
<td>Round Trip Tim</td>
</tr>
<tr>
<td>SBDS</td>
<td>Statistical-based detection system</td>
</tr>
<tr>
<td>SMTP</td>
<td>Simple Mail Transfer Protocol</td>
</tr>
<tr>
<td>SSL</td>
<td>Secured Socket Layer</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TFN</td>
<td>Tribe Flood Network</td>
</tr>
<tr>
<td>TFN2K</td>
<td>Tribe Flood Network 2000</td>
</tr>
<tr>
<td>TTL</td>
<td>Time to Live</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>WAP</td>
<td>Wireless Access Protocol</td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
</tbody>
</table>
CHAPTER 1

Introduction

1.1 Motivation and Rationale

Service availability and information privacy are key requirements of information security in network communications. For a service provider, service availability is the ability of an online service (e.g. Internet service, email service, or website) to perform its required function at a stated instant or over a stated period of time. Among those services, users have the right to protect their confidential information (e.g. customers’ details, financial information, or new product plan) that may fall into the hands of third party such as a competitor or intruder. Failure to provide service availability and data privacy could result in serious consequences such as asset loss. Distributed Denial of Service (DDoS) attacks and traffic analysis attacks are among the major threats to service availability and information privacy.

DDoS attacks are classified as a very high risk for online services. Basically, a Denial of Service (DoS) attack is the explicit attempt to disable the available service of a victim site or node from a legitimate client(s) by an automatic or manual single attack source. The nature of a DoS attack is to deplete the victim’s resources such as network bandwidth, processing power, or operating system data structures. On the other hand, it is also possible that compromised hosts can coordinate to flood the victim with overwhelming attack packets. The attack takes place simultaneously from multiple-attack sources called a Distributed DoS (DDoS) attack. A DDoS network can be established by compromising many computers infected by malware that acts simultaneously and is coordinated under the control of attacker(s) in order to break
into the victim's system, exhaust resources, and force a denial of service. We have seen that since 1999, DoS attack technology has continued to evolve and remain a serious threat that impacts the service availability on Internet infrastructures, organisations and individual hosts.

Moreover, it is a huge challenge to create a universal DDoS detection. The different attack methods and strategies are starting to challenge DDoS defence systems. Current DDoS attacks are carried out by attack tools, worms and botnets using different packet-transmission rates and packet forms to beat defence systems. These various attack strategies lead to defence systems requiring various detection methods in order to identify the attacks. In particular, discriminating DDoS flooding attacks from flash crowds poses a tough challenge for the network security community.

Because of the vulnerability of the original design of the Internet, attackers can easily mimic the patterns of legitimate network traffic, like flash crowd events, to fly under the radar through the victim. The existing fingerprint or feature based algorithms are incapable detecting new attack strategies.

Traffic analysis attack is an approach that reveals private information through analysis of traffic metadata. Instead of obtaining the content of communication, intruders use traffic analysis attacks which usually focus on whether two entities communicate with each other, for example websites that target users’ access. Attackers try to obtain as much information as possible from traffic metadata, such as message lengths, number of packets, and packet arrival time intervals. The traffic analysis attack is classified into two categories: profiling attack and timing attack. In terms of a profiling attack, adversaries have a list of possible websites and profiles, and the task is to find which ones the target user accesses. Timing attacks are based on the fact that low-latency anonymous systems, such as onion routing, do not introduce any delays or
significantly alter the timing patterns of an anonymous connection. In HTTP service, the time interval of arrival packets of HTML texts and HTTP objects are usually similar for the target users and the adversaries. If they access the same webpage, it is easier for the adversaries to figure out which website the target user accesses from the list.

Finally, when increasing the level of access anonymity for web services, a disabling effect is rendered for any traffic analysis of DDoS detection. In information privacy, users can achieve the highest level of so-called perfect anonymity to protect their credential information from a third party. By deploying this method, DDoS detection is hard to perform data extraction in order to identify the anomalous behaviour of the packet source. Therefore, research on this issue of conflict is necessary for optimisation of DDoS attack detection and web access anonymity.

1.2 Thesis Overview

This thesis aims to improve security on service availability and information privacy web services. In particular, we focus on the problems of DDoS attacks and traffic analysis attacks that impede service availability and jeopardise information privacy. Following, is set of methods to solve these problems.

First, we present DDoS attack detection by differentiating DDoS attack flows from flash crowds. We are motivated by the fact that attack flows are generated aggressively and simultaneously by pre-built programs (attack tools). However, legitimate flash-crowd flows come from an increase in demand of human interest events such as breaking news, new promotion campaigns or sport events. These legitimate flows increase hesitantly in arrival packets on the server side. Therefore,
the self-similarity among DDoS attack flows is much weaker than that among flash crowds. We employ abstract distance metrics, the Jeffrey distance, the Sibson distance, and the Hellinger distance to measure the similarity among flows to achieve our goal. We compared these three metrics to find the one most suitable for our purpose. We apply our algorithms to real-trace datasets and demonstrate our results.

Secondly, we propose a behaviour-based detection that can discriminate DDoS attack traffic from legitimate traffic regardless of the various types of the packets and methods. We noticed that the individual source of DDoS attacks have unique features of repeatable patterns which are different from legitimate flash-crowd traffic. We then perform extraction of the repeatable features from packet arrivals in suspicious flows which are either DDoS traffic or flash crowd traffic. We also propose a comparable detection methods based on the Pearson’s correlation coefficient. The extensive simulations were tested for optimisation of the detection methods. We then performed experiments with some sample real-trace datasets and our results are also demonstrated.

In addition, we propose an effective approach with a supervised learning system based on Linear Discriminant Analysis (LDA) to discriminate legitimate traffic from DDoS attack traffic. Among these behaviours of attack sources, there are repeatable and predictable features differentiated from legitimate sources of traffic such as human users and Internet proxies. We then analyse the real trace traffic from publicly available datasets with triple checks on the repeatable pattern of attack sources. Pearson’s correlation coefficient and Shannon’s entropy are deployed for extracting dependency and predictability of traffic data respectively. Moreover, DDoS defence systems have lack of the learning ability to fine-tune the accuracy of detection results.
Our proposed system deploys Linear Discriminant Analysis (LDA) to train and classify legitimate and attack traffic flows.

Furthermore, we propose a creative approach to achieving a high level of information privacy on web-based services. Based on Shannon’s perfect secrecy theory, we establish a mathematical model for anonymity systems, and transform the anonymous communication problem into an optimisation problem. We also perform further investigation to reduce cost of the cover traffic. Instead of using dummy packets, the prefetched web pages can be used as cover traffic to obtain perfect anonymity in web browsing. Moreover, users may expect a trade-off between the degree of anonymity and the cost. We therefore define anonymity level as a metric to measure the degrees of anonymity. The preliminary experiments on the offline record dataset show the huge potential of the proposed strategy in terms of resource saving.

Finally, we notice there are tradeoffs between the improvement of service availability and information privacy. Because we require a high level of anonymity, the anonymous traffic makes it difficult for DDoS detection to protect service availability. However, if we have to identify the source of a suspicious flow with a high rate of accuracy, the anonymity level of traffic must be reduced and the risk of traffic analysis attack may increase. We believe that our proposed methods could offer a solid foundation of optimisation for security among these research topics.

1.3 Thesis Structure

As discussed in the above sections, service availability and information privacy need to be improved in network communications. We introduce methods to improve methods in service availability against DDoS attack. We then propose and discuss an
improvement method for information privacy against traffic-analysis attack which may affect the detection of DDoS attacks. Hence, the rest of this thesis is organised as follows:

Chapter 2 examines current and historical trends in the DDoS attacks. This chapter reviews the problems of DDoS attacks which consist of various attack strategies and packet-transmission methods. This provides a background to the thesis and allows a reader to appreciate the motivation behind the design of trade-offs explored in later chapters.

In Chapter 3, we perform the similarity measurement of suspicious traffic flows which can be discriminated into DDoS attack flows and flash crowds. The Jeffrey distance, the Sibson distance, and the Hellinger distance are the measurement tools used in order to achieve our goal.

Chapter 4 focuses on examining individual sources of suspicious traffic flows using data dependency as a measurement tool. In this experiment, we monitor for suspicious behaviours such as a repeatable feature of arrival rates from DDoS attack traffic. Using the Pearson’s correlation coefficient as a dependency measurement increases the chance of detecting and discriminating DDoS attacks and flash crowds.

In Chapter 5, we continue to detect dependency and predictability of suspicious traffic by using Pearson’s correlation coefficient and Shannon’s entropy respectively. Then, we improve the learning ability of DDoS detection by deploying Linear Discriminant Analysis (LDA) to train and classify the legitimate and attack traffic flows. By doing this triple check, we can discriminate legitimate and attack traffic flows with high accuracy.
Chapter 6 proposes an approach to improve the level of anonymity in order to achieve information privacy in web-based services. With the replacement of dummy packets by prefetched web pages, we can reach perfect anonymity based on Shannon’s perfect secrecy theory. We provide an adjustable trade-off between the level of anonymity and the cost of cover traffic. This links to the optimisation of service availability and information privacy.

Chapter 7 improves perfect anonymity in web browsing using a novel strategy. We propose a mathematical model based on Shannon’s perfect secrecy theory for solving the problem of high traffic cost. The strategy of using prefetched data as cover traffic is proposed to solve this problem. We also define a metric to measure the cost of the proposed cover traffic and evaluate the results from the experiments.

Finally, Chapter 8 provides a summary of the main points of this thesis. We also discuss further possibilities for research in the future.
CHAPTER 2

Related Literature

2.1 DDoS Attack

DDoS attacks have become an increasing threat to online services over the entire Internet. Such attacks can disrupt and discontinue online services from legitimate users. For instance, in August 1999, a single system at the University of Minnesota was overwhelmed and rendered by a DDoS attack for over two days [1]. Another incident happened in August 2001. The worm named Code Red caused isolated network conditions due to high scanning and propagation rates [2]. Computer Economics [3] estimated that the total economic impact of Code Red was $US 2.6 billion. Recently, the attacks [4] against the Estonian government and corporate Websites in April 2007 and the attacks [5] against the Georgian government and news Websites in August 2008 were the most notable incidents of regional hacktivism, with hacktivism presenting a real threat of DDoS attack and the impact on national infrastructure. A recent report [6] has revealed the largest attack size doubled year-by-year, to more than 100 Gbps which is a surprising 1000% increase in attacks size since 2005.

A DoS attack is an explicit attempt to disable the available service of a victim site or node from a legitimate client(s) by an automatic or manual single attack source. The nature of a DoS attack is to deplete the victim's resources. These resources can be network bandwidth, processing power, or operating system data structures. We have seen that since 1999, DoS attack technology has continued to evolve and has remained a serious threat with significant impact on Internet infrastructures,
organisations and individual hosts [2]. However, it is also possible that compromised
hosts coordinate to flood a victim with overwhelming attack packets. The attack takes
place simultaneously from multiple-attack sources called a Distributed DoS (DDoS)
attack. A DDoS network can be established by compromising many computers
infected by the malware that acts simultaneously and is coordinated under the control
of an attacker(s) in order to break into the system of the victim, exhaust resources, and
force a denial of service. There are two main types of DDoS attacks: typical DDoS
attacks and distributed reflector DoS (DRDoS) attacks [7].

As a variant of DDoS attacks, Reduction of Quality (RoQ) attacks are carried out to
challenge DDoS defence systems. An RoQ attack is a low-rate DDoS attack that
attempts to degrade the Quality of Service (QoS) to a victim’s system, but not
completely deny the services from legitimate client(s) [8, 9]. An RoQ attack is also
known as a Shrew attack or a Pulsing DoS attack (PDoS) [10, 11]. This type of attack
is more difficult to detect than the flooding DDoS attacks. In contrast, typical DDoS
flooding attacks are characterised by a sustained high-rate or high volume of packet
transmission, however a RoQ attack eludes detection of DoS defense mechanisms by
sending a sufficiently low-average rate. This class of new attacks can be further
categorised into timeout-based attacks and AIMD-based attacks (AIMD stands for
Additive Increase and Multiplicative Decrease), depending on the timing of the attack
pulses with respect to the TCP’s congestion window movement [12, 13].

In this chapter, we address two complementary problems and goals: (1) a taxonomy
for classifying DDoS attack stratagems, and (2) reactive defence strategies. We
provide more details about DoS, DDoS, and reactive defence techniques including the
trend and evolution of each topic section. Section 2.2 proposes DDoS network models
that can be used for investigation and forensics purposes. In Section 2.3, we collect
more information about tools, worms and botnets that were part of previous DDoS attacks around the world. We also analyse the trend of DDoS attacks from the past and consider the possible trend of DDoS attacks in the future. Section 2.4 provides an introduction to DDoS defence mechanisms with some details of the reactive mechanisms. Section 2.5 classifies detection techniques that have been deployed in organisations and developed in research. Section 2.6 introduces traffic analysis attack and its possible threat to information privacy. Section 2.7 discusses the challenges of DDoS attacks to provide general knowledge for the remaining chapters. In the final section of this chapter, we summarise all the information we have discussed about DDoS.

2.2 DDoS Network Models

This section proposes DDoS network models that can be used for investigation and forensic purposes. These models cover any current crime scene of a DDoS attack. By classifying the attacker-victim relationship of a DDoS network, these models can be divided into three different categories:

- Client-Server DoS network model
- Typical DDoS network model
- Distributed Reflector DoS network model

The details of these classified models are given in the following sub-sections.

2.2.1 Client-Server DDoS Network Model

A Client-Server DoS network model is based on the primitive network model of DoS attacks [14]. As a direct DoS attack, the attacker client generates and directly sends out a large number of attack packets to a victim. It not only impacts the destination
victim, but the network resources between the attacker and destination victim are also directly affected by the attack. As shown in Figure 1, the proposed model considers the main attack components with the additional layer from the reflected attack. The model consists of the following layers:

![Hierarchical diagram of a Client-Server DoS network model](image)

Figure 1 Hierarchical diagram of a Client-Server DoS network model

Table 1 A sample of victim responses to typical attacks [15]

<table>
<thead>
<tr>
<th>Packet Sent</th>
<th>Response from Victim (Backscatter)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP SYN (to open port)</td>
<td>TCP SYN/ACK</td>
</tr>
<tr>
<td>TCP FIN (to close port)</td>
<td>TCP RST (ACK) and TCP FIN (to close port)</td>
</tr>
<tr>
<td>TCP ACK</td>
<td>TCP RST (ACK)</td>
</tr>
<tr>
<td>TCP DATA</td>
<td>TCP RST (ACK)</td>
</tr>
<tr>
<td>TCP RST</td>
<td>No response</td>
</tr>
<tr>
<td>TCP NULL</td>
<td>TCP RST (ACK)</td>
</tr>
<tr>
<td>ICMP ECHO Request</td>
<td>ICMP ECHO Reply</td>
</tr>
<tr>
<td>ICMP TS Request</td>
<td>ICMP TS Reply</td>
</tr>
<tr>
<td>UDP packet (to open port)</td>
<td>Protocol dependent</td>
</tr>
<tr>
<td>UDP packet (to close port)</td>
<td>ICMP port unreachable</td>
</tr>
</tbody>
</table>
a) **Attacker Layer** consists of attackers acting in mainly an infamous role during DoS attacks. In high-rate DoS attacks, the attacker client(s) generates and directly sends out a large number of attack packets to the victim. Not only is the target victim, but also the network resources between the attacker and target victim are directly affected by the attacks.

b) **Target Victim Layer** consists of a single or a group of victims which are targeted by DoS attacker(s). In high-rate DoS attacks, the victim’s host and resources forcibly receive a large number of attack packets. Without proper DoS defence systems, the victim’s resources may be disrupted and forced down. This potentially causes a DoS.

c) **Side-Effect Victim Layer** introduces additional layers after the Target Victim Layer in order to represent the secondary effects of DoS attacks. This layer may consist of third-party hosts or network entities, also called backscatter victims. The side-effects can occur when the attacker generates a spoofed source IP address in the IP header of each attack packet. If the attacker generates source IP addresses randomly, the backscatter response packets from the victim will be sent back to random destinations as shown in Table 1. For example, some attack packets, such as ICMP requests, require a response (reply) from the victim’s host or network equipment. These reply messages may be sent directly to other hosts instead of the original attack sources. This effect can be collected by network telescopes as indirect evidence of such attacks. For investigation and forensics purposes, we propose and divide this layer into two sub-classes of third-party (backscatter) victims:

- **Desired Effect Victim** is the third-party victim the attacker specifies in the source IP address in each attack packet. These third-party victims may
impact the increasing volume of traffic from the reply messages. For any reason, the target victim may believe that all attacks come from third-party victims. The attacker can take advantage of this and hide the attacker’s clients, blame the scapegoats and make the investigation process harder. There is a highly risky potential that the attacker can cause conflict between the third-party and target victims if they lack proper defence mechanisms, transparent and there is strong evidence. We hope this scenario never arises up like the cyber-war incident between Estonia and Russia in April 2007 [4] and the incident between Georgia and Russia in August 2008 [5].

- **Adverse Effect Victim** is the third-party victim whose source IP address is not specified by the attacker in each attack packet. This sub-class refers to the nature of the most DDoS attacks with randomly generated forged source IP addresses. Although these third-party victims may rarely be impacted with increasing volumes of traffic from the reply messages, the target victim may still believe that all attacks came from third-party victims. The attacker can take this advantage of this by hiding the attacker’s clients and making the investigation process harder. In addition, the flooding with spoofed source IP addresses can amplify the attack when the reply message does not reach the destination hosts. The router will generate and return victim ICMP “Host Unreachable” packets and TCP RST packets if the destination does not exist or the TTL has reached 0 as shown in Table 1.

#### 2.2.2 Typical DDoS Network Model

Typical DDoS network mode is based on a hierarchy of DDoS attack tools. The agents are representatives of the attacker to offend the victim. An agent can be a botnet or zombie that is running the daemon program that listens to the attacker’s
commands. The attacker may create additional computers, called *handler* or *master server*, in order to issue the command to the compromised agents. The attacker may directly issue the attack commands to the handler or connect the handler by a computer host, called the *attacker client*. This technique can avoid resource limitation problems and render attack tracking more difficult. A set of agents may be created in order to amplify the attack by establishing DDoS. With various techniques and programs for compromising and attacking the victim, this group of computers is created as a *DDoS network* and this set of programs using in these computers is created as a *DDoS attack tool*. As shown in Figure 2, the proposed model may considerate the main attack components and additional layers that are affected from the reflected attack. This model consists of the following layers:

![Hierarchical diagram of a Typical DDoS network model](image)

**Figure 2 Hierarchical diagram of a Typical DDoS network model**

**a) Attacker Layer** consists of the attackers as described in Section 2.2.1. However, in this model, the attacker client(s) may not directly attack the target victim but issues attack controls and commands to its masters.
b) **Master Layer** is an additional layer between the attackers and the target victims in order to conceal the attacker’s clients from the investigation. The layer consists of master servers or handlers which are controlled by the attacker’s client. Not only listening to the commands issued from the attacker, handlers also have responsibility to compromise a group of machines in case of DDoS attack. When the commands are issued, a handler may provide a translation or forward them to the compromised machines belonging to this handler. The handler is the key control point and the effective anonymiser of the network that the attacker has additional care of [16]. In order to eliminate a single point of failure, more than one handler is found in practice, and in most cases each handler has equal power over its agents [16].

c) **Slave Layer** is an additional layer between the attackers and the target victims in order to conceal the attacker’s client and handler from the investigation. This layer consists of slave machines (also known as daemons, zombies, agents, or botnets). Each slave agent may report and listen to a single or group of handlers, but does not have direct contact to the attacker’s client. In high-rate DDoS attacks, instead of direct attack from an attacker's client, the slave agents generate and send out large numbers of attack packets directly to the victim. Not only is the target victim affected, but the network resources between the slave agents and target victim are also directly affected by the attacks.

d) **Target Victim Layer** consists of a single or a group of victims as described in Section 2.2.1(b).

e) **Side-Effect Victim Layer** introduces additional layers after the Target Victim as described in Section 2.2.1(c).
2.2.3 Distributed Reflector DoS Network Model

Distributed reflector DoS (DRDoS) network mode is based on DDoS network architecture with reflectors. DRDoS is also called *Distributed Reflection DoS* or *Distributed Reflected DoS*. Reflector-based attack is an effect from a spoofed source IP address. An attacker may alter the source IP address in the attack packets, which are aimed at the victim. The attack packets are then transmitted to a third-party computers or network devices. Without proper spoofed IP checking, replies to the requests will be made as common services. Unfortunately, the reply packets will be transmitted directly to the target victim as the source IP address specified rather than transmitted to the original attacking sources. As shown in Figure 3, the proposed model considers the main attack components and additional layers that were affected from the reflected attack. The model consists of the following layers:

a) **Attacker Layer** consists of the attackers as described in Section 2.2.1(a). However, in this model, the attacker client(s) may not directly attack the target victim but issue attack controls and commands to its masters.
b) **Master Layer** is an additional layer between the attackers and the target victims in order to conceal the attacker’s client from the investigation. This layer consists of master servers or handlers as described in Section 2.2.2(b).

c) **Slave Layer** is an additional layer between the attackers and the target victims in order to conceal attacker’s client and handler from the investigation. This layer consists of slave machines as described in Section 2.2.2(c). However, in high-rate DDoS attacks, instead of a direct attack to the target victim, the slave agents generate and send out a large number of attack packets directly to other machines called reflectors in order to aggregate the reflected response/reply messages to add the target victim. In fact, the DDoS attack with reflectors needs to alter the source IP address into the victim’s IP address. The destination IP address that targets reflectors may be generated randomly or specifically.

d) **Reflector Layer** consists of a single or a number of reflector machines. The reflectors are not the infected or compromised machines. However, they can be used as an attack relay or amplifier to gain the degree of damage. They are also used to conceal the main network architecture (consisting of attackers, masters and slaves) from the investigation.

e) **Target Victim Layer** consists of a single or group of victims as described in Section 2.2.1(b).

### 2.2.4 Evolution of the DDoS Network Architecture

Early DoS attack technology involved simple tools that generated and sent packets from a single source aimed at a single destination. In June 1999, DoS tools were deployed and evolved to execute a single source attack against multiple targets,
multiple source attacks against single targets, and multiple source attacks against multiple targets [2].

The trend of attacks is to deploy multiple-attack sources against single-target attacks since 1999. The distributed computing models have evolved and coordinated many one-to-one attacks that sufficiently escape the traditional model. Rather than relying on a single attack source, attackers can now take advantage of thousands of more systems to force a DoS to the victims [16].

Today, the most common type of DoS attack involves sending a large number of packets, called *packet flooding attack*, to a destination. This causes overwhelming disruption for the host and for network performance, including the availability of victim sites, routers, servers and even firewalls [2]. Packet flooding attacks also affect the upstream Internet Service Provider (ISP) [17]. Based on reported DoS activity, multiple-target attacks are less common [2].

Most DDoS attacks have involved thousands of compromised host systems that were external to the victim’s own system or network. In many cases, the launch point consists of one or more systems that have been subverted by an attack via a security-related compromise rather than from the attacker’s own system or systems [2].

### 2.3 DDoS Attack Sources

The mass-intrusion phase is an initial installation of DDoS attack architecture. Automated tools are utilised to compromise a large number of computers remotely. These compromised systems then become DDoS server (handlers) in the Master Layer and DDoS agents (zombies) in the Slave Layer. Therefore, these computers are the primary victims [18]. After the mass-intrusion phase, these compromised systems, which constitute the DDoS handlers and DDoS agents, are ready to begin the attack.
For example, a Stacheldraht handler can control up to 5,000 agents [17]. The DDoS systems are artificially generate massive attacks against one or more targeted systems and its resources. These are secondary victims in the Victim Layer [18].

Most DDoS attack architecture is implemented in a 4-tier client/server model. As shown in Figure 4, the attacker has to install the front-end client and communicates with the handlers. The handler controls a number of agents on a compromised system to perform DDOS attacks [17]. We have collected many DDoS attack tools, worms and bots as we have seen many infamous attack incidents around the world.

![Network diagram of DDoS attack tools](image)

**Figure 4** Network diagram of DDoS attack tools

### 2.3.1 DDoS Attack Tools

In the past decade, DDoS attack tools have been developed from DoS attack programs, hacking programs, malware, client-server communication models, and
hactivities. The following lists examples of well-known and infamous DDoS attack tools.

a) **The early DDoS attack tools** appeared in 1998. These were clumsy attempts to naturally evolve beyond coordinated attacks, but nevertheless laid the foundation for the subsequent tools. The first of them, fapi, featured UDP, TCP (SYN and ACK), and ICMP Echo floods. Its handler to agent communication was UDP-based. It did not provide easy controls for setting up the DDoS network, and did not handle networks over 10 hosts very well. The second one, fuck_them, was a distributed ICMP Echo Reply flooder, where the attacker either supplied the source address to spoof or randomised source addresses were generated (all 32 bits of the IP address) [16].

b) **Trinoo (or Trin00)** is the first well known distributed network DDoS tool which was used to mount an attack against a system at the University of Minnesota in August 1999 [1, 19]. Trinoo tools are made up of master (server) and slave (daemon) programs. A Trinoo network is able to set up thousands of systems on the Internet that have been compromised by a remote buffer overrun by exploitation. A Trinoo network is carried out by an attacker (intruder) connecting to a Trinoo master and giving instruction to launch a denial of service attack against one or more IP addresses. The Trinoo master then communicates with the slave daemons giving instructions to attack one or more IP address for a specified period of time. The tool is capable of only generating UDP packets floods [16]. However, source addresses were not spoofed, so systems running the offending slave daemons were contacted [1, 16]. **WinTrinoo** is a Trinoo variant version which is able to run on a Microsoft
Windows Operating systems that was first reported to CERT on February 2000 (CERT IN-2000-01) [20-22].

c) **Mstream (and Mstream2)** is a primitive multiple-stream tool with a very efficient point-to-point stream TCP ACK flood [16]. An Mstream agent was discovered in late April 2000 on a compromised Linux system at a major university [23]. With very limited control and incomplete features compared to earlier DDoS attack tools, Mstream seemed to be in its early development stages. However, the year-2000 version can spoof the source IP address by randomising all 32 bits [16].

d) **Other DDoS attack tools** are Tribe Flood Network (TFN), Stacheldraht, Carko, Shaft, Omega, Trinity, MyServer, Plague, Knight, Kaiten, etc. [2, 14, 16-19, 24-27].

### 2.3.2 DDoS Attack Worms

Malware can carry DDoS attack mechanisms that perform attacks without controlling or issuing a command from an attacker. Because the DDoS malware has been programmed to attack at a specific date and time to the specific targets by specific attack methods, the attack could proceed automatically whether or not it needed synchronisation between each other. The following are examples of well-known and infamous DDoS attack worms.

a) **Nimda worm/virus** was isolated in September 2001. The worm’s name spelled backwards is admin. Nimda affected Windows platforms such as Windows 95, 98, Me, NT, and 2000. Nimda exploits various Microsoft IIS 4.0 / 5.0 directory traversal vulnerabilities. Multiple propagation vectors allowed Nimda to become the Internet’s most widespread virus/worm. The high scanning rate of
the Nimda worm may also cause bandwidth DoS conditions on networks with infected machines [28].

b) **Code Red worm** exploits the Buffer Overflow vulnerability in the Indexing Service on systems running Microsoft IIS [27]. Code Red included functionality to launch a TCP SYN flood attack against a specific target. The worm also caused isolated DoS conditions due to high scanning and propagation rates [2]. More than 250,000 systems were infected with Code Red in just 9 hours on 19 July 2001 [3, 29]. Computer Economics estimated that the total economic impact of Code Red was $US 2.6 billion [3]. The **Code Red II worm** began to propagate much like the earlier Code Red worm in August 2001. The worm exploits the Buffer Overflow in Microsoft IIS 4.0 Servers with Indexing Service DLL and URL Redirection Enabled [2].

c) **Other DDoS attack worms** are Power, Blaster, Mydoom, etc. [27, 30, 31].

### 2.3.3 DDoS Attack Botnets

Rather than relying on a handler network, DDoS attack botnets take advantage of an existing Internet Relay Chat (IRC) network for its handler-to-agent communications, and makes the handler a channel on IRC. This attack architecture compromises a large number of computers, which later install an Internet robot application called bot. The bot typically connects automatically to a remote IRC server to enable remote control by the attacker. The controlled bot is then transformed into a botnet which is used for generating spam emails, viruses, and worms as well as DDoS attacks [32].

This section presents 4 infamous botnets, namely Agobot, RBot, and Storm worm.

Note that these botnets have a few hundred to a thousand variants due to multiple authors working to enhance the exploitation, propagation and communication code. Hence, we present the enhanced version with the most advanced DDoS attack tools.
Figure 5 Network diagram of DDoS attack botnets.

a) **Agobot (Phatbot)** is one of the most popular bots with over 600 different versions [32, 33]. Variants of Agobot include Gaobot, Nortonbot, Phatbot and Polybot. The bot was written in C++ and provides cross platform capabilities. Its structure is designed into modules and allows extension for additional modules. Agobots have many features such as a password protected IRC client, remotely updating and removing the installed bot, executing programs and commands, port scanner to search and infect other hosts, and DDoS attacks [33]. This has the most comprehensive set of DDoS attack tools that combine attack features such as SYN flood, UDP flood, ICMP flood, HTTP flood and TARGA3 attack [32]. Agobot may contain other features such as packet
sniffer, keylogger, polymorphic code, rootkit installer, Information harvest, SMTP client, or HTTP client [33].

b) **RBot (GTBot)** has over 1600 variants. It is also written in C++ and targets Windows systems. RBot and its variants have the features of the Local Security Authority Subsystem Service (LSASS) to exploit and master passwords for scanning and compromising Optix servers. Their attack features are SYN flood, ACK flood, random (SYN or ACK) flood, UDP flood, ICMP flood, and ping flood [32].

c) **Storm worm (bot)** created DDoS attacks against a number of anti-spam websites on January 12, 2007 [34]. It has been estimated that 85,000 machines have been compromised into botnets with 35,000 botnets sending 3 billion spams emails per day [35]. Strom worm is a variant of Nuwar and also called Small.DAM, Peacomm, Zhelatin, Dorf, Downloader, SMALL.EDW, Zhelatin, Peed, and Tibs. The DDoS attack was conducted by game4.exe. The control and communication are via HTTP on random ports with base64/zlib encoding, P2P-based server directory [34]. It received the target IP address and attack type by downloading a configuration file from a hard-coded website in the body of the Trojan. Attacks can be either a (port 80) TCP/SYN flood, ping flood, or both. The configuration file specifies the target by IP address alone because the tool has no provisions to resolve DNS names to addresses [34].

d) **Other DDoS attack botnets** are SDBot, Spybot, Reptile, ZoTob, PBot, Tsunami, Kelvir, MetaFisher, etc. [32, 36].

2.3.4 **Evolution of the DDoS Attack Sources**

Hybrid versions of DDoS attacks are being increasingly developed. From time to time, DDoS attack tools and malware have benefited and learnt from each other.
Malicious and offensive tactics and techniques have been added into the hybrid versions that attack victims. As we have seen in late August/early September 1999, there was a shift from the well-known DDoS attack tool, Trinoo, to TFN. Then one month later, the hybrid version, Stacheldraht, began attacking systems in Europe and the United States [18]. From analysis [18], it was obvious that the encryption of communication channels and more automated maintenance of large networks was in active development. With the impressive features of malware and bots, DDoS attack tools aim to advance DDoS attack botnets which can cause massive destruction to the site of victims. DDoS attack tools have implemented the features of propagation from malware such as worms, viruses, and Trojans. As we witnessed the high-scanning rate and high-propagation rate of Nimda, Code Red, and Blaster, the hybrid botnets implement these features in order to compromise as many slave agents as possible on the Internet. For example, recent reports [37] show that the Kraken botnet has compromised at least 400,000 machines sending 200 billion spam messages in a day. In particular, the DDoS attack botnet named Storm, has been estimated to have compromised 85,000-200,000 machines with only 35,000 botnets sending 3 billion spam email per day and attacking distinct victims [35, 37, 38].

In addition, DDoS attacks are increasing interoperability and independency of platforms. TFN and TFN2K have been found to be the earliest DDoS tool available on the Windows platform, while Stacheldraht to date, only works on a UNIX platform [17]. Windows machines have been used as agents producing various types of DoS attacks for years [21]. However, MacOS 9 can also be used as a traffic amplifier, to flood victims with high traffic volume. An attacker [19] can use this asymmetry to amplify traffic by a factor of approximately 37.5, thus enabling an attack with limited bandwidth to flood a much larger connection. This is similar in effect and structure to
a SMURF attack but it is not necessary to use a directed broadcast to achieve traffic amplification.

Moreover, there is an increasing level of automation for attack sources. Advances in automation techniques for new self-propagating worms in 2001 have been used to deploy DoS attack technology [2]. In the past, most DDoS attack programs, later called DDoS attack tools, were often installed onto compromised machines mostly by manual means. Over time, attackers have developed and employed advanced automation of scanning patterns in multiple aspects of DoS attack technology deployment. Earlier scanning programs/tools manually provided lists of potentially vulnerable hosts which meant it was easier to exploit these machines. The next step was the addition of automated tools in an attempt to exploit and record lists of compromised hosts that later become handlers and slave agents. These types of lists were often used to exploit vulnerable systems and install attack tools. Today, similar to worm’s self-propagation, automatic widespread scanning is one of the self-propagation techniques used to find new vulnerable hosts. Scanning activity is an initial phase of automation to compromise potential of slave agents to be employed by attackers. Automation of high-rate scanning and self-propagation of attack tools effectively become a DoS attack if they are high enough to reach the point. For instance, we have seen DDoS worms like Code Red and Nimda self-propagate to a point of global saturation in less than 18 hours which caused DoS in some organisations’ networks [3].

Furthermore, there is increasing use of IRC channels as a communication tool. Recently, we have seen that control mechanisms for DDoS attack networks are shifting to use the technology of Internet Relay Chat (IRC) [2]. Attacker use of IRC protocols and networks as a communication channel for DDoS networks essentially
removes the unnecessary exploitation and replaces the useful functions of a handler in primitive DDoS network models. IRC-based DDoS networks are sometimes referred to as botnets, which alludes to the concept of bots on IRC networks as being software driven participants rather than human participants. For example, Entitee and Trinity v3 have both been found on the Undernet Internet Relay Chat (IRC) network by the Undernet operators, each using different IRC channels [14] [24]. Later we have also found that Agobot, SDBot, RBot, Spybot and the Storm worm successfully used IRC protocol as a backbone for a communication channel and IRC servers as handlers for DDoS attacks [32]. Other public communication channels such as Simple Mail Transfer Protocol (SMTP), HyperText Transfer Protocol (HTTP) and Instant Messaging (IM) are available for DDoS communication networks [3, 33]. For example, Agobot provides features for an SMTP and HTTP client to issue DDoS commands [33]. As a result, it has become increasingly difficult to differentiate attack/anomaly signatures from normal/legitimate network traffic.

### 2.4 DDoS Reactive Defence Mechanisms

![Diagram showing DDoS defence strategies with timeline diagram.](image)

Figure 6 DDoS defence strategies with time-line diagram.
In general, DDoS defence strategies concern 3 main activities: (1) Prevention, (2) Detection, and (3) Response.

The prevention strategy aims to (1) eliminate the possibility of a DDoS attack altogether, (2) mitigate the effect of a DDoS attack before the zero-day attack begins and (3) enable potential victims to endure the attack without denying services to legitimate clients [39]. These could be successful by implementing proactive mechanisms such as router filtering (Ingress and Egress), and Intrusion Prevention System (IPS).

- **Ingress filtering** is a packet filtering technique employed by ISPs against source spoofing addresses of Internet traffic. Incoming packets need to prove that which network they came from.

- **Egress filtering** is a packet filtering technique employed by an internal network against unauthorised use of machines. The filter is implemented in the router to make sure that outgoing packets are safe when they leave from the internal network to external networks.

- **Intrusion Prevention System (IPS)** is a network security devices that detect network and/or system activities for malicious or unwanted behaviour, and can react in real-time to block or prevent those activities. Intrusion prevention technology is considered by some to be an extension of intrusion detection (IDS) technology.

While ingress and egress filters at the border routers can limit the problems caused by attacking agents faking source addresses, IPS will proactively operate in-line to monitor all network traffic for malicious code or attack. However, the filtering method does not effectively protect against flooding attacks which originate from
valid prefixes (IP addresses), but it will prohibit an attacker within the originating network from launching an attack of this nature using forged source addresses that do not match to ingress filtering rules. For example, TFN2K has specifically been designed to break the security rules of ingress filtering [19].

Reference [18] suggests that RFC-2267 style egress filtering may protect attack packets from somewhere within border routers, or on each subnet. Ethernet switches will make this more difficult to attack local subnets, which mean an intrusion detection system (IDS) just inside the borders would be a recommended way of detection for the entire network.

The next section will explain the reactive mechanism for DDoS defence (also referred to as Early Warning Systems and Early Detection Systems) which is responsible for detecting the attack and respond to it immediately [7]. Because DDoS attacks can threaten the availability of victim’s service, the reactive mechanism shall restrain the degree of impact on the victim site.

### 2.5 DDoS Detection Strategies

Detection is an important process to extract and discriminate DDoS attacks from legitimate network activities. DDoS attack detection is normally implemented on an Intrusion Detection System (IDS), in particular a *Network Intrusion Detection System* (NIDS).

As a statistical result in Table 2, the results of the DDoS detection systems are measured by the following criteria:
Table 2 Measurement of detection systems

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Detected</th>
<th>In Fact</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positive</td>
<td>Legitimate Activity</td>
<td>Legitimate Activity</td>
</tr>
<tr>
<td>True Negative</td>
<td>Intrusive Activity</td>
<td>Intrusive Activity</td>
</tr>
<tr>
<td>False Positive</td>
<td>Intrusive Activity</td>
<td>Legitimate Activity</td>
</tr>
<tr>
<td>False Negative</td>
<td>Legitimate Activity</td>
<td>Intrusive Activity</td>
</tr>
</tbody>
</table>

- **True Positive** occurs when IDS correctly classifies legitimate activities as unharmed activities.
- **True Negative** occurs when IDS correctly classifies intrusive activities as intrusions.
- **False Positive** occurs when IDS incorrectly classifies legitimate activities as intrusions.
- **False Negative** occurs when IDS incorrectly classifies intrusive activities as unharmed activities.

Most of the research we surveyed on detection mechanisms involved a trade-off between false positive and false negative which results in an acceptable level. A threshold value could be concerned with an issue of accuracy for DDoS detection [40], with the more restrictive tests increasing the risk of rejecting true positives, and the more sensitive tests increasing the risk of accepting false positives. Hence, the threshold value can be used to optimise the best results for the tests.

The main detection strategies classified by detection criteria are signature-based detection, anomaly detection, hybrid detection systems and third-party detection.

**2.5.1 Signature-based detection**

Signature-based methods scan and monitor patterns (signatures) in observed network traffic that match known attack signatures from a database. Signature-based detection that implements intrusion detection is also known as misuse detection or pattern matching detection [7, 41, 42].
These methods can easily and reliably detect known attacks with no false positives encountered. However they cannot recognise new attacks or even slight variations of old attacks. Moreover, like anti-virus programs, the signature database must always be kept up-to-date in order to retain the reliability of the system for new attacks [7].

2.5.2 Anomaly detection

Anomaly detection derives from heuristic-based detection and is also known as anomaly-based detection or profile-based detection which compares the parameters of observed network traffic with normal traffic. In other words, anomaly detection attempts to identify behaviour that does not conform to normal behaviour [43].

A heuristic-based method is an expert-based analysis using an algorithm to determine the susceptibility of a system towards a particular threat/risk using various decision rules or weighing methods. Heuristic-based methods are also employed by many anti-virus programs that are designed to detect previously unknown computer viruses, in addition to new variants of viruses already in the wild.

A Network Intrusion Detection System (NIDS) also employs heuristic-based methods using behavioural patterns of users, applications and other program files to develop a pattern of normal and abnormal behaviour, which is then used to detect the occurrence of an attack [40]. Although the detection can be prone to a large number of false positives, it is possible for new attacks to be detected. However, in order to prevent a false alarm, the model of normal traffic must always be kept up-to-date and the threshold of categorising an anomaly must be properly optimised [7]. Based on the specification of normal behaviour, we divide anomaly detection mechanisms into standard and trained mechanisms [39].
a) **Standard mechanisms** use standard specifications of normal behaviour that rely on some standard protocol or set of rules. The advantage of a standard-based specification is that it generates no false positives; all legitimate traffic must comply with the specified behaviour. The disadvantage is that attackers can still perform sophisticated attacks which, on the surface, seem compliant to the standard and thus pass undetected [39].

b) **Trained mechanisms** use trained specifications of normal behaviour to monitor network traffic and system behaviour and generate threshold values for different parameters. All communications exceeding one or more (depending on the approach) of these values are regarded as anomalous. Trained models catch a broad range of attacks, but have two disadvantages. These are (1) Threshold setting. Anomalies are detected when the current system state differs from the model by a certain threshold. The setting of a low threshold leads to many false positives, while a high threshold reduces the sensitivity of the detection mechanism. (2) Model update. Systems and communication patterns evolve with time, and models need to be updated to reflect these changes. Trained specification systems usually perform automatic model updates using statistics gathered at a time when no attack was detected. This approach makes the detection mechanism vulnerable to a slow increase in the rate of attacks that can, over a long period of time, delay or even avoid attack detection, or mislead models [39].

### 2.5.3 Hybrid detection system

A hybrid detection system combines both previous detection methods. These systems update their signature database with attacks detected by anomaly detection. Again the danger is significant because an attacker can fool the system by characterising normal
traffic as an attack. In this case, an Intrusion Detection System (IDS) becomes an attack tool. Thus IDS designers must be very careful because their research can have repercussion [7].

Non-dependence upon signatures and the use of statistical and behavioural patterns as a mean to detect new types of malicious code allows for a low false negative rate.

2.5.4 Third-party detection

Mechanisms that deploy third-party detection do not handle the detection process themselves, but rely on an external message that signals the occurrence of an attack and provides characterisation of attack. Examples of third-party detection are easily found among traceback mechanisms [39].

2.6 Traffic Analysis Attacks

Traffic analysis is the process of intercepting and examining messages in order to infer information from patterns in communication. The information is inferred and extracted for network meta-data, including volumes and timing of network packets, as well as visible network addresses they are originated from and destined for [44]. In general, the greater the number of messages observed, intercepted and or even stored, the more that can be inferred from the traffic.

Traffic analysis can be a concern of computer security. In the case of anonymous communications, an adversary would use this data to perform traffic-analysis with the aim of tracing the originator or the ultimate destination of a connection, therefore violating the anonymity properties that the system is designed to provide [44].
In information security, a traffic analysis attack is used against the encryption mechanism. In general, traffic analysis can be used to resolve what type of information is being communicated such as chat, email, web page requests, etc., even if the data itself is scrambled, or encrypted. To reveal encrypted information, an attacker is required to monitor the frequency and timing of network packets.

A timing attack can use timing information as a way to reveal further information. In SSH protocol, the time between keystroke messages can be studied using hidden Markov models [45]. During an interactive session, SSH transmits each keystroke typed by a user which is sent to a remote machine in a separate IP packet. Learning the inter-keystroke timing information of typed passwords allows eavesdroppers to deduce the content of passwords. This suggests that SSH is not as secure as commonly believed.

Onion routing systems are used to gain traffic anonymity. Traffic analysis can be used to attack anonymous communication systems like the Tor anonymity network [46]. We used to believe that Tor intermediaries, through the use of encrypted tunnels, effectively hid the bit patterns of data travelling through a Tor connection. An adversary therefore, cannot use any information from the content to trace the stream and has to resort to traffic analysis. However, present research [44] showed that traffic analysis allows adversaries to infer which nodes relay the anonymous streams. This reduces the anonymity provided by Tor. Research has also shown that otherwise unrelated streams may be linked back to the same initiator.

Packet padding is an approach to resist a timing analysis attack. In a low-latency onion routing network, data flows are carried by a variable rate. Packets cannot be delayed too much, or dropped in order to satisfy the requirements of quality of service
(QoS). Therefore, to conceal the relationship between incoming and outgoing traffic flows, dummy traffic (padding) must be added to the original data flows [47]. The use of dummy traffic is used to send random data in addition to normal communications. Data packets leaving each node are augmented by dummy packets which the adversary cannot distinguish from (encrypted) real data packets. This can be inconsistent, with more bandwidth and processing power being used, which is usually an optional feature of a secure connection.

2.7 Research Challenges

While most ISPs now have the infrastructure to detect bandwidth flood attacks, many still lack the ability to rapidly mitigate DDoS attacks with some detection and mitigation systems still taking a long time [48]. Rate limiting, packet filtering, and reconfiguration application parameters can, in some cases, mitigate and limit the impact of DoS attacks, but usually only at points where the DoS attack is consuming fewer resources than is available. In many cases, the only defence is a reactive one where the source or sources of an ongoing attack are identified and prevented from continuing the attack. Defending against DoS attacks is therefore, far from an exact or complete science [2].

In general, DDoS detection methods are based on features or fingerprints of specific DDoS attacks. These include activity profiling, sequential change-point detection, wavelet analysis, and chi-square/entropy. Unfortunately, it is very easy for hackers to mimic these features to fool user detection methods. For example, because of the open architecture of the Internet, hackers can spoof the source IP addresses of attack packets according to real Internet IP address distribution against our source address.
distribution based detection algorithms. Hackers can also change the TTL value of the attack packets against our hop-count detection methods according to the real hop distance between zombies and victims respectively.

Lack of flexibility in DDoS detection can cause a rise in false positives/negatives. A counter attack method cannot follow ever changing attack methods, as the attack patterns occasionally change, and the attacker may mimic the network traffic patterns of flash crowds, causing the detector to be quickly disabled. The entropy detector can raise the alarm for crowd access, however, it cannot discriminate DDoS attacks from the surge of legitimate accesses, e.g. flash crowds. The change-point detection method can increase the number of attack packets very slowly but this method can be easily deceived which will almost surely disable the change point detectors, e.g. zombies can increase the number of attack packets very slowly.

Discriminating DDoS flooding attacks from flash crowds poses a tough challenge for the network security community. Because of the vulnerability of the original design of the Internet, attackers can easily mimic the patterns of legitimate network traffic, like flash crowd events, to fly under the radar through a victim. For example, in a flash crowd event, many sports fans will access the official website when an important sports match takes place or many people will the check CNN website when breaking news occurs. Attackers may mimic the behaviours of these flash crowds which become a sudden increase of legitimate traffic. In addition, DDoS attacks and flash crowds share similar behaviours, and we have to differentiate them effectively, otherwise, we may raise a false alarm. In fact, it is a big challenge for defenders to discriminate DDoS flooding attacks from flash events and the consequences are serious if the detection system cannot discriminate between them. On one hand, the attack source can impersonate the traffic features of flash crowds to malfunction the
detectors. On the other hand, the detectors may treat the legitimate flash crowds as DDoS attacks raising false positives.

With different DDoS attack methods and strategies, creating universal DDoS detection is a huge challenge. Current DDoS attacks are carried out by attack tools, worms and botnets using different packet-transmission rates and packet forms to beat defence systems. DDoS attack sources have the ability to generate various packet-transmission forms such as the constant rate attack, increasing rate attack, flash-crowd (FC) attack, and low-rate (LDoS) attack (e.g. Reduction-of-Quality (RoQ) attacks, periodical attack, shrew attack, and pulsing attack). These different forms of attack packet transmission pose more difficult challenges for research on defending valuable online services. Moreover, various forms of attack packets can be generated and transmitted to victims based on their type of service. For example, attack packets may be malformed IP, TCP, UDP, ICMP, Application-based floods, etc. Hence, we require various detection methods for defence systems in order to cover all various attack strategies.

While improvement of information privacy, in particular increasing anonymity levels, may prevent a traffic analysis attack, DDoS detection and traceback are far from successful. This is because DDoS detection needs to perform traffic analysis to extract information of suspicious packets, such as IP headers, arrival rates, application data, etc. However, if the information of packet sources is hidden from the public by a high level of anonymity, it is really difficult for potential victims to implement measures for early detection and prevention. In order to indentify the anomalous behaviour of the packet source, we do need to reach a negotiation stage between a high level of anonymity and DDoS detection. Therefore, a suitable approach that needs improvement is a balance in the adjustment of data privacy.
2.8 Chapter Summary and Conclusion

This chapter has reviewed historical and current trends in the development of DDoS attacks. It examined in detail the three main network models of DDoS attacks: Client-Server DoS network, Typical DDoS network, and Distributed Reflector DoS network. These models could be useful for investigation and forensics purposes when attack incidents occur. We also examined the DDoS attack sources: DDoS attack tools, DDoS attack worms, and DDoS attack botnets. These attack sources also have their own attack abilities and strategies in order to impede their target victims. The aim of this chapter was to provide a background for the rest of this thesis as well as allow the reader to follow research material discussed in later chapters. The following chapters focus on the challenges and problems pointed out here, and where appropriate, are expanded on in the literature review.

This chapter has also discussed the design tradeoffs of DDoS detection strategies compared with different detection criteria. However, these detection approaches lack flexibility and cause a rise in false positive/negative rates. Discrimination between DDoS attacks and legitimate flash crowds aimed to further push research on mitigation techniques. Universal DDoS detection was proposed in order to detect the most infamous DDoS attacks through the use of various methods of packet transmission and various types of attack packets. Finally, the problems with the improvement of information privacy were briefly reviewed to explore optimisation for DDoS detection.
CHAPTER 3

Discrimination of DDoS Attack and Flash Crowd

In this chapter, we aim to differentiate DDoS attack flows from flash crowds. We are motivated by the fact that the feature of traffic volume in flash crowds is different from DDoS attacks. Therefore, the flow similarity among flash crowds is much stronger than that among DDoS attack flows. We employ abstract distance metrics, the Jeffrey distance, the Sibson distance, and the Hellinger distance to measure the similarity among flows to achieve our goal. We compared the three metrics and found that the Sibson distance was the most suitable for our purposes. We applied our algorithm to real datasets and the results indicate that the proposed algorithm can differentiate the DDoS attack with high accuracy.

3.1 Introduction

It is a difficult challenge to identify DDoS attacks when hackers mimic the normal Internet traffic pattern or hide attack flows in legitimate traffic. Because of the vulnerability of the Internet, it is easy for hackers to spoof source IP addresses of attack packets [49], and verify the pattern of attack flows [9, 50], etc. In general, DDoS detection methods include activity profiling [15, 51], sequential change-point detection [52-55], wavelet analysis [56], and chi-square/entropy detector [51, 57]. All these methods are based on the features or fingerprints of specific DDoS attacks. Unfortunately, it is very easy for hackers to mimic these features to fool user detection methods. For example, because of the open architecture of the Internet, hackers can spoof the source IP addresses of attack packets according to real Internet IP address
distribution and use this against our source address distribution based detection algorithms [58, 59]; hackers can change the TTL value of attack packets according to the real hop distance between zombies and the victim respectively in order to use this against our hop-count detection methods [59, 60]. In order to fly under the radar, attackers may also mimic the behaviours of flash crowds [54, 61], or mimic a sudden increase in legitimate traffic, e.g. many fans will access an official website when an important match is happening or many people will check the CNN website when breaking news.

DDoS attacks and flash crowds share similar behaviours, and we need to differentiate these effectively to avoid raising false alarms. In fact, it is a big challenge for defenders to discriminate between DDoS flooding attacks and flash events [54, 61, 62]. There are serious consequences if we cannot do this. On one hand, attackers can mimic the traffic features of flash crowds to disable our detectors. On the other hand, our detectors may treat legitimate flash crowds as DDoS attacks.

We are motivated by the fact that DDoS attacks and flash crowds have different traffic flows features. The DDoS attack flows are generated aggressively and simultaneously by the pre-built programs. This is because the intention of DDoS attacks is to perform the dysfunction of the available service at the victim, thus attack sources must synchronously generate the attacks. However, legitimate flash-crowd flows hesitantly increase in arrival packets observed at the server’s side. This is because flash crowds come from a rise in demand events of human interests such as breaking news, new promotions, campaigns, or sports events. Therefore, self-similarity among flash crowds is much higher than among DDoS attack flows.
In this chapter, we employ three abstract distance metrics, the Jeffrey distance, the Sibson distance, and the Hellinger distance [63] to measure the similarity among network flows. A flow is defined as the packets which are passing a router with packets sharing the same destination address. When a DDoS alarm is raised, we start to sample the suspicious flows, and measure the similarity among the flows using the previously mentioned metrics. If the distance among the flows is sufficiently small, in other words, they are similar enough, we claim them as flash crowds. Otherwise, it is a DDoS attack flow.

The major contributions of this chapter are as follows:

- We present three distance measure metrics, and found that the Sibson metric is the best for our purpose of discrimination.
- The proposed strategy is scalable and practical. The cooperating routers can be any routers on the Internet, rather than with an ISP network or a community network. We can perform our detection with only two cooperative routers on the Internet, which is much easier to achieve. The attack packets may be discarded well before they reach the victim according to the proposed methodology.
- The proposed method is independent of any specific DDoS flooding attack tools. Therefore, it can actively detect any forthcoming new attack fashions.
- The proposed algorithm is tested on real datasets, and we can differentiate DDoS flooding attacks from flash crowds with a high expectation of accuracy.

The rest of this chapter is organised as follows. Section 3.2 presents the background of flash crowds and the research that is related to this chapter. In Section 3.3, we
define the problem and specify our goal. Section 3.4 then explains the three metrics for distance measurement and the design of the discrimination algorithm. The performance analysis of the three metrics is conducted in Section 3.5, as well as the real dataset experiments for the proposed algorithm. Finally, Section 3.6 concludes this chapter.

3.2 Background and Related Work

3.2.1 Background

A flash crowd (FC) is a phenomenon that occurs when a service catches the attention of a large number of network users. In a normal situation of web services, the service requests from legitimate users do not harm the server or service. However, a busy server could suffer a FC event which is observed as a sudden high demand in service requests from network users. The surging requests dramatically raise the use of computational and memory resources as well as causing traffic congestion at the server’s side.

Flash crowds create suspicious flows similar to DDoS attacks. With a significant rise in network traffic, an FC event could overwhelm a server and create a similar DoS condition which results in either a delay of response or a complete crash. The consequences are very serious if we cannot discriminate between flash crowds and DDoS attacks. On the one hand, attackers can mimic the traffic features of flash crowds to confuse and disable a DDoS detection system. On the other hand, the detection system may treat the legitimate flash crowds as DDoS attacks which accelerate the DoS condition to the server. However, flash crowds have a number of different features from DDoS attacks.
The intention of a request generation between a flash crowd and DDoS attack is different. Flash crowds, which are from legitimate network access and request, intend to access and derive a service from a server. For example, in a flash crowd event, many sport fans access the official website when an important sports match happening or many people check CNN website when breaking news occur [64]. Therefore, the server has to perform its function by maintaining availability and handling its legitimate requests during flash crowd events. In contrast, DDoS attacks create illegitimate packets with the intention of interrupting and disrupting the availability of the server. For example, the Storm worm compromised 85,000 machines and created DDoS attacks by sending 3 billion spam emails a day against a number of anti-spam websites [35]. This means the server has to handle a large number of illegitimate requests as well as dispose of a large number of malformed requests.

The feature of traffic volume in flash crowds is different from DDoS attacks. In flash crowd events, the traffic flow increases gradually to reach its peak and then decreases gradually to its normal traffic state. These traffic behaviours are dependent on human interests. For example, the spreading of news raises people’s attention to a particular service during a period of time. In contrast, DDoS attacks generate attack traffic aggressively to reach its traffic peak, then decreases sharply to its normal traffic state, which is controlled by an attacker. For example, a large number of compromised zombies send large traffic volumes to the victim server simultaneously and then stop this attack to hide from a traceback and possible investigation.
The distribution of source IP address from flash crowds is also different from DDoS attacks. In flash crowd events, the source IP addresses of users are similar to each other. If those source IP addresses are categorised from an IP subnet-mask, a small number of subnets may contain of many source IP addresses. This is because the users of a server have a low geographical distribution, therefore, the distribution of source IP addresses is also low. In contrast, the source IP addresses from DDoS attacks have a high distribution. This is caused by spreading attack sources which could be anywhere around the world. Thus, the subnets of these attack sources consist of few source IP addresses.

3.2.2 Related Work

Previous research [62] has attempted to use three dimensions to discriminate flash crowds from DDoS attacks: traffic patterns, client characteristics and file reference characteristics. Unfortunately, this counter attack method cannot keep up with the ever changing methods of attack, as attack patterns also change from time to time, and an attacker may mimic network traffic patterns of flash crowds, causing the detector to be quickly disabled. The entropy detector mentioned [51] can raise an alarm for crowd access, however, it cannot discriminate DDoS attacks from a surge in legitimate accesses, e.g. flash crowds. Reference [54] tried to separate flash crowds from DDoS flows using the change-point detection method, but this method can be easily cheated, e.g. zombies can increase the number of attack packets very slowly, which will almost surely disable the change point detectors.

Some research has been done on trying to solve the similarity problem using stochastic methods in the frequency domain [65, 66]. Cheng et al. [65] mapped DDoS attacks from the time domain to frequency domain, and then transformed it to power
spectral density to identify the DDoS attacks. Spectral analysis [67] employed a
digital signal processing method to expose the hidden shrew DDoS attacking packets.
Reference [66] used data mining technology to find DDoS attack information,
however this is costly in terms of computing and delay. Our previous work [68]
started to explore similarity methodology, and the effectiveness of this proposed
method has been confirmed. Reference [69] used the Hellinger distance to detect
VoIP floods in peer-to-peer networks.

3.3 Problem Statement

We consider a very simple network diagram shown as Figure 7, which could be any
part of the Internet under control or cooperation of defenders. There are three routers,
$R_1, R_2$ and $R_3$, and two traffic flows $f_p$ and $f_q$, which go through router $R_2$ and $R_3$
respectively, with the flows merging at router $R_1$. The dash lines in the diagram mean
the routers may not have connected with each other immediately. In other words, the
routers were probably separate and some distance apart.

![Diagram of network](image)

Figure 7 A sample network with two traffic flows

Let $p(x)$ and $q(x)$ represent the flow probability distribution of flow $f_p$ and $f_q$,
respectively, and $\mathcal{X}$ be the finite sample space for the flows. Moreover, $p(x)$ and
In this chapter, our goal is to measure the similarity among the flows, for example \( f_p \) and \( f_q \) in Figure 7, and to differentiate DDoS attack flows from flash crowds.

### 3.4 Design of Discrimination Algorithm

#### 3.4.1 Metrics for Distance Measures

We discuss three metrics for distance measurement of network traffic flows based on the literature in this section. There are two categories in this kind of measurement: a) measurement based on information theory, and b) measure of affinity [63]. For category a), the original measurement is called the Kullback-Leibler distance [70]. For the two given flows with probability distributions \( p(x) \) and \( q(x) \), the Kullback-Leibler distance is defined as follows:

\[
D(p, q) = \sum_{x \in \chi} p(x) \cdot \log \frac{p(x)}{q(x)}
\]

Where \( \chi \) is the sample space of \( x \). It is obvious that \( D(p, q) \neq D(q, p) \), if \( p(x) \neq q(x) \). As a result, the previous equation cannot be a measure. The Jeffrey distance fixes this asymmetry using a combination of the Kullback-Leibler distance, which is defined as follows:

\[
D_J(p, q) = \frac{1}{2} [D(p, q) + D(q, p)]
\]

A further measurement for this category is the Sibson distance detailed as follows.
\[ D_{S}(p,q) = \frac{1}{2}[D(p,\frac{1}{2}(p+q)) + D(q,\frac{1}{2}(p+q))] \]  

The category b) originally came from Bhattacharyya’s measure of affinity, 
\[ \rho = \sum_{x \in X} \sqrt{p(x) \cdot q(x)} \]. The major metric used for this category is the Hellinger distance, which is defined as follows:

\[ D_{H}(p,q) = \left[ \sum_{x \in X} (\sqrt{p(x)} - \sqrt{q(x)})^2 \right]^{\frac{1}{2}} \]

It is necessary that we choose the most suitable metrics for specific purposes, e.g. measuring the similarity among network flows to discriminate DDoS attack flows.

### 3.4.2 Discrimination Algorithm

In this section, we detail the design of the discrimination algorithms. When there is a surge of network flows, we are unsure whether it is a DDoS attack or flash crowd, therefore, we name the surge flow as suspicious flows at the moment, with the cooperating routers activating the discrimination algorithm to take the decision further.

Once the discrimination process is activated, the cooperating routers start to sample the suspicious flows for a sufficient time slot \( t \), and the sampling is repeated until there are sufficient samples to make decision. The cooperative routers, e.g. router \( R_2 \) and \( R_3 \) in Figure 7, will exchange data when the sampling process is complete. The routers can then independently calculate the similarity of the flows using any one of the previous mentioned metrics (we use the Sibson distance in this chapter). If the distance is smaller than a given threshold, then the flows are flash crowds, otherwise, they are DDoS attack flows. The discrimination algorithm is detailed as in Figure 8.
**Procedure:**

01: Identify the suspicious flow, $f_i$, on a router $i$ ($i>1$), and initialise sample slot $t$, sample size $n$, and the discrimination threshold $\delta$.

02: Take samples on flow $f_i$ until the sample size $\geq n$, therefore, we obtain samples of number of packets as $x_1^i, x_2^i, \ldots, x_n^i$.

03: Router $i$ obtains its probability distribution of the flow as $p(x^i) = x_k^i \cdot \left( \sum_{k=1}^{n} x_k^i \right)^{-1}$, noted as $p(x)$.

04: Router $j$ obtains its probability distribution of the flow as $q(x) = x_k^j \cdot \left( \sum_{k=1}^{n} x_k^j \right)^{-1}$, noted as $q(x)$.

05: Exchange $p(x)$ and $q(x)$ between router $i$ and $j$.

06: The distance between $p(x)$ and $q(x)$ is calculated at router $i$ and $j$ independently using the Sibson distance metric, and noted as $D_s(p, q)$: ! can change to the nominated distance metric.

07: If $D_s(p, q) \leq \delta$, it is a DDoS attack and discard the related packets; otherwise forward the packets to the destination.

08: Go to step 02

---

**3.5 Performance Analysis on Metrics**

**3.5.1 Metric Performance Analysis**

In order to find out which metric is the most suitable one for flow similarity measurement of DDoS attacks, we carefully conducted a number of simulations. In general, people believe that Internet traffic obeys the Normal distribution pattern or the Poisson distribution pattern. Moreover, any distribution can also be represented by the combination of a series of normal distributions with different parameters.

Therefore, we examine the attributes of the three metrics using Normal distribution and Poisson distribution, respectively. There are two critical attributes that we use to compare the metrics: accuracy and sensitivity.
We arrange two flows with Normal distribution, \( \mu = 10, \sigma = 1 \), with the three distance metrics applied to these two flows to measure the information distance. The simulation is conducted for 100 times, and the results are shown in Figure 9. We also performed the same simulation on two Poisson distribution flows with \( \lambda = 10 \), with the results shown in Figure 10.

For two flows to share the same distribution and parameter(s), the distance between them is supposed to be zero in terms of statistics. From Figure 9 and Figure 10, we discovered that the Sibson’s information radius is the better metric in terms of accuracy.

In order to examine the metrics’ sensitivity to traffic flow variations, two more simulations have been performed. We first investigated the metrics’ sensitivity against standard variations of Normal distribution flows with \( \mu = 10 \) and \( \sigma \) varying from 0.1 to 3.0, or namely 1% to 30% variation from the mean. The results are shown in Figure 11.

For the Poisson flows, we examined the metric sensitivity against the arrival rate, which varied from 5 to 12. The results are shown in Figure 12.

Based on Figure 11 and Figure 12, we found that the Sibson’s information radius was the least sensitive metric among the three metrics. The simulations demonstrated that it was quite stable for the change of parameters in both the standard variation of Normal flows and the arrival rate of Poisson flows.
Figure 9 The measurements of two normal flows ($\mu = 10, \sigma = 1$)

Figure 10 The measurements of two Poisson flows ($\lambda = 10$)
Figure 11 The metric sensitivity of normal flows ($\mu = 10$) against standard deviation

Figure 12 The metric sensitivity of Poisson flows against arrival rate
3.5.2 Performance Evaluation of the Discrimination

In this section, we examine the performance of the proposed discrimination algorithm against the real datasets. We use the NLANR PMA Auckland-VIII dataset [71] as the flash crowd, and the MIT LLS DDOS 1.0 intrusion dataset [72] as the DDoS attack dataset. For each dataset, we counted the number of packets, which were addressed to the server (for flash crowds) or the victim (for DDoS attacks), with the sample interval as 100 ms, and the size of samples begin 200.

We processed the flows with the three metrics; the Hellinger distance, the Jeffrey distance, and the Sibson distance respectively. The results are shown in Figure 13, Figure 14 and Figure 15 respectively. Table 3 details the results of the similarity measurement by the three distance metrics based on Figure 13, Figure 14 and Figure 15 respectively. In our experiment, we concluded two preliminary findings:

- The Hellinger distance, the Jeffrey distance, and the Sibson distance can discriminate DDoS attack flows from flash crowds with 85% accuracy with the thresholds (δ) of 0.025, 0.0015 and 0.00025 respectively.
- The three distance metrics have the same ability in similarity measurement but are different in their threshold values in order to maximise the accuracy of the discrimination system.

Table 3 Results of similarity measurement

<table>
<thead>
<tr>
<th>Distance Metric</th>
<th>Threshold (δ)</th>
<th>False Positive</th>
<th>False Negative</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hellinger</td>
<td>0.025</td>
<td>10%</td>
<td>20%</td>
<td>85%</td>
</tr>
<tr>
<td>Jeffreys</td>
<td>0.0015</td>
<td>10%</td>
<td>20%</td>
<td>85%</td>
</tr>
<tr>
<td>Sibson</td>
<td>0.00025</td>
<td>10%</td>
<td>20%</td>
<td>85%</td>
</tr>
</tbody>
</table>
Figure 13 Similarity measure with the Hellinger distance

Figure 14 Similarity measure with the Jeffrey distance
3.6 Chapter Summary and Conclusion

In this chapter, we proposed a discrimination algorithm to differentiate DDoS attack flows from flash crowds by employing information distance to fulfil the task. We presented three metrics for information distance measures; the Jeffrey distance, the Hellinger distance, and the Sibson distance. Our simulations indicate that the Sibson distance is the least sensitive but the most stable metric among the previously mentioned metrics. Moreover, we maximised the accuracy of our discrimination system by adjusting the threshold values of the three distance metrics. The results of adjustment show these three distance metrics have the same ability in similarity measurement which can identify DDoS attacks from flash crowds with an accuracy of 85% in the real dataset experiments.
CHAPTER 4

DDoS Detection: A Predictable Behaviour of Attack Sources

In this chapter, we propose behaviour-based detection algorithms that can discriminate a Distributed Denial of Service (DDoS) attack traffic from legitimate traffic regardless of the various types of attack packets and methods. Current DDoS attacks are carried out by attack tools, worms and botnets using different packet-transmission rates and packet forms to beat defence systems. These various attack strategies lead to defence systems requiring different detection methods in order to identify the attacks. Moreover, DDoS attackers can craft traffic like flash crowd events and fly under the radar through the victim. We noticed DDoS attacks have repeatable patterns that are different from legitimate flash crowd traffic. In this chapter, we propose a comparable detection method based on the Pearson’s correlation coefficient. Our methods can extract repeatable features from the packet arrivals in DDoS traffic but not in flash crowd traffic. The extensive simulations were tested for the optimisation of detection methods. We then performed experiments with several datasets and our results confirmed that the proposed methods can differentiate DDoS attacks from legitimate traffic.

4.1 Introduction

Current DDoS attacks remain a significant threat to IT security on the Internet. Attacks can be carried out by attack tools [72], worms [73], and botnets [74] with many variants of attack packet transmission such as TCP/SYN, UDP and HTTP
request floods [75]. These sources of DDoS attack are powerful and can overwhelm any online host and server. Moreover, one of the biggest challenges for DDoS attack detection is a *flash-crowd attack*. A flash-crowd attack [75] occurs when there is a high volume of illegitimate packets from attack sources. The attack traffic is viewed in the same way as traffic with a high volume from legitimate users called *flash crowd*. Attack sources pretend to be real users and pump a large volume of request packets that flood the targeted victim. In this case, the defence/detection system may be beaten and the server has difficulty surviving the attack which causes it to crush or downgrade the service.

Statistical-based defence systems [51, 59, 76, 77] are weak from crafted information when a DDoS attack occurs. This approach relies on header information from IP packets such as IP address, time-to-live (TTL), and protocol type (port number), etc. The detection can discriminate “normal” traffic from “abnormal” traffic which is more likely to be an attack. However, some botnets, e.g. Mydoom [73] can bypass detection approaches through the victim. This is because approaches consider the Transport layer and/or Network layer. Therefore, botnets which generate similar legitimate HTTP packets can avoid detection. Even though the attacking HTTP traffic is aggregated, they still look like a flash crowd.

Heuristic-based defence systems [42, 64, 75, 78] against DDoS attack confront the problem of threshold adjustment. These approached may need to calculate its own threshold to judge the current observation traffic. The similarity, distance, classification, clustering and/or prediction analysis may be applied this research area. In contrast to the statistic-based approach, the heuristic-based approach does not need to learn/define the normal situation before making a comparison to an anomalous situation. The drawback of the heuristic detection approach is its inability to consider
legitimate traffic mixed with attacking traffic. Hence, packets from legitimate users may be blocked or eliminated during attack incidents. In addition, the threshold of this approach needs to be optimised when deployed to a DDoS defence system.

In this chapter, we propose a solution to detect the pattern behaviour of traffic sources by observing packet arrivals. Since attack sources have been programmed and work according to their attack functions, pattern detection based on their behaviours is also possible. The worms work as an automatic program which can be differentiated from human users. The botnets and DDoS attack tools work as a semi-automatic program after an attacker issues the attack command based on C&C fashion. Hence, these attack sources could repeatedly generate attack packets with different transmission abilities. These anomaly behaviours may be predictable and explainable in pattern styles. In contrast, the arrival rate based on human users, including a proxy, server seems to constitute the nonpatternable (random) cases.

Our proposed technique derives from Pearson’s correlation coefficient which is an effective method to discriminate packets among DDoS attack sources and real users including proxies. We propose two methods using the correlation to measure the data of packet arrival from an individual suspicious source with the result being either a dependent or independent relationship. The predictable (patternable) features seem to be stronger in dependent relationship of the data. The unpredictable (nonpatternable) data tended to be in the independent relationship. Since we can measure the degree of pattern behaviour, we can push the right actions to the right packets. The packets from the attack sources must be eliminated, but the user packets must get through the server.

The contributions of this chapter are listed as follows:
• **Reliability:** Our detection methods caused low false positive and false negative in the results. By using the measurement of a statistical relationship between the two sets of flow data, we can expect the result to be highly accurate.

• **Feasibility:** Our detection methods may be implemented in real-world cases based on current Internet technology. With the light calculation and low complexity, it is possible the proposed methods could be implemented in various kinds of network equipment such as hubs/switches, firewalls, routers and IDS.

• **Real-time implementation:** Our detection methods are able to detect DDoS attacks in a short period of time. Fast detection would benefit a defence system because the action response could be performed as soon as the suspicious flash-crowd traffic arrives at the server.

• **Flexibility:** Our detection methods may be able to detect any form of attack packets such as malformed IP, TCP, UDP, ICMP, and Application-based floods. Our detection methods also work well with a periodic attack with low traffic volume.

The rest of this chapter is organised as follows. Section 4.2 reviews the background and related works of our research. Section 4.3 states the problem and defines the methods to solve the problem. Each method will be discussed in detail with the adjustment of thresholds and variables in Section 4.4. Section 4.5 uses the adjusted threshold and variables to experiment with the publicly available datasets from the real traces. In the final section, we provide the summary and discuss about the direction of our research in the future.
4.2 Background and Related Work

4.2.1 Background

Our proposed approach to discriminate DDoS attack traffic from user traffic is to observe the packet transmission rate. An individual host may require access to a service from a server by sending a request. The request packets can be, for example, TCP/SYN, or HTTP requests, etc. Hence, the request packet transmission can be observed using the degree of automation, as we know attack sources work following the instructions from the programmer and have a very high degree of automation to work through after instructions are issued. When the attack sources perform a DDoS attack on the victim, their transmission rate appears to be predictable and itself becomes a pattern in a short period of time. However, Internet users have a limited time for the response from the outcome after his/her requests. For example, after a webpage has been shown, the user may take time to skim and respond, for example, clicking on a link. In other words, human users as well as proxies unpredictably create request packets at any period of time. Hence, we can test the pattern of packet transmission by using some mathematical models or statistical analysis.

As we know, attack rates depend on the characteristics of packet transmission. From the victim-end perspective, the attack packets received can be observed as an arrival rate. This attack behaviour can be divided into two main types. There are:

a) **Predictable rate:** The attack sources send out the attack packets in a predictable way to the victim. For instance, if we have enough data from a packet arrival at the time interval, we would know what is going to occur at the next time interval. This is important behaviour from an attack source, which is an automatic program, because the program follows the instructions from the (malicious) programmer. For example, the botnet program usually repeats
packet transmission until other commands are issued. There are various arrival rates (attack rates) and they can be classified as follows:

- **Constant rate** [14, 77] can be considered a stable attack rate. The attack agent (botnet) may use a constant attack rate that may be considered from the available bandwidth, the performance of a computer, and so forth. With a low bandwidth rate, the attack can fly under the radar and get through the defence system. Therefore, this attack can disturb and/or reduce the quality of services until a denial of services occurs that depends on the aggregate rate at the victim site. In cases of DDoS attack, the attack agents may continue sending the attack packets to the victim with maximum available bandwidth and full ability for transmission which may destroy the victim’s service. When a large number of agents flood a huge number of attack packets simultaneously, the vulnerable victim will be overwhelmed and unable to serve legitimate client requests. In a worst case scenario, the victim’s servers can completely crash.

- **Increasing rate** [14, 77] can be considered a linear or an exponential attack rate and is also known as an abrupt rate attack. The attack agent may increase its packet transmission rate gradually or dramatically. As a result, the victim’s resources are either slowly or rapidly exhausted. A slowly increasing attack rate can delay sensory detection of an attack. The attack agent, however, may increase the attack rate to maximum or decrease its attack rate at a later stage.

- **Periodical rate** [67, 77, 79] generates a predictable attack rate. The attack agents may not continue the same attack rates, but may repeat transmission behaviour of attack packets as a regular pattern. A periodical rate attack is
also defined in a Pulsing DoS attack which considers period of the attack (T), length of the peak (L), and magnitude of the peak (R) [8, 9].

b) **Unpredictable rate:** The variable rate attack (or fluctuating rate attack) [14] is varying the transmission rate of attack packets to avoid detection and response. To generate an unpredictable attack rate, the attack agents may randomise the transmission rate and the attack delay time for the attack packets. The attack could be generated in a continuous and/or discontinuous traffic style. The detection system may allow this type of attack to pass through victims because it appears as flash-crowd traffic, which is in high demand by legitimate Internet users.

### 4.2.2 Related Work

In order to create a DDoS defence system, the DDoS detection system plays an important role to secure the availability of services. With various methods for detecting DDoS attacks, there are several survey papers [39, 49, 80] that classify the existing methods. In general, we can classify the detection mechanisms into statistical-based and heuristics-based methods based on detection algorithms.

A statistical-based detection system (SBDS) determines normal traffic/packet data and then generalises the scope of normal. The traffic/packets that fall outside this scope are judged as anomalous (or attack). The process of an SBDS is to learn and analyse patterns of continuous network traffic. To improve accuracy, SBDS needs to learn traffic with constant patterns as much as the SBDS can be active on the network. Network traffic/packet information is processed with complex statistical algorithms. It differentiates anomalous traffic/packets from normal patterns of established network traffic. All traffic/packets are measured by an anomaly score for the specific event and...
if the score is higher than a defined threshold, the detection system will give a further action to the anomalous traffic/packets.

A heuristic-based detection system (HBDS) employs algorithmic logic from statistical analysis of the network traffic on which to base their threshold decisions. HBDS requires fine tuning to adapt to network traffic and minimise the false positives/negatives. Because heuristics are fallible, it is important to understand their limitations. Their intention is to be of assistance in order to make quick estimates and preliminary process designs.

While the key to any SBDS is its ability to learn and distinguish normal from anomalous network activity, the HBDS relies on optimisation of its threshold decision. DDoS detection with SBDS relies on header information from IP packets such as IP address, time-to-live (TTL), and protocol type (port number). The detection can discriminate normal traffic, from abnormal traffic which is more likely to be an attack. However, some botnets, e.g. Mydoom [73] can bypass detection approaches through the victim. This is because the approaches consider the Transport layer and/or Network layer. Therefore, the botnets which generate similar legitimate HTTP packets can avoid detection, and even though the attacking HTTP traffic is aggregated, they still look like a flash crowd.

HBDS against DDoS attack relies on an adjustable threshold. Each approach may need to calculate its own threshold to judge the current observing traffic. The drawback of heuristic detection approaches is their inability to consider legitimate traffic that is mixed with attacking traffic. Hence, packets from legitimate users may be blocked or eliminated during attacks. In addition, HBDS also consumes computational resources such as CPU and memory. This is an important consideration when planning to deploy HBDS.
Source-based packet filtering [59] defines the defence scheme against various source IP address spoofing. The novel method is based on source IP addresses and TTL, and follows some statistical patterns that compare non-attacking and attacking periods, then divides the source IP addresses into $n \ (1 \leq n \leq 32)$ segments. This statistic-based detection works independently at the potential victim’s side, and there is no requirement for cooperation among routers in the defence scheme. This could benefit on storage space and speed of information retrieval for the potential victims.

Xie and Yu [76, 77] created DDoS detection architecture for monitoring Web flash-crowd traffic in order to reveal dynamic shifts in normal bursts of traffic, which might signal the onset of application-layer DDoS (App-DDoS) attacks during a flash crowd event. The proposed method is based on principal component analysis (PCA), independent component analysis (ICA), and hidden semi-Markov model (HsMM). The Access Matrix (AM) is designed to capture spatial-temporal patterns of a normal flash crowd. The entropy of document popularity fitting to the model is used to detect potential App-DDoS attacks. The experiment could differentiate App-DDoS attack modes (i.e., constant rate attacks, increasing rate attacks and stochastic pulsing attack) from a flash crowd event. When the detection threshold of entropy is set as $\mu \pm 3\sigma$, the detection rate (DR) is 91.08\% and FPR is 1.78\%. The model was also tested based on user browsing behaviour. When the detection threshold of entropy is set as $\mu \pm 3\sigma$, the DR is 97.90\% and the FPR is 1.80\%. However, the AM needs to learn the pattern of “normal” legitimated traffic and the threshold of entropy needs to be adjusted to maximise DR and minimise FPR.

An anomaly detection system [42] deploys Support Vector Machine (SVM) in order to create a learnable algorithm. The detection is implemented for classification and deploys the Dynamically Growing Self-Organising Tree (DGSOT) algorithm for
clustering analysis. The approach was compared with the Rocchio Bundling technique and random selection in terms of accuracy loss and gain in training time. While the SVM + DGSOT accuracy rate is very low (69.8%), the training time is long (13.18 hours), the false negative rate (FNR) is very high (37.8%), as is the false positive rate (FPR) at 29.8%.

Human-vs.-bot differentiation [75] by human behaviour modelling was proposed with an adjustable threshold. The approach derives the three aspects of human behaviour: 1) request dynamics, 2) request semantics and 3) ability to process visual cues. The evaluation processes are based on a series of web traffic logs, interlaced with synthetically generated attacks. This heuristic approach could discover the flash-crowd attacks hidden in the test traffics with high accuracy (around 95-99%). The adjustable threshold of 0.05 gives a low FNR (0.00%) and low FPR (1.94%).

Chonka, Singh and Zhou [78] deploy the theory of network self-similarity to differentiate DDoS flooding attack from legitimate self-similar traffic in the network. A neural network detector was developed for training by a DDoS prediction algorithm. When the threshold of sensitivity has a range of 88% to 94%, FPR is also very low with a range of 0.05% to 0.45%.

In the previous chapter, we created a heuristic-based DDoS discriminator from flash crowd traffic. We nominated distance metrics (the Jeffrey distance, the Sibson distance, and the Hellinger distance) to measure the similarity among flows. We compared the three metrics and found that the Sibson distance is the most suitable one for our purposes as it differentiates DDoS attacks from a flash crowd with an accuracy of around 85%.
4.3 Problem Statement

We consider the situation where a server is overwhelmed by flash crowd flows and/or DDoS attacks as illustrated in Figure 16. A server connects to the Internet and provides a service to public Internet users. Legitimate users do not harm the server or the service. However, the busy server may suffer a flash crowd (FC) event which is observed as a sudden high demand in service requests from Internet users. A flash crowd could overwhelm a server and create a DoS condition which results in either a delay of response or a disappearance of service.

A DDoS attack is, however, more harmful than a flash crowd (represented as FC1, FC2 and FC3 in Figure 16). Zombie machines or bots (represented as Z1, Z2 and Z3 in Figure 16) are compromised and controlled by attackers. The (botnet) attacks can be synchronised to overwhelm the victim (represented as Server in Figure 16) during a specific period of time. The situation may worsen when a flash crowd merges with a DDoS attack as shown in Figure 17. This accelerates the DoS condition to the server. As a result, other users (represented as U1, U2 and U3 in Figure 16) are unable to access the service.

The behaviour of the bot can be detected by the victim’s server side by observing the predictable arrival rate. To minimise the cost of calculation, the server can observe the arrival rate ($\lambda_k$) from a high risk group of users. A study [74] found that in a botnet attack scenario, at most around 30% of bots were online at the same time during attack activities. This could possibly be an approximate number of IP addresses that is needed to run a check on bot behaviour. In particular, only 30% of user IP addresses that express high arrival rates can be checked in a given period of time. This chapter
covers only two methods using the correlation coefficient to check arrival rates as data.

![A sample server environment](image1)

**Figure 16** A sample server environment

![Accumulative arrival rate](image2)

**Figure 17** Accumulative arrival rate $\lambda$ (packet/time interval) from $k$ source IP address(es)

### 4.3.1 Mathematical Models

Based on data from arrival rates, we need mathematical models to identify the degree of prediction. Since we categorise data into predictable and unpredictable data, the mathematical models must be able to judge the data by using a threshold. These
possible models are the tools for self-similarity analysis such as the correlation coefficient and distance matrix. In this chapter, we use Pearson's correlation coefficient (hereafter called the correlation) [81], which is defined as:

\[
\rho_{X,Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}
\]

The correlation is used to measure dependence between two quantities (variables) \(X\) and \(Y\) with expected values \(\mu_X\) and \(\mu_Y\) and standard deviations \(\sigma_X\) and \(\sigma_Y\). Both the value of the standard deviations are finite and nonzero \((0 < \sigma_X < \infty\) and \(0 < \sigma_Y < \infty\)).

One of the impressive properties of the correlation is symmetric measurement \((\rho_{X,Y} = \rho_{Y,X})\). In other words, whichever data comes first, we can still achieve the same result as measuring.

The correlation value is between -1 and 1 \((-1 \leq \rho_{X,Y} \leq 1\)). Hence its absolute value \(|\rho_{X,Y}|\) cannot exceed 1. The absolute correlation value is 1 \((|\rho_{X,Y}| = 1)\), which is represented by the stronger relationship between two variables called linear dependence. However, the absolute value from the correlation may reach zero \((|\rho_{X,Y}| = 0)\), but this does not always mean the two variables are uncorrelated. In a special case where both are normal, the uncorrelated result is also equivalent to independence. In our research, we define the data that gives us this value of 1 \((|\rho_{X,Y}| = 1)\) as predictable data with a linear form. The absolute correlation value of 0 \((|\rho_{X,Y}| = 0)\) also defines predictable data with a symmetric form.
4.3.2 Methodology Algorithms

a) Method 1: Correlation between arrival rate and sequence number.

Inputs:
- $\{\lambda_k\}$: The sample set of arrival rate
- $\{k\}$: The sample set of sequence number // sample data
- $t$: The period of time for packet arrivals // by default $t = 0.1$ second
- MAX_K: The maximum number of sample data
- MAX_I: The maximum number of sample correlation

Output:
- $\{\rho_i\}$: The set of correlation coefficient

Procedure:
01: Let MAX_K = 20 // by default
02: Let MAX_I = 10 // by default
03: Let $k = 0$, $i = 0$
04: For each $i$ Until MAX_I
05: For each $k$ Until MAX_K
06: $\lambda_k = \text{number of arrival packets during time } t$
07: $X[k] = \lambda_k$
08: $Y[k] = k$
09: $k$ Increases 1
10: Calculate $\rho_{XY}$ // using Equation (5)
11: $\rho_i = |\rho_{XY}|$ // absolute value of correlation
12: $i$ Increases 1
13: Let $k = 0$ // reset $k$ for the next loop $i$

Figure 18 Algorithm of Method 1

As described in Figure 18, we denote $X$ as a sample set of an arrival rate ($X = \{\lambda_k\}$, where $k = 0, 1, 2, \ldots, N$) and $Y$ as a sample set of sequence numbers ($Y = \{k\}$ where $k = 0, 1, 2, \ldots, N$). For example, $X = \{\lambda_0, \lambda_1, \lambda_2, \ldots, \lambda_k\}$ and $Y = \{0, 1, 2, \ldots, k\}$. Then, we calculate the correlation value ($\rho_{XY}$) from the two variables, $X$ and $Y$. The value that we expect is between -1 and 1 ($-1 \leq \rho_{XY} \leq 1$).
b) Method 2: Correlation of self-arrival rate.

As described in Figure 19, we denote \( X \) as a sample sequence of an arrival rate \( (X = \{ \lambda(2k) \}, \text{ where } k = 0, 1, 2, \ldots, N) \) and \( Y \) as a sequence number of the time interval \( (Y = \{ \lambda(2k+1) \} \text{ where } k = 0, 1, 2, \ldots, N) \). For example, \( X = \{ \lambda_0, \lambda_2, \lambda_4, \ldots, \lambda_{(2k)} \} \) and \( Y = \{ \lambda_1, \lambda_3, \lambda_5, \ldots, \lambda_{(2k+1)} \} \). Then we calculate the correlation value \( (\rho_{XY}) \) from the two variables: \( X \) and \( Y \). The value that we expect is between -1 and 1 \((-1 \leq \rho_{XY} \leq 1)\).

<table>
<thead>
<tr>
<th>Inputs:</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ \lambda_k }: The sample set of arrival rate (even no. data)</td>
</tr>
<tr>
<td>{ \lambda_{k+1} }: The sample set of arrival rate (odd no. data)</td>
</tr>
<tr>
<td>( t ): The period of time for packet arrivals // by default ( t = 0.1 ) second</td>
</tr>
<tr>
<td>( MAX_K ): The maximum number of sample data</td>
</tr>
<tr>
<td>( MAX_I ): The maximum number of sample correlation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output:</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ \rho_i }: The set of correlation coefficient</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Procedure:</th>
</tr>
</thead>
<tbody>
<tr>
<td>01: Let ( MAX_K = 10 ) // by default</td>
</tr>
<tr>
<td>02: Let ( MAX_I = 5 ) // by default</td>
</tr>
<tr>
<td>03: Let ( k = 0, i = 0 )</td>
</tr>
<tr>
<td>04: For each ( i ) Until ( MAX_I )</td>
</tr>
<tr>
<td>05: For each ( k ) Until ( MAX_K )</td>
</tr>
<tr>
<td>06: ( \lambda_k ) = number of arrival packets during time ( t )</td>
</tr>
<tr>
<td>07: ( X[(k/2)] = \lambda_k )</td>
</tr>
<tr>
<td>08: ( \lambda_{k+1} ) = number of arrival packets during next time ( t )</td>
</tr>
<tr>
<td>09: ( Y[(k/2)+1] = \lambda_{k+1} )</td>
</tr>
<tr>
<td>10: ( k ) Increases 2</td>
</tr>
<tr>
<td>11: Calculate ( \rho_{XY} ) // using Equation (5)</td>
</tr>
<tr>
<td>12: ( \rho_i =</td>
</tr>
<tr>
<td>13: ( i ) Increases 1</td>
</tr>
<tr>
<td>14: Let ( k = 0 ) // reset ( k ) for the next loop ( i )</td>
</tr>
</tbody>
</table>

Figure 19 Algorithm of Method 2.
4.3.3 Threshold Algorithm

For both methods we calculate the correlation value and define two thresholds: upper threshold ($\tau_U$) and lower threshold ($\tau_L$). We can calculate these thresholds as follows:

\[
\tau_U = \alpha \cdot 1.0 \tag{6}
\]

\[
\tau_L = 1.0 - (\alpha \cdot 1.0) \tag{7}
\]

As shown in Figure 20, the value of the upper threshold must not exceed 1 but should be less than the lower threshold ($1 \geq \tau_U \geq \tau_L$). On the other hand, the value of the lower threshold must not be below 0, but should be greater than the upper threshold ($\tau_U \geq \tau_L \geq 0$). The confidence value ($\alpha$) is another adjustable value that we will discuss in the next section. As we stated in our goals, these thresholds will help us to identify the degree of dependency on the arrival rate data and allow us to put these into two categories:

a) **Predictable attack rate**: The data will be classified as a predictable attack rate if the correlation value is close to 0 or 1 as we discussed in the previous...
section. If the absolute correlation value is less than the lower threshold \(0 \leq |\rho_{X,Y}| \leq \tau_L\), or greater than the upper threshold \(\tau_U \leq |\rho_{X,Y}| \leq 1\), the data is judged as a predictable attack rate. However, we still need to define how close the correlation value can be for it to be considered a dependency arrival rate. This issue will be explained in more detail in the next section.

b) **Unpredictable attack rate**: The data will be classified as an unpredictable attack rate if the correlation value is not close to 0 or 1. In other words, the data is expressed as a non-attack arrival rate and is legitimate to the service of the server. If the absolute correlation value is between the lower and upper thresholds \(\tau_L < |\rho_{X,Y}| < \tau_U\), the data is judged as an unpredictable attack rate. However, we still need to define the range of the correlation value that can be judged as an independency arrival rate. This issue will be explained in more detail in the next section.

Unfortunately, only one correlation result \(\rho_{X,Y}\) is unable to determine whether the arrival data is attacking or legitimate. We need a series of correlation results to confirm this situation. Hence we define \(\{\rho_i\}\) is a set of the consecutive results of the correlation coefficient. The \(i\) variable \((i = 0, 1, 2, \ldots, N)\) could be the limited number observing the correlation value. For instance, if we want to observe the correlation for 10 values, we will have \(\{\rho_0, \rho_1, \rho_2, \ldots, \rho_9\}\). Each of the correlation values will be calculated with the upper \((\tau_U)\) and lower threshold \((\tau_L)\) to define whether the data is predictable or not. All of these results will be calculated for a number of predictability points \((P)\) and then the average predictability point \((\bar{P})\) respectively. We can then identify the traffic as an attack or a legitimate flow as shown in Figure 21. Finally, we
decided to only drop the IP traffic that expressed repeatable features, which is an indication of the predictable traffic. For more details about the above variables, we provide a discussion with value analysis in the next section.

\[
\begin{align*}
\text{Inputs:} \\
\{\rho_i\}: & \text{ The set of correlation coefficient} \\
m: & \text{ The size of sequence number // sample correlation data} \\
\tau_u: & \text{ The upper threshold // using Equation (6)} \\
\tau_l: & \text{ The lower threshold // using Equation (7)} \\
\text{Output:} \\
\text{Sample of arrival data} \{\lambda_k\} & \text{ contains either predictable attack or not.} \\
\text{Procedure:} \\
01: & \text{ Let } P = 0 \\
02: & \text{ For each } \rho_i \text{ Until } m \text{ // using Equation (8)} \\
03: & \text{ Case 1: } \left( |\rho_i| \geq \tau_u \right) \text{ OR } \left( |\rho_i| \leq \tau_l \right) \\
04: & \text{ The } i^{th} \text{ set of arrival data } \{\lambda_k\} \text{ is dependence} \\
05: & \text{ } P \text{ increases 1 // 1 point} \\
06: & \text{ Case 2: } \left( |\rho_i| < \tau_u \right) \text{ AND } \left( |\rho_i| > \tau_l \right) \\
07: & \text{ The } i^{th} \text{ set of arrival data } \{\lambda_k\} \text{ is independence} \\
08: & \text{ Calculate } \overline{P} \text{ // using Equation (9)} \\
09: & \text{ If } (\overline{P} = 1) \text{ Then } \text{// identify as an attack traffic} \\
10: & \text{ The arrival data } \{\lambda_k\} \text{ is predictable} \\
11: & \text{ Drop packets from this IP address} \\
12: & \text{ Else } \text{// identify as an legitimate traffic} \\
13: & \text{ The arrival data } \{\lambda_k\} \text{ is unpredictable} \\
14: & \text{ Pass packets from this IP address} \\
15: & \text{ End}
\end{align*}
\]

Figure 21 Algorithm of making a decision by thresholds.

\[4.4 \quad \text{System Optimisation Analysis}\]

In this section, we discuss optimising variables. As we proposed in our goals, the attack detection system must respond as quickly as possible after the attack reaches
the victim. The computational resources also need to be minimised with simplified methods. To find the optimised variables, we analysed the following:

### 4.4.1 Size of Sample

We begin from size \((k)\) of a sample set of arrival rates \(\{\lambda_k\}, \) where \(k = 0, 1, 2, \ldots, N\). The question is how much sample data should be used? If the size is very small, for example \(k = 3\), the calculation process is very quick. However, the correlation result \((\rho_{X,Y})\) may be misleading. As a result, the performance measurement gives us a high rate of false negatives/positives. On the contrary, if the size is quite large, for example \(k = 100\), the calculation process is very slow. This also means we wait for a long period of time to obtain all sample data \((\{\lambda_0, \lambda_1, \lambda_2, \ldots, \lambda_{99}\})\). For example, if each \(k\) has a time slot of 0.1 seconds. Our defence system needs at least 10 seconds to obtain the first sample correlation coefficient \((\rho_0)\). Moreover, if we observe up to 10 sample correlation data \(\{\rho_0, \rho_1, \rho_2, \ldots, \rho_9\}\), the detection will give us the result in at least 11 seconds (10 seconds for the first correlation data \((\rho_0)\), plus 1 second for the remaining data \(\{\rho_1, \rho_2, \rho_3, \ldots, \rho_9\}\)). This delay in making the decision means a weak victim may have an increased chance of crashing.

### 4.4.2 Correlation Thresholds and Confidence Value

There are two thresholds we consider in minimising the false positive/negative rate: the upper threshold \((\tau_U)\) and lower threshold \((\tau_L)\). To catch predictable attacks, we need to adjust the absolute correlation value to be greater than the upper threshold \((\tau_U \leq |\rho_{X,Y}| \leq 1)\) or less than the lower threshold \((0 \leq |\rho_{X,Y}| \leq \tau_L)\). The question arises what the best values for \(\tau_U\) and \(\tau_L\) are. If we adjust \(\tau_U\) too high and \(\tau_L\) too low, our
detection system may fail, and as a result, the defence system may allow most attack packets to get through. On the contrary, if we adjust $\tau_U$ too low and $\tau_L$ too high, we may confront the DoS condition earlier because most packets would be considered a predictable attack. Since the two thresholds are important, the adjusted values may rely on how much confidence we have. Hence, the confidence value ($\alpha$) would be calculated using these thresholds. By default, we assign the confidence value of 85% ($\alpha = 0.85$). Thus, the $\tau_U$ is 0.85 of the correlation value and $\tau_L$ is 0.15 of the correlation value.

4.4.3 Predictability point

Before the final decision on the detection system is made, the predictability point ($P$) is a variable that needs to be discussed because it would be assigned based on the result of each correlation ($\rho_i$). Because the correlation result may be close to 0 or 1 ($\rho_{X,Y} \to 0$ OR $\rho_{X,Y} \to 1$), we need to transform this result into a marking score fashion. In case of predictable data, if the correlation value is not lower than the upper threshold ($\tau_U$), or is not higher than the lower threshold ($\tau_L$), we then set $P = 1$. In case of unpredictable data, if the correlation value is between the upper threshold ($\tau_U$) and the lower threshold ($\tau_L$), we then set $P = 0$.

$$P = \begin{cases} 1, & \text{if } (\tau_U \leq |\rho_{X,Y}| \leq 1) \text{ or } (0 \leq |\rho_{X,Y}| \leq \tau_L) \\ 0, & \text{otherwise} \end{cases}$$

(8)

The final step is to judge whether or not the arrival data $\{\lambda_k\}$ is a predictable attack from the average predictability point ($\bar{P}$). The $\bar{P}$ is calculated from the total predictability point divided by the total number ($m$) of observing correlation ($\{\rho_i\}$) as below:
\[ \bar{P} = \frac{1}{m} \sum_{j=0}^{m} p_j \] 

The question is how much \( m \) should be used. If the number of consecutive correlation values is very small, for example \( m = 2 \), the calculation process is very quick. However, the \( \bar{P} \) may be misleading. As a result, the performance measurement gives us a high rate of false negatives/positives. On the contrary, if the number of consecutive correlation values is quite large, for example \( m = 20 \), the detection process is very slow. It also means we wait for a longer time to obtain all the data \( \{P_0, P_1, P_2, \ldots, P_{19}\} \), for example, if each \( k = 10 \) has a time slot of 0.1 seconds. Our defence system needs at least 1.0 second to get the first sample correlation coefficient \( (\rho_0) \). If we observe up to 20 consecutive correlation data \( \{\rho_0, \rho_1, \rho_2, \ldots, \rho_{19}\} \), we need to do calculations for 20 predictability points \( \{P_0, P_1, P_2, \ldots, P_{19}\} \). The detection then gives us the result in at least 2.9 seconds (1.0 second for the first predictability point \( (\rho_0) \) plus 1.9 seconds for the remaining data \( \{P_1, P_2, \ldots, P_{19}\} \). This delay in making the decision means a weak victim may increase their probability of crashing.

By default, we assign the total number of observing correlation to 5 \( (m = 5) \). However, we strongly advise the collection of 100% of predictability points to detect the predictability feature of the data. This means that the average predictability points must equal 1 \( (\bar{P} = 1.0) \). Otherwise, we define the data as unpredictable.

4.4.4 Generated Dataset Analysis

We may optimise all the variables from the experiment on generated datasets. The datasets illustrate the samples of scenarios that may help us to discover the suitable value for those variables. To help us achieve this we first generated two datasets:
random linear and peak curve as depicted in Figure 22(a) and Figure 24(a) respectively. Secondly we captured some part of the real datasets that expressed the screw attacks.

In the generated dataset 1, we assume in the scenario that the attack source slowly increases the attack rate, which delays the detection of an aggressive rate. Figure 22(a) depicts the straight relationship between data sequence number and arrival rate in method 1. Figure 23(a) depicts the straight relationship between the arrival rate itself in method 2.

Dataset 2 assumes the scenario that the attack source slowly increases and then decreases the attack rate creating a peak form which is able to delay the detection of an aggressive rate. Figure 24(a) depicts the symmetry-peak relationship between the data sequence number and the arrival rate in method 1. Figure 25(a) depicts the straight relationship between the arrival rate itself in method 2. For each generated dataset, we tested the two methods and compared their results. Based on the theory, our generated datasets must give us the result as predictable data. This means that each dataset may give us a close correlation value of zero or one ($\rho_{X,Y} \to 0 \ OR \rho_{X,Y} \to 1$).
Figure 22 Experiment on generated dataset 1 (exponential line) with method 1, (a) packet arrival plot, (b) accumulative correlation, and (c) correlation from different $k$.
Figure 23 Experiment on generated dataset 1 (exponential line) with method 2, (a) packet arrival plot, (b) accumulative correlation, and (c) correlation from different $k$. 
Figure 24: Experiment on generated dataset 2 (peak) with method 1, (a) packet arrival plot, (b) accumulative correlation, and (c) correlation from different $k$
Figure 25 Experiment on generated dataset 2 (peak) with method 2, (a) packet arrival plot, (b) accumulative correlation, and (c) correlation from different $k$
4.5 Optimisation of the Results

Optimisation is an important step in maximising our detection performance. We begin with the calculation of the accumulative correlation and the correlation with different \( k \) values. All generated datasets were tested to find the proper \( k \) as a default \( k \) value for all other real datasets. The proper \( k \) value links to the proper thresholds and confidence value. The results of the generated datasets are as follows:

4.5.1 Generated Dataset 1 (Linear Increasing Attack)

For the result of accumulative correlation in Figure 22(b) and Figure 23(b), both methods provide excellent results with a higher confidence value of more than 95\% (\( \alpha > 95\% \)). This means the results of the correlation \( \{\rho_1\} \) are between 0.95 and 1.00 (0.95 \( \geq \rho_1 \geq 1.00 \)), which is higher than the upper threshold (\( \tau_U \)). Hence, this scenario can apply to any size (\( k \)) of the sample data without misleading information about the correlation. However, we have to test for different sizes (\( k \)) of sample data to be sure.

For the results of correlation from different sizes (\( k \)) of the sample data, both methods had different capabilities to judge the results. In method 1, the lower size (\( k \)) sample data was selected when there was a higher degree of misleading information about correlation. With \( k = 5 \) and \( k = 10 \), only 64\% and 89\% of the sample data can be judged as predictably linear. With \( k = 20 \), we can confidentially confirm that this data is predictably linear. On the other hand, we can also use method 2 with any \( k \) of more than 5 and achieve confidence of at least 95\% with which to determine this data as predictably linear. Hence, in the case of a linear relationship, method 2 seems more closely identifies the relationship and provides more accuracy than method 1.
4.5.2 Generated Dataset 2 (Peak Curve Attack)

For the result of accumulative correlation, both methods give us very different results. Method 1 gives us results of correlation $\{\rho_i\}$ varying between 0 and 1. These results will be confusing when we try to distinguish predictable behaviour. However, method 2 gives the confidence value ($\alpha$) up to 99%, which is higher than the default confidence value. Hence, this scenario can apply method 1 to any size ($k$) of sample data without providing misleading information about correlation. To be absolutely positive about this we need to test for different sizes ($k$) of the sample data.

For the result of correlation from different sizes ($k$) of the sample data, both methods provide different avenues to judge the results. In method 1, we found that the higher the degree of misleading information about correlation, equated to the selection of a larger size ($k$) of the sample data. With $k = 10$ and $k = 20$, only 91% and 80% of sample data respectively can be determined as predictably linear. With $k = 5$, we can confirm that 97% of this data is predictably linear. On the other hand, if we use method 2 with any $k$ more than 10, and we achieve confidence of at least 95% to determine the data as predictably linear. Hence, in the case of a linear relationship, method 2 more closely identifies the relationship and provides more accuracy than method 1.

However, we do not calculate too much arrival data because this process is costly. Only 20 continuous correlation values would be enough to judge whether the arrival data is either predictable or unpredictable. If the average predictability point ($\bar{P}$) of one part of the arrival data expresses predictable behaviour, we can then proceed with further actions such as dropping packets.
4.6 Evaluations

We tested our methods with the generated datasets and analysed how to optimise all variables in our discrimination detection system. However, we were unable to use all details of the test due to the number of results. Based on the real datasets, we tested these with optimised variables using both methods. The following examples are provided with a description:

Table 4 List of initial variables.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Method 1</th>
<th>Method 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time interval</td>
<td>0.1 second</td>
<td>0.1 second</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>85%</td>
<td>85%</td>
</tr>
<tr>
<td>$\tau_u / \tau_L$</td>
<td>0.85 / 0.15</td>
<td>0.85 / 0.15</td>
</tr>
<tr>
<td>$k$</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>$m$</td>
<td>10</td>
<td>5</td>
</tr>
</tbody>
</table>

4.6.1 Sample Dataset 1 (WC55)

WC55 is an example of clients from a website of the World Cup 98 [82]. Its arrival rates are similar to screw attacks as depicted in Figure 26(a). However, our belief is that this may not be a screw attack but an automatic program. Perhaps, this is an effect of using Java script to update data from the website.

Now if we consider the results of correlation from both methods, accumulative correlation provides us with very different meanings. Method 1 detects the traffic flaw as an attack with a confidence value of 94% (as shown in Figure 26(b)), but the other method does not (as shown in Figure 27(b)). In this case, we must rely on the method
that can detect the relationship of dependence, and therefore we will not consider correlation with \( k = 10 \) from method 2.

As we have stated, method 1 is more reliable in this scenario. The accumulative correlation is more stable and less than 0.05 after \( k > 20 \). Hence, we consider \( k = 20 \). As a result, the maximum average predictability point \((\bar{P})\) is 0.80, if \( \alpha = 95\% \).

Because we set the confidence value too high, this scenario did not detect whether it was a screw attack.

The system administration may ignore this kind of low attack rate which may not harm to the service system. Perhaps this was a downloaded program or Java scripts that regularly download HTTP objects from the website. However, if we consider it as an automatic program, we could reset \( \alpha = 85\% \) and then the system could detect an attack after 30 time intervals (equal to 3.0 seconds).

### 4.6.2 Sample Dataset 2 (MIT17060)

MIT17060 is sample traffic of the client from the MStream attack project [72]. Its arrival rates are transmitted in random mode and are hard to detect as shown in Figure 28(a). As this is a high arrival rate, we expected our method to detect it as soon as possible before it could harm the server.

If we consider the result of correlation from both methods as shown in Figure 28(b) and Figure 29(b) respectively, the accumulative correlation tells us something similar; method 1 and 2 detect the traffic as an attack with 85% of the confidence threshold value. As a result, the maximum average predictability point \((\bar{P})\) is 1.00, if \( \alpha = 85\% \).

In regards to the detection time, the two methods can detect the attack within 1.7 seconds and 2.4 seconds respectively.
4.6.3 Sample Dataset 3 (WC1387)

WC1387 is an example of the clients from the World Cup 98 website [82]. Its arrival rates are transmitted like a flash crowd as depicted in Figure 30(a). As this is a high arrival rate, we expected our method to detect this flow as legitimate flash crowd traffic with a low degree of harm to the server.

Now let us consider the results of correlation from both methods. The result of accumulative correlation as shown in Figure 30(b) and Figure 31(b) tells us a similar meaning; method 1 and 2 could not detect a strong relationship for the traffic flow with 85% of the confidence threshold value. As a result, the maximum average predictability point ($\overline{P}$) is only 0.10 and 0.60 from both methods respectively, if $\alpha = 85\%$. This is because the traffic is clean and therefore, all attack request data passes through the server.

Table 5 Comparison of detection results.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Detection result (Best Detection Time)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Method 1</td>
</tr>
<tr>
<td>WC55</td>
<td>Attack (3.0 seconds)</td>
</tr>
<tr>
<td>MIT17060</td>
<td>Attack (1.7 seconds)</td>
</tr>
<tr>
<td>WC1387</td>
<td>Non-Attack</td>
</tr>
</tbody>
</table>
Figure 26 Experiment on sample dataset 1 (WC55) with method 1, (a) packet arrival plot, and (b) correlation from $k = 20$. 

(b) correlation from $k = 20$. 


Figure 27 Experiment on generated dataset 1 (WC55) with method 2, (a) packet arrival plot, and (b) correlation with $k = 10$. 

and (b) correlation with $k = 10$. 
Figure 28 Experiment on sample dataset 2 (MIT17060) with method 1, (a) packet arrival plot, and (b) correlation from $k = 20$. 
Figure 29 Experiment on sample dataset 2 (MIT17060) with method 2, (a) packet arrival plot, and (b) correlation with $k = 10$. 
Figure 30 Experiment on sample dataset 3 (WC1387) with method 1, (a) packet arrival plot, and (b) correlation from $k = 20$. 
Figure 31 Experiment on sample dataset 3 (WC1387) with method 2, (a) packet arrival plot, and (b) correlation with k=10.

4.7 Chapter Summary and Conclusion

As we previously mentioned, DDoS attack sources have a form of pattern behaviour for packet transmission. The predictability measurement of known patterns is a very effective approach to detect anomalous behaviour of DDoS attacks. We have proposed two methods using the Pearson’s correlation coefficient to detect the known
patterns. Moreover, we tested these methods with generated data and real-trace datasets from the website of the World Cup 98 and MStream attack project. We found hidden predictable behaviour from both datasets. The best results we achieved were 1.7 seconds and 2.4 seconds from the first and second method respectively. We can also differentiate flash crowd traffic from DDoS attack traffic. The detection performance so far was good enough to protect the server from crashing during a DDoS attack incident. We believe that our experiment is a significant step to providing universal DDoS detection which could be implemented on any network equipment or any Internet layer.
CHAPTER 5

Smart DDoS Detection: A Learning Algorithm with LDA

In this chapter, we propose an effective approach with a supervised learning system based on Linear Discriminant Analysis (LDA) to discriminate legitimate traffic from DDoS attack traffic. Currently, there is a wide outbreak of DDoS attacks that remain risky for the entire Internet. Different attack methods and various attack strategies are trying to challenge DDoS defence systems. Among the behaviours of attack sources, repeatable and predictable features differ from legitimate sources of traffic such as humans and Internet proxies. In addition, the DDoS defence systems lack the learning ability to fine-tune their accuracy of detection results. This chapter analyses real trace traffic from publicly available datasets with triple checks of repeatable patterns on attack sources. Pearson’s correlation coefficient and Shannon’s entropy are deployed for extracting dependency and predictability of traffic data respectively. Then Linear Discriminant Analysis (LDA) is used to train and classify legitimate and attack traffic flows. From the results of our experiment, we can confirm that the proposed discrimination system can differentiate DDoS attacks from legitimate traffic with a high rate of accuracy.

5.1 Introduction

Today, Distributed Denial of Service (DDoS) attacks are serious threats to computer hosts on the Internet. A recent report [6] has revealed the largest attack size doubled year after year, to more than 100 Gbps, which is a surprising 1000% increase in attack
size since 2005. The attacks can be carried out by a large number of compromised hosts, called zombie armies. These hosts become the attack tools associated with performing DDoS attacks and are the reason why legitimate users experience a decline or absence of their service. Moreover, mimicking DDoS attacks [78] place more pressure on the defence system to differentiate attacks as opposed to legitimate flows (flash crowd). On an individual attack source, the packet transmission is crafted in a random fashion. This method helps the attack traffic fly under the radar through the victim.

Research on DDoS detection has been able to identify DDoS attack packets and/or traffic flows. In addition to the detection of common attacks, DDoS detection uses statistical and behavioural analysis methods to identify attacks in progress. While, the key to any statistical-based detection system (SBDS) [59, 76, 77] is its ability to learn and distinguish normal from anomalous network activity, the heuristic-based detection system (HBDS) [42, 75, 78, 83] relies on optimisation of its threshold decision. Moreover, HBDS needs fine-tuning to produce stability, and improve results of anomaly detection in network traffic and minimise false positives/negatives. This configuration process needs manual and semi-automatic adjustment for its threshold. Hence, creating an autonomous DDoS detection system with a learning algorithm is still a mystery in this research area.

As we follow the assumption from the previous chapter, the action of a DDoS attack source follows the instructions programmed by an attacker. Its function is repeatable to generate and transmit DDoS attack packets to the reflector/victim. As a repeatable feature from the attack source, the repeatable behaviour is apparently different from the behaviour of a human user or an Internet proxy. Since we can catch the repeatable feature of an attack source, we can measure the degree of predictability in the
behaviour of packet transmission. Unfortunately, we are unable to detect this repeatable feature closer to the attack source from the side of the victim. Hence, we could measure their behaviours by observing the packet arrivals instead. The rate of packet arrivals may be moderately altered from the original rate of packet generation during the process of packet transmission from the source to the destination. However, we can be successful in measuring the predictability feature by using the Pearson product-moment correlation coefficient and Shannon entropy. We can also classify these features into two groups (attack and legitimate) by Linear Discriminant Function as we will explain later in the next section.

In this chapter, we propose a solution to discriminate DDoS using the supervised learning model from the pattern behaviour of traffic sources by observing packet arrivals. This proposed technique is an effective method to discriminate packets among DDoS attack sources and legitimate users including human users and proxies. The packet arrival rate is detected as measurement data to differentiate attack-source traffic from legitimate traffic. We have doubled the measurements of pattern behaviour from Pearson’s correlation and the Shannon entropy. Since we can measure the degree of pattern behaviour, we can classify the suspicious flows into either legitimate flow or attack flow by Linear Discriminant Analysis (LDA). The flows from the attack sources must be filtered out but legitimate flows must get through the server.

The contributions of this chapter are listed as follows:

- **Reliability**: Our DDoS discrimination system maximises the accuracy in detecting and minimising a false positive rate (FPR) and a false negative rate (FNR) in the results. By using the triple check from Pearson’s correlation, the
Shannon entropy and LDA, the statistical relationship of a traffic flow is measured a high rate of accuracy.

- **Feasibility**: Our DDoS discrimination system could be implemented in real-world cases based on current Internet technology. With light calculation and low complexity, the proposed methods are feasible to be implemented in many kinds of network equipment such as hub/switches, firewalls, routers, IDS and IPS.

- **Early detection**: Our DDoS discrimination system is able to detect DDoS attacks in any part of the network route from the packet source to the server. This increases chance of implementing it as a cooperative security networks. As soon as the suspicious flash-crowd traffic arrives at the system, we can use an alarm to alert the server in the early stage.

- **Flexibility**: Our DDoS discrimination system could detect any form of attack packets such as malformed IP, TCP, UDP, ICMP, Application-based floods, etc. Our detection methods also work well with low-rate attacks, flash-crowd attacks, shrew attacks, periodical attacks, and pulsing attacks.

- **Ability to Learn**: Our DDoS discrimination system may be enhanced to learn the classification based on its knowledge. Since we have measurement decision modules to provide a double check for the accuracy of results, the knowledge from feedback can be reused by the training algorithm. Hence, the amendable knowledge is supervised to the classification module and maximise the accuracy.

The rest of this chapter is organised as follows. Section 5.2 reviews the background and the related work of our research. In Section 5.3, we summarise the mathematical tools related to our research. Section 5.4 discusses the problems of DDoS attacks and
the specific challenges these raise. In the next section, we propose a solution with the system modelling. In Section 5.6, we provide the results of our experiment with publicly available datasets. In the final section, we provide a summary of this chapter.

5.2 Background and Related Work

5.2.1 Background

A DDoS attack is an anomalous incident when a server is overwhelmed by a large volume of accumulative/simultaneous packet flows. When a server is connected to the Internet and overwhelmed by huge suspicious flows, these flows may be a flash crowd and/or a DDoS attacks as illustrated in Figure 16. In a normal situation, legitimate users do not harm the server or the service. However, a busy server could suffer a flash crowd (FC) event which occurs when there is a sudden high demand in service requests from Internet users. A FC could overwhelm the server and create a similar DoS condition which results in either a delay of response or a complete crash. A DDoS attack is, however, more harmful than a FC (represented as FC1, FC2 and FC3). Zombie machines or attack sources (represented as Z1, Z2 and Z3) are compromised and controlled by attackers, while a botnet attack can be synchronised to overwhelm the victim (represented as Server in Figure 16) during a specific period of time. The situation may be worse when a flash crowd merges with a DDoS attack. This accelerates the DoS condition for the server. As a result, other users (represented as U1, U2 and U3) are unable access to the service.

In our proposed approach, DDoS traffics will be discriminated from legitimate traffics by observing the packet arrival rate. When the attack sources perform a DDoS attack on the victim, their arrival rates appears to be predictable in a given period of time
However, Internet users have a limit on packet generation for their response to the Internet application [75]. For example, after a webpage has been shown, the user may take time to view and respond, for example, clicking on a link. In other words, human users unpredictably create request packets at any period of time. Therefore, we can discriminate the pattern of packet arrivals by using certain mathematical models or statistical analysis.

Our approach has no limitation on the types of attack packets or attack methods. The attack packets can be in many forms such as falsified IP, TCP, UDP, ICMP, and Application-layer packets [75]. We also made sure that our approach handled various attack methods such as a constant rate attack [77], an increasing rate attack [77], a flash-crowd (FC) attack [75], a low-rate (LDoS) attack [84], a Reduction-of-Quality (RoQ) attacks [9, 85], a periodical attack, a shrew attack [67, 85], a pulsing attack [77, 84]) and more importantly, a mimicking attack [78].

5.2.2 Related Work

Both a heuristic-based detection system (HBDS) and a statistical-based detection system (SBDS) have their limitations. While, the key to any SBDS is its ability to learn and distinguish normal from anomalous network activity, the HBDS relies on optimisation of its threshold decision and requires fine-tuning to produce stability, improvement, or precise results of anomaly detection in network traffic. It also relies on optimisation to minimise the false positives/negatives.

An anomaly detection system [42] deploys Support Vector Machine (SVM) and Dynamically Growing Self-Organising Tree (DGSOT) in order to create a learnable algorithm and deploy a clustering analysis respectively. The approach was compared with the Rocchio Bundling technique and random selection in terms of accuracy loss.
and training time gain. SVM + DGSOT achieve the learning system for DDoS detection, however, the results are very low in accuracy (69.8%), very long in training time (13.18 hours), very high in its false negative rate (FNR) (37.8%), and very high in its false positive rate (FPR) (29.8%).

Human-vs.-bot differentiation [75] by human behaviour modelling was proposed with adjustable threshold. This approach derives three aspects of human behaviour: 1) request dynamics, 2) request semantics and 3) the ability to process visual cues. The evaluation process was based on a series of web traffic logs, interlaced with synthetically generated attacks. This heuristic approach discovered the flash-crowd attacks hidden in test traffics with a high accuracy (around 95-99%). The adjustable threshold of 0.05 gave a low FNR (0.00%) and low FPR (1.94%).

A DDoS detection based on Chaos Theory [78] deploys the theory of network self-similarity to differentiate DDoS flooding attack from legitimate self-similar traffic in the network. In the experiment, the authors developed a neural network detector for training by the DDoS prediction algorithm. When the threshold of sensitivity had a range of 88% to 94%, FPR varied with a range of 0.05% to 0.45%.

In the previous chapter, we implemented an approach to discriminate attack traffic from a flash crowd. The detection can extract the repeatable features in a DDoS attack flow with rapid performance. However, we confronted problems differentiating when the attack flow was in mimicking (random) form, as this is similar to a flash crowd.

5.3 Mathematical Models

Based on data from arrival rates, we need mathematical models to identify the degree of prediction. Since we categorise data into predictable and unpredictable data, the
mathematical models must be able to judge the data by using a threshold. These possible models are the tools for self-similarity analysis such as the correlation coefficient and distance matrix. In this chapter, we deploy Pearson’s correlation coefficient, Shannon Entropy and Linear Discriminant Analysis as detection and discrimination tools.

5.3.1 Pearson’s Correlation Coefficient

In this chapter, we select a feature that can measure the degree of dependence in data by using *Pearson’s correlation coefficient* [81] (hereafter called the correlation), which is defined as:

$$\rho_{X,Y} = \frac{\sum(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y} \quad (10)$$

The correlation is used to measure dependence between two quantities (variables), $X$ and $Y$ with expected values $\mu_X$ and $\mu_Y$ and standard deviations $\sigma_X$ and $\sigma_Y$. Both value of the standard deviations are finite and nonzero ($0 < \sigma_X < \infty$ and $0 < \sigma_Y < \infty$). One of the impressive properties of the correlation is symmetric measurement ($\rho_{X,Y} = \rho_{Y,X}$). In other words, whatever data comes first, we can still achieve the same result as measuring.

The correlation value is between -1 and 1 ($-1 \leq \rho_{X,Y} \leq 1$). Hence its absolute value ($|\rho_{X,Y}|$) cannot exceed 1. The absolute correlation value is equal 1 ($|\rho_{X,Y}| = 1$) represented by the stronger relationship between the two variables called *linear dependence*. However, the absolute value from the correlation may reach zero ($|\rho_{X,Y}| = 0$). This means the two variables are uncorrelated. In a special case where both are normal, the uncorrelated result is also equivalent to independence. In this chapter, we define the data that gives us this value of 1 ($|\rho_{X,Y}| = 1$) as predictable.
data with a linear form. The value of 0 ($|\rho_{XY}| = 0$) defines unpredictable data in linear form.

### 5.3.2 Shannon Entropy

We select another feature that can measure the degree of uncertainty in data by using Shannon entropy (here after called the entropy), which is defined as:

$$
H(X) = -\sum_{i=1}^{N} p(x_i) \log p(x_i)
$$

The entropy is a measure of uncertainty/unpredictability for a random variable $X = \{x_i : i = 1, 2, 3, ..., N\}$ where $p(x_i)$ is the probability mass function of outcome $x_i$. The entropy value is between 0 and $\log N$ ($0 \leq H(X) \leq \log N$). The entropy value is equal 0 ($H(X) = 0$) represented by the high predictability of the random variable $X$. However, the entropy value could reach its maximum value ($H(X) = \log N$) if the random variable $X$ is high in uncertainty/unpredictability. In this chapter, we define the data that gives us an entropy value of 0 ($H(X) = 0$) as predictable data. On the contrary, the value of maximum entropy ($H(X) = \log N$) defines unpredictable data.

### 5.3.3 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a statistical approach in classifying objects (people, things, events, etc.) based on theirs sets of features that can be placed in two or more characteristic groups. A feature must be defined as an observation, property, attribute, variable or measurement of an object. In the training process, groups are known or predetermined and do not have order (i.e. nominal scale). A feature set of those objects is then measured which helps to solve the classification problem.
For classification of $g$ groups, the Bayes' rule [86] could minimise total errors by assigning the object to group $i$ which expresses the highest conditional probability against group $j$:

$$P(i|x) > P(j|x), \quad \text{for } \forall j \neq i$$  \hspace{1cm} (12)

However, we cannot achieve the probability $P(i|x)$ of the class directly from the given measurement of the object. We can obtain the measurement and compute the probability for each class by knowing $P(x|i)$ as given in the Bayes Theorem [87]:

$$P(i|x) = \frac{P(x|i) \cdot P(i)}{P(x)} = \frac{P(x|i) \cdot P(i)}{\sum_j P(x|j) \cdot P(j)}$$  \hspace{1cm} (13)

Thus, the Bayes’ rule (Equation 13) becomes:

$$\frac{P(x|i) \cdot P(i)}{\sum_k P(x|k) \cdot P(k)} > \frac{P(x|j) \cdot P(j)}{\sum_k P(x|k) \cdot P(k)}, \quad \text{for } \forall j \neq i$$  \hspace{1cm} (14)

We can simplify the Equation 14 as follow:

$$P(x|i) \cdot P(i) > P(x|j) \cdot P(j), \quad \text{for } \forall j \neq i$$  \hspace{1cm} (15)

We assign the object to group $i$ if the proposed formula (Equation 15) is satisfied. If we have multiple classes and multiple dimensions of measurement, with each dimension having many values, the computation of conditional probability $P(x|i)$ requires a large amount of data. In general, we assume that data comes from some theoretical distribution. In this chapter, we assume that the data comes from the multivariate Gaussian distribution [87] in the following formula:

$$P(x|i) = \left(\frac{1}{k\sqrt{(2\pi)^d \prod_{j=1}^d \Sigma_j^{-1}}}\right) \exp\left(-\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i)\right)$$  \hspace{1cm} (16)
where, $\mu_i$ is vector mean and $\Sigma_i$ is the covariance matrix of group $i$. Replacing the distribution formula (Equation 16) into Bayes’ rule in Equation 15 creates a new formula:

$$\left( \frac{P(i)}{k (2\pi)^\frac{1}{2} |\Sigma_i|^\frac{1}{2}} \right) \exp \left( -\frac{1}{2} (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) \right) >$$

$$\left( \frac{P(j)}{k (2\pi)^\frac{1}{2} |\Sigma_j|^\frac{1}{2}} \right) \exp \left( -\frac{1}{2} (x - \mu_j)^T \Sigma_j^{-1} (x - \mu_j) \right), \quad i \neq j \quad (17)$$

We can simplify Equation 17 by taking out $(2\pi)^\frac{k}{2}$ as follows:

$$\left( \frac{P(i)}{|\Sigma_i|^\frac{1}{2}} \right) \exp \left( -\frac{1}{2} (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) \right) >$$

$$\left( \frac{P(j)}{|\Sigma_j|^\frac{1}{2}} \right) \exp \left( -\frac{1}{2} (x - \mu_j)^T \Sigma_j^{-1} (x - \mu_j) \right), \quad i \neq j \quad (18)$$

We can then take the natural logarithm from both sides of Equation 18:

$$\ln(P(i)) - \frac{1}{2} \ln|\Sigma_i| - \frac{1}{2} (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) >$$

$$\ln(P(j)) - \frac{1}{2} \ln|\Sigma_j| - \frac{1}{2} (x - \mu_j)^T \Sigma_j^{-1} (x - \mu_j), \quad i \neq j \quad (19)$$

We simplify the Equation 19 by multiplying both sides with -2 and then reversing the inequality sign as follows:

$$\ln|\Sigma_i| + (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) - 2 \ln(P(i)) <$$

$$\ln|\Sigma_j| + (x - \mu_j)^T \Sigma_j^{-1} (x - \mu_j) - 2 \ln(P(j)), \quad i \neq j \quad (20)$$
If all covariance matrices are equal ($\Sigma = \Sigma_i = \Sigma_j$), then Equation 20 is simplified as follows:

$$(x - \mu_i)^T \Sigma^{-1} (x - \mu_i) - 2 \ln(P(i)) <$$

$$(x - \mu_j)^T \Sigma^{-1} (x - \mu_j) - 2 \ln(P(j)), \quad i \neq j$$

(21)

In Equation 21, we can rewrite the first term of each side as follows:

$$x \Sigma^{-1} x^T - 2 \mu_i \Sigma^{-1} x^T + \mu_i \Sigma^{-1} \mu_i^T - 2 \ln(P(i)) <$$

$$x \Sigma^{-1} x^T - 2 \mu_j \Sigma^{-1} x^T + \mu_j \Sigma^{-1} \mu_j^T - 2 \ln(P(j)), \quad i \neq j$$

(22)

Finally, we multiply $-\frac{1}{2}$ for both sides of Equation 22, reversing the inequality sign and then simplifying the formula as follows:

$$\ln(P(i)) + \mu_i \Sigma^{-1} x^T - \frac{1}{2} \mu_i \Sigma^{-1} \mu_i^T >$$

$$\ln(P(j)) + \mu_j \Sigma^{-1} x^T - \frac{1}{2} \mu_j \Sigma^{-1} \mu_j^T, \quad i \neq j$$

(23)

Denote that:

$$f_i = \ln(P(i)) + \mu_i \Sigma^{-1} x_k - \frac{1}{2} \mu_i \Sigma^{-1} \mu_i^T, \quad i \neq j$$

(24)

By deriving Equation 23 and 24, we then have the Linear Discriminant Function:

$$f_i > f_j, \quad \text{for } \forall j \neq i$$

(25)

We could assign object by measurement $x$ to group $i$ if Equation 25 is satisfied. In this chapter, we select the two features (data independence ($|\rho_{X,Y}|$) and data predictability ($H(X)$)) as a measurement $x$. Then we use LDA to train and classify measurement $x$.
into two different groups based on its nature. We explain this in detail later in the following section.

5.4 Problem Statement

We considered the situation when a server derives by suspicious traffic flow via its router. As illustrated in 0, a server connects to the Internet and provides a service to public Internet users. In normal situation, the server can handle legitimate users with limited resources such as CPU processes and memory/buffer. However, a busy server could suffer a flash crowd (FC) event which is observed as a sudden high demand in service requests from Internet users. This FC event forces the server to work hard with its limited resources. The FC flow may also overwhelm the server and create a Denial of Service (DoS) condition which results in either a delay of response or a complete crash. Hence, the FC flow must be monitored and considered a suspicious flow when it arrives at the server.

Since the flow could be either legitimate traffic and/or DDoS attack traffic, we treat the flow as suspicious flow, which has the potential to harm the server. The process of investigation and mitigation begins by taking a sample of the arrival rate \( \lambda_k \) of individual traffic from an individual source. Then we measure the degree of dependency and predictability of an individual flow. If the flow can be classified as a dependent and predictable flow, then it is an attack, as shown in Table 6. If the flow can be classified as an independent and unpredictable flow, then it is a legitimate flow of traffic. Otherwise, the flow remains unknown and is considered suspicious.
Table 6 Measurement of a sample flow

<table>
<thead>
<tr>
<th>Dependency</th>
<th>Predictability</th>
<th>Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>Attack</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>Unknown</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>Unknown</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>Legitimate</td>
</tr>
</tbody>
</table>

5.5 System Modelling

The goal of this chapter is to classify a suspicious flow into either legitimate or DDoS attack traffic. To satisfy this investigation process, the following system model (as shown in Figure 32) will be proposed:

![Figure 32 Architecture of a traffic discrimination system](image)
Table 7 List of initial variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Correlation</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time interval</td>
<td>0.1 second</td>
<td>0.1 second</td>
</tr>
<tr>
<td>No. of sample</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

5.5.1 Dependency

In the process of dependency measurement of the arrival rate ($\lambda_k$), we nominate the correlation using Equation 10. We denote $X$ as a sample set of arrival rate ($X = \{\lambda_k : k = 1, 2, 3, ..., N\}$), and $Y$ as a sample set of sequence number ($Y = \{k : k = 1, 2, 3, ..., N\}$). For example, $X = \{\lambda_1, \lambda_2, \lambda_3, ..., \lambda_k\}$ and $Y = \{1, 2, 3, ..., k\}$. Then we calculate the correlation value ($\rho_{X,Y}$) from the two variables ($X$ and $Y$). The value that we expect is between -1 and 1 ($-1 \leq \rho_{X,Y} \leq 1$). Then we pass the dataset of absolute values of the correlation ($|\rho_{X,Y}| = \{|\rho_1|, |\rho_2|, |\rho_3|, ..., |\rho_k| : k = 1, 2, 3, ..., N\}$) to the process of Linear Discriminant Analysis (LDA).

5.5.2 Predictability

In the process of predictably measuring the arrival rate ($\lambda_k$), we nominate entropy using Equation 11. We denote $X$ as a sample set of arrival rate ($X = \{\lambda_k : k = 1, 2, 3, ..., N\}$). For example, $X = \{\lambda_1, \lambda_2, \lambda_3, ..., \lambda_k\}$. Then we calculate the entropy value $H(X)$ from the variable ($X$). The value that we expect is between 0 and $\log N$ ($0 \leq H(X) \leq \log N$). By default, we initial the number ($N$) of sample arrival rate to 10. Then we can expect the entropy to be between 0 and 1 (derived from $\log N = \log 10 = 1$). Finally, the dataset of the entropy ($H(X) = \{H_1, H_2, H_3, ..., H_k : k = 1, 2, 3, ..., N\}$) is passed to the process of Linear Discriminant Analysis (LDA).
5.5.3 Linear Discriminant Analysis (LDA)

In the process of LDA, we derive two features of the arrival rate ($\lambda_k$) which are processed by the correlation ($|\rho_{X,Y}| = \{|\rho_1|, |\rho_2|, |\rho_3|, \ldots, |\rho_k|: k = 1, 2, 3, \ldots, N\}$) and the entropy ($H(X) = \{H_1, H_2, H_3, \ldots, H_k: k = 1, 2, 3, \ldots, N\}$) modules. Then these datasets are processed in one of the following phases:

a) **Training phase:** One of the LDA processes is to let the discrimination system learns the classified (known) object category. We provide some samples from the DDoS attack datasets and legitimate FC dataset to the system. This is an important process for selecting the right training data for our discrimination system. Then the LDA will create its threshold for our system for the purposes of decision making.

b) **Classification phase:** Since the discrimination system supervises the nature of data, it will have a general knowledge to classify the test data that we need to investigate. By using the LDA threshold ($\mathcal{T}$), we can measure the accuracy of our system.

In the process of LDA, we calculate the two features using the Linear Discriminant Function (Equation 25). If the coordinate results ($f(x) = \{(f_i, f_j): \forall i \neq j\}$) are in a training phase, this will create the threshold ($\mathcal{T}$) for further evaluation measurement. If the coordinate results ($f(x) = \{(f_i, f_j): \forall i \neq j\}$) are in classification phase, they will be used to compare and classify the threshold.

5.5.4 Measurement

The approach of measurement against threshold ($\mathcal{T}$) for the classification process is similar to the training process. However, the measurement data ($x$) is tested against
the criteria of probability \( P(i) \) and the vector mean \( \mu_i \) of group \( i \) plus the covariance matrix \( \Sigma \) of all groups. We also tested the measurement data \( x \) with group \( j \) as shown in Figure 33. Then the results were evaluated against the accuracy as shown in Table 8.

Table 8 List of accuracy measurement

<table>
<thead>
<tr>
<th>( f_i &gt; f_j ? )</th>
<th>Result</th>
<th>Fact</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Group ( i )</td>
<td>Group ( i )</td>
<td>True Positive</td>
</tr>
<tr>
<td>True</td>
<td>Group ( i )</td>
<td>Group ( j )</td>
<td>False Negative</td>
</tr>
<tr>
<td>False</td>
<td>Group ( j )</td>
<td>Group ( i )</td>
<td>False Positive</td>
</tr>
<tr>
<td>False</td>
<td>Group ( j )</td>
<td>Group ( j )</td>
<td>True Negative</td>
</tr>
</tbody>
</table>

![Figure 33 Threshold of Decision Making](image)

**5.5.5 Decision Making**

The module of decision making is based on the threshold \( \mathcal{T} \) from the training datasets. As shown in Table 8 and Figure 33, if the measurement data \( x \) satisfies \( f_i > f_j \) (Equation 25), then it is classified into group \( i \). Otherwise, it is in group \( j \). In this chapter, we classify group \( i \) and \( j \) as legitimate traffic and attack traffic.
respectively. Finally, we inform the next module in order to generate a report of the measurement result.

In the case of false detections increasing, this module gives us a double check from the previous and on-going behaviours for measurement data \( (x) \). We are looking forward to developing a learning discrimination system using its feedback as shown in Figure 32. This advance technology could provide us with an artificial intelligence (AI) of a smart DDoS detection system.

### 5.5.6 Security Report

The last part of the discrimination is the module of the security report. After a decision has been made by the decision making module, then an action command/message is then generated to order/inform the security system. We are looking forward to customising this report generation and serving various types of intrusion detection systems (IDS) and intrusion prevention systems (IPS).

### 5.6 Evaluation

In order to measure the discrimination performance, we have to test 2 datasets from real traces (the World Cup 98 website (WC) [82] and the MIT project of MStream attacks (MIT) [72]). We took each sample flow from legitimate and attack traffic from these datasets for the training of our system. Then we took another traffic flows from each dataset to test processes of the classification and measurement. The results of the discrimination processes are:

#### 5.6.1 Results of Dependency

The result of the dependency process derives from the measurement of the arrival rate \( (\lambda_k) \). We took the samples of the arrival rate \( (\lambda_k) \) from the 2 datasets. The experiments
were performed into 2 phases. In the training phase, we selected WC200657 and WC55 as a legitimate flow (group \( i \)) and an attack flow (group \( j \)) respectively. In classification phase, we select the WC1387 and MIT17060 as suspicious flows. We then calculated the degree of predictability for both datasets by using the correlation.

As shown in Figure 34, the correlation values of these datasets have different degrees of dependency. While the WC55 seems to be stable with periodical change in its values, other datasets have varied degrees of dependency. With a high correlation value, the dataset is defined as a high degree of dependency which is linked to the repeatable feature of DDoS attack sources. Otherwise, the low correlation values define the dataset as a low degree of dependency, which is linked to unpredictable (random) behaviour of legitimate users. However, judging either legitimate or attack flow is not accurate enough when using a single method of correlation measurement.

We then deliver all of the correlation values to the training and classification process in order to classify their groups based on the nature of data.

Figure 34 Correlation plots of the datasets for training and classification phases
5.6.2 Results of Predictability

The result of the predictability process derives from the measurement of the arrival rate ($\lambda_k$). We took the samples of the arrival rate ($\lambda_k$) from the 2 datasets. The experiments were performed into 2 phases. In the training phase, we selected the WC200657 and WC55 as a legitimate flow (group $i$) and an attack flow (group $j$) respectively. In the classification phase, we selected the WC1387 and MIT17060 as suspicious flows. We then calculated the degree of predictability for both datasets by using entropy.

As shown in Figure 35, the entropy values of these datasets have different degrees of predictability. While the WC55 seems to be stable with periodical change in its values, other datasets have varied degrees of predictability. With a low entropy value, the dataset is defined as having a high degree of predictability which is linked to the repeatable feature of DDoS attack sources. Otherwise, high entropy values define the dataset as a low degree of predictability, which is linked to unpredictable (random) behaviour of legitimate users. However, judging either a legitimate or attack flow is not accurate enough by only using a single method of entropy measurement. We delivered all of these entropy values to the training and classification process in order to classify their groups based on the nature of the data.
5.6.3 Results of Training LDA

The result of the training process derives from the measurement data \((x)\) which is passed from the Dependency (correlation) and Predictability (entropy) modules. We selected the WC200657 and WC55 as a legitimate flow (group \(i\)) and an attack flow (group \(j\)) respectively, as depicted in Figure 36. Then, we trained the system to know their groups and create their own threshold as depicted in Figure 37 and Figure 38. As shown in Table 9, the training process can accurately discriminate between the legitimate and attack flows 100% of the time. This means that the calculated threshold is perfect for discriminating a legitimate flow from an attack flow in the early stages of the training process.

We can also confirm the high performance of discrimination by the statistics in Table 11 and Figure 39. The threshold gives us the equal average distance (1.327) between
the legitimate group and the threshold, and between the attack group and the threshold. Since we found a good threshold for our discrimination system, we expect this threshold to work well for the test datasets.

Table 9 Result of training

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>False</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

Table 10 Result of classification

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>100.0%</td>
<td>92.3%</td>
</tr>
<tr>
<td>False</td>
<td>0.0%</td>
<td>7.7%</td>
</tr>
</tbody>
</table>

Table 11 List of LDA scores

<table>
<thead>
<tr>
<th>LDA Data</th>
<th>Average</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train Leg.</td>
<td>1.327</td>
<td>0.979</td>
<td>1.614</td>
</tr>
<tr>
<td>Train Att.</td>
<td>-1.327</td>
<td>-2.156</td>
<td>-0.750</td>
</tr>
<tr>
<td>Test Leg.</td>
<td>1.142</td>
<td>0.273</td>
<td>2.089</td>
</tr>
<tr>
<td>Test Att.</td>
<td>-1.829</td>
<td>-4.306</td>
<td>0.670</td>
</tr>
</tbody>
</table>
Figure 36 Scatter plot of original data for training

Figure 37 Scatter plot with threshold from the LDA training process
Figure 38 Scatter plot with threshold from the LDA training process with zoom from Figure 37.

Figure 39 LDA score from the training process.
5.6.4 Results of Classification LDA

The results of the classification process derives from the measurement data \((x)\) which is also passed by the correlation and entropy modules. We selected the WC1387 and MIT17060 as a legitimate flow \((\text{group } i)\) and an attack flow \((\text{group } j)\) respectively as depicted in Figure 40. We measured the accuracy of the classification process against the trained threshold as depicted in Figure 41 and Figure 42. As shown in Table 10, the classification process can accurately discriminate the legitimate flow 100% of the time. However, the attack flow can only be detected with a rate of 92% accuracy and a false negative of 7.7%. This means the legitimate flows has unique behaviour regarding unpredictable packet transmission as well as a high accuracy for detecting DDoS attack flow that can protect the system of the server.

We can also confirm the high performance of discrimination by the statistic from Table 11 and Figure 43. The distance measurement gives us the absolute average LDA score (1.829) of an attack group that is higher than the absolute average LDA score (1.142) of the legitimate group. This means that the discrimination system has a high level of differentiation in the attack flow.
Figure 40 Scatter plot of the original data for classification

Figure 41 Scatter plot with the threshold from the LDA classification process
Figure 42 Scatter plot with the threshold from the LDA classification process with zoom from Figure 41.

Figure 43 LDA score from the classification process.
5.7 Chapter Summary and Conclusion

In this chapter, we proposed an effective approach with a supervised learning system based on Linear Discriminant Analysis (LDA) to discriminate legitimate traffic from DDoS attack traffic. With high efficiency in the deployment of LDA, we introduced a new LDA approach in this research area against DDoS attacks. This helps reduce the complexity and configuration of the discrimination system. In addition, this approach also adopted a method of detecting in human-based behaviour when browsing websites, which is a huge difference from predictable behaviour of DDoS attacks. This method could detect DDoS attacks regardless of the types of attack packets and transmission methods.

Our discrimination system consists of 2 major stages. In the first stage, we need to perform training to provide some knowledge of the known groups of our discrimination system. We performed the experiment with famous trace datasets from the World Cup 98 website and MIT project of MStream attacks. We firstly setup two measurements which were the correlation and entropy to extract the features of data dependency and predictability respectively. Then, we injected these features into LDA as its measurement data. The result of this process was good enough to create its own effective threshold. At the final stage, we used the threshold from the previous process to classify the group as legitimate or attack flows. This classification process provides us with a total accuracy of 96.2%, with 100% accuracy in the detection of legitimate flows.
CHAPTER 6

Web Privacy Protection:

A Perfect Anonymity Approach on Web Browsing

Anonymous web browsing is a topical subject at the moment as we are seeing an increasing number of events performed on the Internet. The dominant strategy of achieving anonymity is packet padding with dummy packets as cover traffic. However, this method introduces two inherent problems: extra bandwidth and extra delay. Therefore, it cannot satisfy both perfect anonymity and strict delay constraints of web browsing. In order to resolve these challenges, we propose a creative approach that uses the predicted web pages that users are going to access as cover traffic rather than dummy packets. Moreover, users may expect a trade-off between the degree of anonymity and cost. We therefore defined the anonymity level as a metric to measure the degrees of anonymity. We established a mathematical model for anonymity systems, and transformed the anonymous communication problem into an optimisation problem, and as a result, users will find trade-offs between the two contradictory constraints. Based on the model, we can describe and compare our proposal and the previous schemas in a theoretical style. We believe that this model offers a solid foundation for further research in this area. The preliminary experiments on the real dataset demonstrated the huge potential for the proposed strategy in terms of resource saving.
6.1 Introduction

The purpose of this chapter is to present an efficient and novel way for web browsing with perfect anonymity. Moreover, we will also focus on solving the contradictory constraints between high level anonymity and long delays in the applications for web browsing. Anonymous web browsing has become a hot topic recently because of the boom in Internet based applications, such as information retrieval, online shopping, and online voting. In order to meet the privacy needs of these kinds of activities, a number of anonymous systems have been proposed, implemented and used on the Internet, such as, mix and mix networks [88], Crowds [89], Onion Routing [90], and Tor [46]. The best solution for defenders is to design a system with perfect anonymity that which can never be breached in any condition. This is possible according to Shannon’s perfect secrecy theory (we use perfect anonymity to replace perfect secrecy in this chapter) [91], however, the cost of perfect anonymity is extremely high, and it may not be practical under certain constraints, for example, the delay constraint for web browsing.

The anonymity issue of web browsing has been widely explored in recent years, with focus on various attacks and defences on the Tor system [44, 92-97]. Traffic analysis is the most powerful tool against anonymous communications. The traffic analysis attack includes two categories: profiling attack [98, 99] and timing attack [100-102]. Rather than obtaining the content of communication, adversaries who use traffic analysis attacks usually focus on whether two entities communicate with each other, which websites the target user accesses, and so on. They try to derive as much information as possible from traffic metadata, such as message lengths, number of packets, and packet arrival time intervals. In terms of a profiling attack, adversaries have a list of possible websites and already have the profiles. This means the task is to
find which websites the target user accesses. Timing attacks [100-102] based on the fact that low-latency anonymous systems, such as onion routing, do not introduce any delays or significantly alters the timing patterns of an anonymous connection. The time interval of arrival packets of HTML text and HTTP objects is usually similar for target users and their adversaries. If they access the same webpage, then it is easy for adversaries to figure out which website the target user accesses from the list.

Researchers currently employ the dummy packet padding technique to fight against traffic analysis. Usually, dummy packets are injected into the intended network traffic to change the patterns or fingerprints of the actual traffic to achieve anonymity [103, 104]. In order to disguise the timing information of connections, the packet rate of a connection should constantly be the same, which mean we need to add dummy packets when real traffic is low or idle, and on the other side, when the actual traffic rate is high, we have to drop some packets to be in alignment with the planned packet rate. This is called link padding [105-107]. Wright, Coull and Monrose [108] recently proposed a traffic morphing method to protect the anonymity of communication. This work is quite creative, however, it still uses a dummy packet padding strategy and requires extra network resources, such as bandwidth.

The strategy of dummy packet padding [109] results in two major problems in communication: extra delay and extra bandwidth demand. These disadvantages are extraordinarily challenging in wireless, ad hoc, and sensor networks [103]. Because of the strict delay constraint from web viewers, a high level of anonymisation on web browsing may not always be achievable using dummy packet padding.

In this chapter, we are motivated by these challenges and propose a novel strategy to resolve these challenges in web browsing. Our proposal comes from the fact that users
usually access a number of web pages at one website according to their own habits and interests, and this has been confirmed by the applications of web caching and web page prefetching technologies [110-112]. Therefore, we can use prefetched data to replace dummy packets for padding. This novel strategy fundamentally solves the problems of extra delay and extra bandwidth cost of packet padding. Moreover, our proposal makes it possible to achieve perfect anonymity of web browsing.

The contributions of this chapter are summarised as follows.

− We propose a novel strategy for packet padding using prefetched data as cover traffic for anonymous web browsing. The proposed strategy makes it possible to achieve perfect anonymity in web browsing under rigorous delay constraints. At the same time, the proposed schema can significantly reduce bandwidth waste and network delay.

− We established a mathematical model to describe and analyse the anonymisation systems. We believe this model can be used for further research in this field.

− We transformed the anonymous communication problem into an optimisation problem based on our model. As a result, we can figure out the tradeoffs between the anonymity level and the cost for applications. This makes it possible for users to find the best anonymity level once the delay constraint is known.

The rest of this chapter is organised as follows. Related work and background are presented in Section 6.2. We finally set the problem up in Section 6.3. In Section 6.4,
we present the details of system modelling and analysis, followed by performance evaluation in Section 6.5. Finally, we summarise this chapter in Section 6.6.

6.2 Background and Related Work

The HTTP protocol document [113] clearly shows that when a client submits an HTTP request to a URL, the corresponding server will deliver the HTML text to the client, with the HTML text including the references of related objects, e.g. images, flashes, etc. The objects will be downloaded to the client one after another, with each web page having its own fingerprint. Some web servers may encrypt the content of packets. However, an observer can clearly see the packet header, which includes sensitive information, such as IP address of the server.

A number of works have already been completed in regards to traffic analysis. Sun et al. [99] tried to identify encrypted network traffic using the HTTP object counts and size. Their investigation demonstrated that it was able to identify a significant fraction of the World Wide Web sites quite reliably. Following this direction, Wright, Monrose and Masson [98] also discovered that website can be identified with a high positive probability even in its encrypted channels. Hintz [104] suggested that noise should be added to user traffic (also referred to as cover traffic in some papers) to users which will change the fingerprints of the server, and transparent pictures should be employed to add extra dummy connections against fingerprint attacks.

Researchers have also explored profiling attacks and proposed solutions. Coull et al. [114] evaluated the strength of the anonymisation method in terms of preventing the assembly of behavioural profiles. They concluded that anonymisation offers less privacy to web browsing traffic than what we expected. Liberatore and Levine [115]
used a profiling method to infer the sources of encrypted HTTP connections. They applied packet length and direction as attributes, and established a profile database for individual encrypted network traffic, and as a result, they knew the source of each individual piece of encrypted network traffic. The adversary obtained the features of the target network traffic and then compared it with the individual record in the profile database, then inferred the possible source of the target traffic. The match technique is based on a similarity metric (Jaccard’s coefficient) and a supervised learning technique (the naïve Bayesian classifier). The extensive experiments demonstrated that the proposed method can identify the source with an accuracy of up to 90%.

Wright, Coull and Monrose [108] recently proposed a traffic morphing method to protect the anonymity of communication. They transformed the intended website (e.g. www.webmd.com) fingerprint to the fingerprint of another website (e.g. www.espn.com). The methods they used included packet padding, and packet splitting. They tested their algorithm against the data set offered in Liberatore and Levine’s work [115], and found that the proposed method improved the anonymity of website access and reduced overhead at the same time.

Venkitasubramaniam, He and Tong [103] notice the delay caused by adding dummy packets into communication channel, and proposed transmission schedules relay nodes to maximise network throughput, thus giving the desired level of anonymity. Similar to the other works, this research was based on the platform of dummy packet padding.

Web caching and prefetching are effective and efficient solutions for web browsing when constrained by bandwidth. Award, Khan and Thuraisingham [112] tried to
predict a user’s web surfing path by using a hybrid model, which combines the Markov model and Supporting Vector Machine. The fusion of the two models complements the weakness of the other and it worked well. Montgomery et al. [116] found that the information of a user’s web browsing path offered important information for a transaction, and a successful deal usually followed by accessing the same path. Teng, Chang and Chen [111] argued that there must be elaborate coordination between client side caching and prefetching. They also formulated a normalised profit function to evaluate the profit of caching an object, with the proposed function integrating a number of factors, such as object size, fetching cost, reference rate, invalidation cost, and invalidation frequency. Their event-driven simulations demonstrated that the proposed method performed well.

6.3 Problem Statement

6.3.1 Background of the Problem

We suppose that an adversary (Bob) is focused on finding which website the monitored user (Alice) accesses from a list of possible websites. We suppose Bob has the knowledge of all websites on the list, and he is also able to captures all network traffics from Alice’s computer.

In general, every web page is different from the others, such as length of HTML text, number of web objects, timing of packet transportation, and number of packets for each web object. We use the web fingerprint to represent the uniqueness of a website. The web service may be accessed in an encrypted way for a user, such as using SSL. However, encryption brings limited changes to the fingerprint.
We suppose there are \( n \) possible websites that Alice accesses, \( \{w_1, w_2, w_3, \ldots, w_n\} \), and this is known to Bob. The prior of \( w_i (1 \leq i \leq n) \) is denoted as \( p(w_i) (1 \leq i \leq n) \). For each website \( w_i (1 \leq i \leq n) \), we denote its fingerprint with \( \{p_i^1, p_i^2, p_i^3, \ldots, p_i^k\} \). For example, for a given website \( w_i \), if we count the number of packets for every web object, such as HTML text, different images etc., and save them as \( \{x_1, x_2, x_3, \ldots, x_k\} \).

We unify this vector and obtain the distribution as \( \{p_i^1, p_i^2, p_i^3, \ldots, p_i^k\} \), where \( p_i^j = x_j \left( \sum_{m=1}^{k} x_m \right)^{-1}, 1 \leq j \leq k \). Bob monitors Alice’s local network, and makes a number of observations

\[ \tau = \{\tau_1, \tau_2, \tau_3, \ldots\} \]  

(26)

Based on these observations and the Bayesian Theorem, Bob can claim that Alice accesses website \( w_i \) with the following probability

\[
p(w_i | \tau) = \frac{p(w_i) \cdot p(\tau|w_i)}{p(\tau)}
\]  

(27)

On the other hand, the task for defenders is to decrease \( p(w_i | \tau) \) to the minimum by anonymisation operations, such as packet padding, and link padding.

According to Shannon’s perfect secrecy theory [91], an adversary cannot break the anonymity if the following equation holds.

\[
p(w_i | \tau) = p(w_i)
\]  

(28)

Namely, observation \( \tau \) offers no information to the adversary. However, the cost for perfect anonymity is extremely expensive by injecting dummy packets according to the perfect secrecy theory [91].
6.3.2 Anonymity Measure

Because of the strict delay constraint of a user’s perception, we cannot always achieve perfect anonymity in web browsing. Suppose a user’s perception threshold is $\Delta$ in terms of seconds, we can bear this constraint in mind and try our best to improve the anonymity of web browsing. In order to measure the degree of anonymity, we create the following definition.

**Definition 1:** Anonymity level. Let $S$ be the real network traffic for a given session of network activity, $S$ may be encrypted or covered. Therefore, the adversary can only obtain an observation $\tau$ about $S$. We define a measure for the degree of anonymity as follows.

$$\alpha = \frac{H(S|\tau)}{H(S)}$$

(29)

Where $H(S)$ is the entropy [70] of the random variable $S$, which is defined as follows.

$$H(S) = -\sum_{s \in \chi} p(s) \log p(s)$$

(30)

where $p(s)$ is the distribution of $S$ and $\chi$ is the probability space of $S$.

$H(S|\tau)$ is the conditional entropy [70] of $S$ given $\tau$, which is defined as follows.

$$H(S|\tau) = -\sum_{\tau_i \in \Gamma} p(\tau_i)H(S|\tau = \tau_i)$$

(31)

where $p(\tau_i)$ is the distribution of $\tau$ and $\Gamma$ is the probability space of $\tau$.

Because $H(S|\tau) \leq H(S)$, we have $0 \leq \alpha \leq 1$. Following this definition, we obtain $\alpha = 1$ when $H(S|\tau) = H(S)$ holds, namely, $S$ and $\tau$ are independent of each
other, and the adversary cannot infer any information about \( S \) from \( \tau \), therefore we achieve perfect anonymity. If \( \tau = S \), then we have \( \alpha = 0 \), namely the adversary is absolutely sure about session \( S \), and there is no anonymity to him at all.

Let \( S \) be the intended traffic. In order to achieve a given anonymity level \( \alpha \), we need a cover for traffic \( Y \). We use function \( C(X) \) to represent the cost of network traffic \( X \). The cost could be bandwidth, delay or number of packets, etc. Therefore, the cost for the intended traffic is \( C(S) \), and the cost of cover traffic under anonymity level \( \alpha \) is \( C(Y|S, \alpha) \). We define a measure for the cost as follows.

**Definition 2: Anonymity cost coefficient.** For a given network traffic \( S \), in order to achieve an anonymity level \( \alpha \), we inject cover traffic \( Y \). The anonymity cost coefficient is defined as

\[
\beta = \frac{C(Y|S, \alpha)}{C(S)}
\]  

(32)

We expect \( \beta \) to be small as possible in practice.

### 6.4 System Modelling and Analysis

In this section, we will model the anonymisation system with our definitions and compare the proposed strategy against the dummy packet padding schemas based on our model.

#### 6.4.1 System Modelling

Real applications, situations and accessing patterns can be very complex. In this chapter, we focus on the novelty of our anonymisation mechanism and the potential of
the proposed strategy. Therefore, we make the following assumptions in order to make our explanations clear.

- We only study cases where the adversary focuses on one target user and knows the possible list of websites that the target user accesses.
- We focus on the anonymity of network traffic sessions, and ignore the link padding issue in this chapter.
- We only discuss the cases of anonymity achieved by packet padding. There is no packet splitting in this study, as the cost is the extra delay, rather than bandwidth.
- We only analyse the attack method of packet counting. Our model and strategy is also effective for other attack methods, such as packet arrival time intervals.
- We suppose Alice accesses one website for one session, and she opens the default web page (index.html) of the website first, then follows the hyperlinks to open other web pages at the website. Our model and strategy can deal with other accessing patterns, however we need more time and computing.

Figure 44 A packet padding system for anonymous communication.
A typical anonymous communication with packet padding is shown in Figure 44. As a user, Alice sends an HTTP request to web server \( w_i \) through an anonymous channel. The web server returns the intended traffic \( P = \{P_1, P_2, P_3, \ldots, P_k\} \), where \( P_i \) represents the number of packets of web object \( i \). Let \( \|P\| = \sum_{i=1}^{k} P_i \) denote the total number of packets of the intended traffic \( P \), and we extract the fingerprint of this session as \( p = \{p_1, p_2, p_3, \ldots, p_k\} \), where \( p_i = P_i/\|P\|, 1 \leq i \leq k \), and \( \sum_{i=1}^{k} p_i = 1 \). In order to make it anonymous to adversaries, we create the cover traffic \( Q = \{Q_1, Q_2, Q_3, \ldots, Q_k\} \) at the server side, where \( Q_i \) denotes the number of packets that is expected by \( P_i \), and \( \|Q\| = \sum_{i=1}^{k} Q_i \). Similar to intended traffic \( P \), the fingerprint of \( Q \) is \( q = \{q_1, q_2, q_3, \ldots, q_k\} \). We use \( \oplus \) to represent the anonymisation operations, and \( V = P \oplus Q \) is the output of the anonymisation operation on \( P \) and \( Q \). The fingerprint of \( V \) is \( v = \{v_1, v_2, v_3, \ldots, v_k\}, \sum_{i=1}^{k} v_i = 1 \), and \( \|V\| \) is the total packet number of \( V \). The adversary’s observation \( \tau \) is the mix of \( V \) and other network background traffic. Once \( V \) arrives at the user’s side, there is a de-anonymisation operation to recover \( P \) from \( V \).

In previous work, dummy packets are employed to work as the cover traffic \( Q \). In this chapter, we propose to use prefetching web pages as cover traffic, rather than dummy packets. Let \( W \) be the set of all possible contents of website \( w_i \), \( S \) be a traffic session of accessing \( w_i \), and \( \Delta \) be the threshold of the maximum cost that users’ can tolerate. Then the anonymisation problem can be transformed into an optimisation issue as follows:

Maximise \( \alpha \)

s. t.
\[ C(S \oplus Y) \leq \Delta \]
\[ S \in W \]
\[ Y \in W \]
\[ \|Y\| \leq C_b \]  \hspace{1cm} (33)

Where \( C_b \) is the storage capacity of the client, we suppose \( C_b = \infty \) in this chapter.

In previous dummy packet padding solutions, \( Y \notin W \) and \( S \cap Y = \phi \).

### 6.4.2 System Analysis

Following the definitions and expressions in the previous section, and in order to achieve perfect anonymity, the following equations must hold \( v_1 = v_2 = v_3 = \ldots = v_k \).

Suppose \( p_m = \max\{p_1, p_2, p_3, \ldots, p_k\} \), then the minimum cover traffic to achieve perfect anonymity is given as follows,

\[ Q = \{(p_m - p_1) \cdot \|P\|, (p_m - p_2) \cdot \|P\|, (p_m - p_3) \cdot \|P\|, \ldots, (p_m - p_k) \cdot \|P\|\} \]  \hspace{1cm} (34)

We have fingerprint of \( Q \) as

\[ q = \{q_1, q_2, q_3, \ldots, q_k\} = \{(p_m - p_1), (p_m - p_2), (p_m - p_3), \ldots, (p_m - p_k)\} \]  \hspace{1cm} (35)

Then the anonymity cost coefficient in terms of the number of packet is

\[ \beta = \frac{\sum_{i=1}^{k} Q_i}{\sum_{i=1}^{k} P_i} = \frac{\|Q\|}{\|P\|} \]  \hspace{1cm} (36)
In general, given the intended traffic $P$ and an anonymity level $\alpha \ (0 \leq \alpha \leq 1)$, the cost of the cover traffic could be expressed as

$$C(Q|P, \alpha)$$  \hspace{1cm} (37)

For dummy packet padding strategies, the anonymity cost coefficient $\beta_d$ could be denoted as follows:

$$\beta_d = \frac{C(Q|P, \alpha)}{C(P)}$$  \hspace{1cm} (38)

In our proposed strategy, the cover traffic is part of $P$ in a long term point of view. The extra cost is caused by the data we prefetched but did not use. If we suppose the prefetching accuracy is $\eta \ (0 \leq \eta \leq 1)$, then the extra cost from the cover traffic is as follows.

$$Q' = P \cdot (1 - \eta)$$  \hspace{1cm} (39)

Then $\beta_p$, the anonymity cost coefficient of our strategy is

$$\beta_p = \frac{C(Q'|P, \alpha)}{C(P)} = \frac{C(P \cdot (1 - \eta)|P, \alpha)}{C(P)}$$  \hspace{1cm} (40)

If $\eta = 0$, then $\beta_p = \beta_d$, namely the anonymity cost coefficient of our strategy is the same as the others. On the other hand, if $\eta = 0$, then $\|Q\| = 0$, we have $\beta_p = 0$. In other words, there is no wastage of resources at all. In general, for $0 \leq \eta \leq 1$, if the cost function $C(\cdot)$ is a linear function, then we can simplify $\beta_p$ as follows.

$$\beta_p = (1 - \eta) \cdot \alpha$$  \hspace{1cm} (41)
6.5 Performance Evaluation

In order to show the potentials of the proposed strategy, we extract the relationship between the anonymity level and the cost against the real data set used in [115]. The same data set was also used in [108] to indicate the cost efficiency of their morphing strategy.

The data set included the tcpdump files of 2000 websites from February 10, 2006 to April 28, 2006, which are sorted by popularity from Site0 to Site1999 from the most to the least. There are a few different sessions everyday for each website and, for each session users may access different web pages in the same day. We take Site0, Site500, and Site1999 as three representatives from the dataset. For a given network session, we use the number of packets for web objects as fingerprints. For the same website, user may access different web pages, therefore, the fingerprint for each session is only a part of the entire fingerprint. The fingerprints of the aforementioned 6 sessions are shown in Figure 45.

In the case of perfect anonymity, the number of packets for every web object was the same since we added cover traffic to the intended traffic. In order to obtain a different level of anonymity, we partially started the perfect anonymity operation, and pad the related cover traffic to the intended traffic, one web object at a time, starting from the first web object to the last. This is a simple way for anonymisation with $\alpha < 1$. The result of the anonymity level against the progress of anonymisation is shown in Figure 46. The results show that $\alpha$ approaches 1 when we process increasing numbers of web objects. This indicates that a higher anonymity level requires more cover traffic, and this matches our analysis in the previous sections.
Figure 45 Fingerprints of network sessions

Figure 46 Anonymity level against anonymisation progress on web objects
In order to show the direct relationship between anonymity level and the anonymity cost coefficient, we carefully extracted this information from the 6 network sessions. The results are presented in Figure 47 as follows.

![Figure 47 Anonymity cost coefficient (ACC) against anonymity level](image)

From Figure 47, we find that the cost for perfect anonymity is extremely expensive. For example, the volume of cover traffic for Site1999-2 is more than 120 times that of the intended traffic and is more than 10 times for a simple website, such as Site500, which has a small number of packets as indicated in Figure 45. These preliminary experiments showed the huge potential for saving resources from our proposed strategy. Moreover, the experiments on Site0 in Figure 47 show an anomaly at the
beginning where more cover traffic results in a lower anonymity level. We are extremely interested to investigate this in our future research.

### 6.6 Chapter Summary and Conclusion

We noticed that the cost of dummy packet padding is very expensive for anonymous web browsing. Therefore, a novel strategy is proposed to significantly reduce the cost for web browsing – take web prefetched data as cover traffic instead of dummy packets. Moreover, it is hard to achieve perfect anonymity in some cases of web browsing and therefore, we defined a measure (anonymity level) to be a metric for measuring user requirements of anonymity. We modelled the anonymous system in a theoretical way, and described the relationship between the anonymity level and the cost in theory. Furthermore, we transformed the anonymity problem into an optimisation problem, and this transformation brought numerous possible solutions from the optimisation field. We believe that this model offers a solid platform for further research.
CHAPTER 7

Enhanced Web Privacy Protection: Anonymous Web Browsing through Predicted Pages

Anonymous web browsing is an emerging topic of interest with many potential applications for privacy and security. However, most of the related research focuses heavily on latency anonymous communications. The research on light latency anonymous communication, such as web browsing, is quite limited. One reason for this is the huge delay caused by the current dominant dummy packet padding strategy. As a result, it is hard to achieve perfect anonymity for web browsing applications. We have proposed the use of prefetched web pages as cover traffic to obtain perfect anonymity in web browsing. Based on Shannon’s perfect secrecy theory, we formally establish a mathematical model for the problem, and define a metric to measure the cost of achieving perfect anonymity. The experiments on the real-trace dataset demonstrated that the proposed strategy can reduce the delay ten times compared to the dummy packet padding methods. This fact alone confirms the vast potential for our proposed strategy.

7.1 Introduction

The purpose of this chapter is to present a novel strategy to achieve perfect anonymity in web browsing. Anonymous web browsing is demanded by Internet users for privacy and security reasons, yet web browsing with perfect anonymity has rarely been achieved using traditional methods, such as packet padding. According to the recent ACM survey [109], anonymous communication systems can often be classified
into two general categories: high-latency systems and low-latency systems. High-latency anonymity systems are able to provide strong anonymity, but are typically only applicable for non-interactive applications that can tolerate delays of several hours or more, such as mix networks [88] for email messages. On the other hand, low-latency anonymity systems often provide better performance and are intended for real-time applications, particularly web browsing. The examples for this category are the Tor system [46] and, the Crowds system [89]. Because of the strict time constraints, a significant challenge is posed in creating perfect anonymity in low-latency systems, such as web browsing. The general goal of attacks on anonymous communication is to identify pairwise entities in the systems, rather than the content of the communication, which is usually encrypted. For example, does user A communicate with user C? Does user D access website E?

Data encryption is usually used to provide security. However, data encryption for anonymous web browsing is vulnerable to traffic analysis. In general, every web page is different, and differences may include length of HTML text, number of web objects, number of packets for each web object, or timing information of packet transportation. Each website has its own distinctive features which are a combination of all features of its web pages. In this chapter, we use fingerprints to represent the uniqueness of a web page. A website may be accessed in an encrypted way, such as using SSL, however, encryption brings limited changes to the fingerprint [99]. As a result, attackers usually use traffic analysis techniques to break the anonymity in communication [92, 93].

Packet padding techniques are normally employed on top of data encryption to protect anonymity during web browsing. Perfect anonymity in communication means that anonymity cannot be breached in any situation. According to Shannon’s perfect
secrecy theory [91], perfect anonymity in communication is possible. Researchers currently employ packet padding techniques to disguise fingerprints in communication. For example, hiding fingerprints for traffic sessions [103, 104]. Moreover, in order to hide the timing information of connections, link padding is employed [106]. Currently, the dominant strategy of packet padding is using dummy packets as cover traffic. This strategy results in two major problems for communication: huge delay and extra bandwidth demand. Because of the strict delay constraints from web viewers, it is almost impossible to achieve perfect anonymity in web browsing using the dummy packet padding strategy.

In this chapter, we propose a novel approach to address the aforementioned problem, which is an extension of our previous work [117] as detailed in Chapter 6. In our previous work, we have shown the great potential for obtaining anonymity of web browsing using prefetched data and in this chapter we explore further aspects. Our proposal comes from the fact that users generally access a number of web pages from one website according to their own habits or interests. This has been confirmed by applications of web caching and web page prefetching technologies [111, 112]. Therefore, we can use prefetched data to replace dummy packets for padding. We propose to disguise the fingerprints of web sites at the server side by injecting predicted web pages that users download as cover traffic, rather than using dummy packets as cover traffic. From a long term point of view, this novel strategy wastes limited bandwidth and causes limited delay.

The contributions of this chapter are listed as follows.

- We propose to use predicted web pages to conduct packet padding for web browsing, instead of using dummy packets. This fundamentally addresses the problems of extra delay and extra bandwidth demand which is an issue for
traditional dummy packet padding methods. Based on our knowledge, this chapter proposes a brand new strategy for anonymous web browsing.

- We establish a simple mathematical model based on the perfect secrecy theory for the proposed strategy and define a metric to measure the cost of web browsing with perfect anonymity.

The rest of this chapter is structured as follows. Section 7.2 introduces related work. We present the setting of the problem in Section 7.3, followed by the system modelling and analysis in Section 7.4. The preliminary performance evaluations are conducted in Section 7.5. Finally, Section 7.6 summarises the chapter.

### 7.2 Background and Related Work

The HTTP protocol document [113] shows that when a client submits an HTTP request to a URL, the corresponding server will deliver the HTML text to the client, with the HTML text including the references of the related objects such as images and flashes. The objects will be downloaded to the client one after the other. Therefore, each web page has its own fingerprint in terms of the number of web objects, packet arrival time intervals, and so on. Some web servers may encrypt the content of packets, however, the fingerprint cannot be disguised by the encryption against traffic analysis.

A number of works have been completed in terms of traffic analysis. Sun et al. [99] tried to identify encrypted network traffic using the HTTP object number and size. Their investigation showed that it was sufficient to reliably identify a significant fraction of World Wide Web sites. Following this direction, Wright, Monrose and
Masson [98] further confirmed that websites can be identified with high probability even if it is an encrypted channel. Hintz [104] suggested that noise should be added to traffic (also referred to as cover traffic in some papers) which will change the fingerprints of the server, and transparent pictures should be employed to increase the dummy connections and counter fingerprint attacks.

Researchers also explored profiling attacks and proposed solutions. Timing attacks [100, 101] based on the fact that low-latency anonymous systems, such as onion routing, do not introduce any delays or significant changes to the timing patterns of an anonymous connection. The time intervals of the arrival packets of HTML texts and HTTP objects are usually similar for the target user and the adversary. If they access the same web page, then it is easier for the adversary to figure out which website the target user accessed from the list. Coull et al. [114] evaluated the strength of the anonymisation method in terms of preventing the assembly of behavioural profiles, and concluded that anonymisation offers less privacy to web browsing traffic than what we expected. Liberatore and Levine [115] used a profiling method to infer the sources of encrypted HTTP connections. They applied packet length and direction as attributes, and established a profile database for individual encrypted network traffic. Based on this information, they can infer the source of individual encrypted network traffic. The match technique is based on a similarity metric (Jaccard’s coefficient) and a supervised learning technique (the naive Bayesian classifier). Their extensive experiments showed that the proposed method can identify the source with an accuracy of up to 90%.

Wright, Coull and Monrose [108] recently proposed a traffic morphing method to protect the anonymity of communication. They transformed the intended website (e.g. www.webmd.com) fingerprint to the fingerprint of another web site (e.g. www.example.com)
The transformation methods they took include packet padding and packet splitting. Optimal techniques were employed to find the best cover website (in terms of minimum cost for transformation) from a list. They tested their algorithm against the dataset offered in Liberatore and Levine’s work [115], and found that the proposed method can improve the anonymity of accessing website and reducing overheads at the same time. Venkitasubramaniam, He and Tong [103] noticed the delay caused by adding dummy packets into communication channels, and proposed transmission schedules on relay nodes to maximise network throughput given a desired level of anonymity. Similar to other work, this is also based on the platform of dummy packet padding.

Web caching and prefetching are also effective and efficient solutions for web browsing when bandwidth is a constraint. Award, Khan and Thuraisingham [112] tried to predict paths of web surfers using a hybrid model, which combines the Markov model and the Supporting Vector Machine. The fusion of the two models complements the weakness of the other and they work together well. Montgomery et al. [116] found that the information from a user’s web browsing path offers important information for a transaction, with a successful deal usually followed by a number of similar accessing path. Teng, Chang and Chen [111] argued that there must be an elaborate coordination between client side caching and prefetching, with formulation of a normalised profit function to evaluate the profit from caching an object. The proposed function integrated a number of factors, such as object size, fetching cost, reference rate, invalidation cost, and invalidation frequency. Their event-driven simulations showed that the proposed method performed well.
7.3 Problem Statement

Alice accesses web sites via encrypted channels, and an adversary (Bob) focuses on identifying which website Alice chooses from a list of possible websites. We suppose Bob has knowledge of all the websites on the list. Bob also captures all network traffic from Alice’s computer. We assume there are \( n \) possible websites that Alice accesses, \( \{w_1, w_2, w_3, ..., w_n\} \). The priori of \( w_i(1 \leq i \leq n) \) is denoted as \( p(w_i)(1 \leq i \leq n) \). For each website \( w_i(1 \leq i \leq n) \), we denote its fingerprint with \( \{p_i^1, p_i^2, p_i^3, ..., p_i^k\} \). For example, for a given website \( w_i \), we count the number of packets for every web object, such as HTML text, different images etc., and save them as \( \{x_1, x_2, x_3, ..., x_k\} \). We unify this vector and make the distribution as \( \{p_i^1, p_i^2, p_i^3, ..., p_i^k\} \), where \( p_i^j = x_j(\sum_{m=1}^{k} x_m)^{-1} \), \( 1 \leq j \leq k \). Bob monitors Alice’s local network, and obtains a number of observations

\[
\tau = \{\tau_1, \tau_2, \tau_3, ... \}
\]

(42)

Based on these observations and Bayesian Theorem, Bob can claim that Alice accesses website \( w_i \) with the following probability.

\[
p(w_i|\tau) = \frac{p(w_i) \cdot p(\tau|w_i)}{p(\tau)}
\]

(43)

where \( p(w_i) \), \( p(\tau|w_i) \) and \( p(\tau) \) are known to Bob, because Bob can actually access the \( n \) websites individually to obtain this information.

On the other hand, the task for Alice is to decrease \( p(w_i|\tau) \) to the minimum. As we know that data encryption itself cannot achieve the goal of anonymity, Alice has to employ further anonymisation operations, such as packet padding and link padding to fight against Bob.
According to Shannon’s perfect secrecy theory [91], an adversary cannot break the anonymity if the following equation holds.

\[ p(w_i | \tau) = p(w_i) \]  (44)

Namely, the observation offers no information to the adversary. However, the cost for perfect anonymity is extremely expensive by injecting dummy packets according to the perfect secrecy theory. In order to measure the cost for perfect anonymity, we define the following:

**Definition:** *Cost Coefficient of Anonymity.* Let function \( C(S) \) represents the cost function for a given network traffic \( S \). For given intended network traffic \( X \), we inject cover traffic \( Y \) to achieve the goal of anonymity, then the cost coefficient of anonymity is defined as

\[ \beta = \frac{C(Y | X) + C(X)}{C(X)} \]  (45)

This metric will be used to indicate the cost efficiency for perfect anonymity operations in this chapter.

### 7.4 System Modelling and Analysis

In real applications, accessing patterns can be very complex. However, in this chapter, we focus on presenting the effectiveness of our proposed strategy as a new anonymisation method, and demonstrate the great potential for our proposed strategy. Therefore, we confined our research space with the following conditions:

- We focus on perfect anonymity of network traffic sessions and ignored the link padding issue.
- We only discuss cases for achieving anonymity by packet padding and exclude the packet splitting operations.
We use the number of packets for web page objects as the fingerprint and focus on this kind of attacks in this context. We do not discuss timing traffic analysis attacks in this chapter.

A typical anonymous web browsing system with data encryption (at the Internet channels) and packet padding (at the server side) is shown in Figure 48. As a client, Alice sends a HTTP request to web server $w_i$ via an encrypted channel. The web server also employs an encrypted channel to return the intended traffic $X = \{x_1, x_2, x_3, ..., x_k\}$, where $x_i (1 \leq i \leq k)$ represents the number of packets for web object $i$. Let $\|X\| = \sum_{i=1}^{k} x_i$ denote the total number of packets for the intended traffic $X$. We then extract the fingerprint of this session as $p = \{p_1, p_2, p_3, ..., p_k\}$ where $p_i = x_i/\|X\|, 1 \leq i \leq k$, and $\sum_{i=1}^{k} p_i = 1$. In order to make it anonymous to adversaries, we create cover traffic $Y = \{y_1, y_2, y_3, ..., y_k\}$ at the server side. Let $y_i (1 \leq i \leq k)$ denotes the number of packets assigned to cover $x_i$, and let $\|Y\| = \sum_{i=1}^{k} y_i$. Figure 48 A packet padding system for anonymous web browsing

\[\text{de-anonymization operations}\]

\[\text{encrypted channel}\]

\[\text{anonymization operations}\]

\[\text{web server}\]

\[\text{user}\]

\[\text{adversary}\]
\( \sum_{i=1}^{k} y_i \). Similar to intended traffic \( X \), the fingerprint of \( Y \) is \( q = \{q_1, q_2, q_3, \ldots, q_k\} \). If \( Z = \{z_1, z_2, z_3, \ldots, z_k\} \) represents the mixture of intended traffic \( X \) and cover traffic \( Y \), then the fingerprint of \( Z \) is \( r = \{r_1, r_2, r_3, \ldots, r_k\} \), and the total number of mixed traffic is \( \|Z\| \). The adversary’s observation \( \tau \) is the mixture of \( Z \) and other background traffic on the network.

In the previous chapter, dummy packets were employed to work as cover traffic \( Y \). Once \( Z \) arrives at the client side, dummy packet \( Y \) will be discarded. Another solution is to use transparent images as cover traffic. However, in the proposed strategy, the predicted web data is used as cover traffic \( Y \), with the client decomposing the received traffic, intended traffic \( X \) goes to the web browser, prefetched data \( Y \) is stored in the cache of the local computer, and \( Y \) may be used by the following requests. In this case, the client will fetch the expected web data from the cache rather than downloading it again from the server. From a long term point of view, the bandwidth is not wasted and the average extra delay is limited in the proposed scheme.

In order to achieve perfect anonymity as described by Shannon [91], the following equation must hold.

\[
   r_1 = r_2 = r_3 = \cdots = r_k
\]  

(46)

Furthermore, the following condition must also hold.

\[
   z_1 = z_2 = z_3 = \cdots = z_k
\]  

(47)

This means that every traffic session is the same. As a result, Bob cannot obtain any information from his observation.

Let \( x_{max} = \max\{x_1, x_2, x_3, \ldots, x_k\} \), with the minimum cover traffic to achieve perfect anonymity given as follows.
\[
\begin{align*}
    y_1 &= x_{\text{max}} - x_1 \\
    y_2 &= x_{\text{max}} - x_2 \\
    y_3 &= x_{\text{max}} - x_3 \\
    \vdots \\
    y_k &= x_{\text{max}} - x_k
\end{align*}
\]  
(48)

Let the cost function \( C(\cdot) \) be the number of packets, with the cost coefficient for anonymity expressed as follows.

\[
\beta = \frac{\|Z\|}{\|X\|} = \frac{\sum_{i=1}^{k} (x_i + y_i)}{\sum_{i=1}^{k} x_i}
\]  
(49)

Let \( \beta_d \) represent the cost coefficient for perfect anonymity using the dummy packet padding strategy, then

\[
\beta_d = \frac{\|Z\|}{\|X\|} = \frac{k \cdot x_{\text{max}}}{\sum_{i=1}^{k} x_i}
\]  
(50)

On the other hand, with the proposed strategy, the cover traffic is part of \( X \) in the long term. The extra cost for the proposed mechanism is part of the cover traffic prefetched by the client but never accessed. We define the missing rate as the ratio of the hitless prefetched data and the total prefetched data. Let \( \eta \) \((0 \leq \eta \leq 1)\) be the missing rate in the local cache, then the cost coefficient of perfect anonymity of the proposed strategy \( \beta_p \) is

\[
\beta_p = \frac{\eta \cdot \|Y\| + \|X\|}{\|X\|} = \frac{\sum_{i=1}^{k} \eta \cdot (x_{\text{max}} - x_i)}{\sum_{i=1}^{k} x_i} + 1
\]  
(51)

Comparing Equation 50 and Equation 51, we reach the following conclusion.

\[
\beta_p \leq \beta_d, \quad (0 \leq \eta \leq 1)
\]  
(52)
The worst case of the proposed strategy is when $\eta = 1$ (all prefetched data is unused in the future), $\beta_p = \beta_d$ holds. In other words, our strategy is no worse than dummy packet padding approaches.

### 7.5 Performance Evaluation

In order to confirm the advantages of the proposed strategy, we conducted two preliminary experiments using a real world data set [118], which is widely used by the community, such as in [108] and [115]. The dataset includes the tcpdump files of 2000 websites from February 10, 2006 to April 28, 2006. These have been sorted by popularity with all data encrypted. We took 30 continuous days from the most popular web site as the data set for the experiments in this chapter, and treated each day as one session. We extracted the fingerprint (number of TCP packets for each web page object) of every session for the 30 days.

We first investigated the cost coefficient of perfect anonymity for the proposed strategy with different missing rate (namely, different prefetching accuracy). The results are shown in Figure 49.
We can see that when there is nothing missing ($\eta = 0$), the cost coefficient for perfect anonymity achieves the minimum, 1, which is the ideal for users. When the missing rate is 0.5 ($\eta = 0.5$), the mean of the cost coefficient for perfect anonymity is around 7.59. Furthermore, the mean of the cost coefficient for perfect anonymity is 14.35 when all prefetched data is missing ($\eta = 1$, equal the case of dummy packet padding strategy), with this variation depending on the fingerprint distribution of web pages. This preliminary experiment indicates that the cost for perfect anonymity using the dummy packet padding strategy is much higher (around 15 times more traffic volume and consequently, a 15 fold increase in delays) than using the proposed strategy. In other words, our strategy can reduce the delay up to 15 times compared to the dummy packet padding method.
We are also interested in the relationship between the cost coefficients of perfect anonymity against the length of a session. We took 4 random samples from the data set over the 30 days period. We calculated the cost coefficient of perfect anonymity for the dummy packet padding strategy against the length of each session (we increased the length of sessions in the experiment). The results are shown in Figure 50. The cost coefficient of perfect anonymity for the dummy packet padding strategy was an increase in function against the length of each session. Every change point indicated bigger objects in terms of packet number in the past, namely, the change point depended on the distribution of larger objects. In other words, the longer the session length was, the higher the cost for the dummy packet padding method in order to achieve perfect anonymity. However, this was not a problem for our proposed strategy.
7.6 Chapter Summary and Conclusion

In this chapter, we propose a novel strategy to achieve perfect anonymity in web browsing by using prefetched data as cover traffic, rather than using dummy packets as cover traffic. The proposed strategy made web browsing with perfect anonymity much easier to achieve for Internet users, which was extremely hard to accomplish using the traditional dummy packet padding strategy. We established a mathematical model for the problem based on Shannon’s perfect secrecy theory, and our analysis demonstrated that the proposed strategy was always equal to or better than the dummy packet padding strategy in terms of delay. Furthermore, we revealed the huge advantages of the proposed method in this chapter. The preliminary experiments confirmed our theoretical analysis, and demonstrated that the proposed strategy outperforms the traditional dummy packet padding method approximately 15 times in terms of delay and bandwidth cost.
CHAPTER 8

Conclusion and Future Works

8.1 Methodology and Thesis Overview

This thesis has covered a range of algorithms that help to improve the security of web services. We have focused on the problems of DDoS attack and traffic analysis attack against service availability and information privacy respectively.

In DDoS attacks, attackers can set up different types of attack sources and then implement various packet-transmission strategies and various packet forms to disrupt the available services of their victims. Consequently, DDoS defence systems need to deploy variety of DDoS detection methods in order to mitigate the damage from the attacks. This can be slow depending on computational processes and they may be unable to serve mitigation purpose in real-time.

In particular circumstances, it is difficult to discriminate between DDoS attacks and flash crowds. Flash crowds are described as the sudden surge of legitimate network traffic which comes from legitimate users of available services. Since DDoS attacks can be crafted to look like a flash crowd, the existing DDoS detection systems can be fooled into letting the flash-crowd attack pass through to the victim(s). Moreover, false detection results can be very risky to available services. In the case of false positives, if the detection system treats the legitimate flash crowds as DDoS attacks, legitimate users are unable to access the available service, which accelerates the DoS condition to services. In the case of false negatives, if the detection system treats the DDoS attack as a high volume of legitimate traffic, then the available services receive undesired traffic as the result of a successful DDoS attacks.
Existing DDoS detection systems lack autonomous learning ability to adjust their detection results. Most detection methods deploy statistical and heuristic approaches to measure the differences between DDoS attacks and legitimate traffics. While, the key to any statistical-based detection system (SBDS) is its ability to learn and distinguish normal from anomalous network activity, the heuristic-based system (HBDS) relies on optimisation of its threshold decision and requires fine-tuning to produce stability, improvement, or precise results of anomaly detection in network traffic to minimise the false positives/negatives. However, these detection methods need to be manually adjusted in order to fine-tune the results.

A major threat to information privacy on web services is traffic analysis attacks. Users of a web service have the right to protect their confidential information from falling into the hands of a third party. Failure to provide information privacy on a web service may result in an undesirable outcome. The aim of traffic analysis attacks is to obtain the content of communications. Adversaries who use traffic analysis attacks usually focus on discovering whether two entities communicate with each other. Moreover, it is a significant challenge to achieve perfect anonymity for web browsing applications. The research on light latency anonymous communication, such as web browsing, is quite limited because of the huge delays and high processing costs caused by the current dominant dummy packet padding strategy.

8.2 Thesis Contribution

The major contribution of this thesis is the development of a set of methods to overcome the problems outlined above. Each method consists of a set of algorithms which have been tested by several sample datasets. We have summarised these
contributions based on the research areas of DDoS attacks detection and web access anonymity.

Chapter 3 proposed DDoS attacks detection by discriminating DDoS attacks from legitimate flash crowds. We performed a similarity measurement of suspicious traffic flows which could be classified as either a DDoS attack flow or flash crowd. We proposed the Jeffrey distance, the Sibson distance, and the Hellinger distance as measurement tools in order to achieve our goal. The main contributions of Chapter 3 are as follows:

- From our simulations, the Sibson distance is the best measurement tools compared to the other two distance metrics for our DDoS discrimination purpose. This is because the Sibson distance is the least sensitive and the most stable metric among the previously mentioned metrics.

- The proposed discrimination method is scalable and practical. The detection method required only two cooperative routers and it can implement cooperating detection in any routers on the Internet, rather than limiting it to only within an ISP network or a community network. This would make it much easier to achieve early detection of a DDoS attack well before it reaches the targeted victim.

- The proposed detection method is not dependent upon any specific type of DDoS attack sources. Therefore, this method can detect any forthcoming easily.

- The proposed detection method was used on real-trace datasets. It proved that it was able to differentiate DDoS flooding attacks from flash crowds with a high rate of accuracy.
Chapter 4 proposed DDoS detection that can discriminate DDoS attacks from legitimate flash crowds. We focused on examining the individual source of suspicious traffic flows using data dependency as a measurement tool. We proposed a set of attack signatures detected by suspicious behaviours such as a repeatable feature of arrival rates from DDoS attack traffic. By using Pearson’s correlation coefficient as a dependency measurement, we matched the signature on DDoS attacks and discriminated them from flash crowds. The overall contributions of this chapter are as follows:

- The proposed detection methods can detect DDoS attacks with a high rate of accuracy. We measured the data dependency of DDoS attack flows and compared this to flash crowds. The results of our experiments showed low false positives and false negatives.

- The detection methods can be implemented in real-world cases based on current Internet technology. With light calculation and low complexity, the proposed methods can feasibly be implemented on many kinds of network equipment such as hubs/switches, firewalls, routers and IDS.

- The detection methods were able to detect DDoS attacks in real-time. With a short period of detection time, the methods benefit the defence system by responding to an action. This could support early DDoS detection before the suspicious flash-crowd traffic arrives at the server.

- The detection methods are independent of methods for attack packet transmission and types of attack packets. We detect repeatable features of packets transmission as DDoS signatures. Hence, the detection methods may be able to handle any form of attack packets such as malformed IP, TCP, UDP, ICMP, or Application-based floods, etc.
Chapter 5 proposed an improved method for DDoS attacks detection. We measure the data dependency and predictability of suspicious traffics by using Pearson’s correlation coefficient and Shannon entropy respectively. The results of these measurements were then transferred to the process of Linear Discriminant Analysis (LDA). Based on previous knowledge of known groups, LDA can discriminate legitimate flash crowds from DDoS attack traffics with a high rate of accuracy. The contributions of this chapter are as follows:

- The proposed DDoS detection system gave us a high rate of accuracy in detection results. By using the triple checks from Pearson’s correlation, Shannon’s entropy and LDA, the statistical relationship of a traffic flow was measured intensively. The detection system maximised the accuracy in detection and minimised the false positive rate (FPR) and false negative rate (FNR) in the results.

- The proposed DDoS detection system was flexible and extensible in implementation. With light calculation and low complexity, the proposed method can be implemented on many kinds of network equipments such as hubs/switches, firewalls, routers, IDS and IPS. We can distribute the coordinate detection to participating network equipments in order to extend the secured border.

- This method is developing for the purposes of supporting early DDoS detection. The detection system may be able implement in any part of the network route from the packet source to the server. This feature supports the implementation of cooperative security networks. As soon as suspicious flash-crowd traffic arrives at the detection sensors, we can raise the alarm to alert
the server in the early stages. This helps the service provider prepare for the proper action to curb the suspicious traffic flows.

- The DDoS detection system is independent from packet forms and transmission strategies. This system can support the examination of any form of attack packets such as malformed IP, TCP, UDP, ICMP, or Application-based floods, etc. It also works well with low-rate attacks, flash-crowd attacks, shrew attacks, periodical attacks, and pulsing attacks.

- We proposed the learning ability of our DDoS detection system. The classification module allows the detection system to adjust its knowledge. Since the decision making module provides the double checks for the accuracy of results, the knowledge from feedback can be reused by the training algorithm. Hence, the knowledge is optimised and the accuracy is maximised.

Chapter 6 propose a novel strategy to improve information privacy on web-based services. We can achieve perfect anonymity based on Shannon’s perfect secrecy theory by covering the original traffic with packet padding. In our proposed methods, we replace the dummy packet strategy using prefetched web pages for padding traffic in order to minimise the cost. We also transformed the anonymity problem into an optimisation trade-off between the anonymity level and the cost of cover traffics. The principal contributions of this chapter are as follows:

- Our novel strategy for packet padding using prefetched data as cover traffic for anonymous web browsing. The proposed strategy allows for the highest level of anonymity in web browsing under low latency network conditions to be reached as well as the reduction of bandwidth wastage and network delays significantly.
A mathematical model has been established to describe and analyse the anonymisation systems. We believe this model can be deployed for further research in the security area.

We transformed the anonymous communication problem into an optimisation problem based on our model. As a result, we can solve trade-off between the anonymity level and the cost of applications. This allows users to deploy the proper anonymity level once the delay constraint is known.

Chapter 7 proposed a novel strategy to improve perfect anonymity in web browsing. We formally established a mathematical model based on Shannon’s perfect secrecy theory for solving the problem of high traffic cost. We also defined a metric to measure the cost of the prefetched data as cover traffic using our method. The main contributions of this chapter are listed as follows.

- We proposed a new strategy for anonymous web browsing which deploys predicted web pages to conduct packet padding for web browsing, instead of dummy packets.
- We established a simple mathematical model based on the perfect secrecy theory for the proposed strategy and defined a metric to measure the cost of web browsing with perfect anonymity.
- Our strategy was proven to essentially reduce the problems of extra delay and demand for extra bandwidth using the traditional dummy packet padding methods.

8.3 Future Directions

Security on availability and information privacy of web services is an urgent global issue. The vision of this thesis was to develop and improve defence systems, models
and architectures against DDoS attack and traffic analysis attack. These attacks are actively for searching for new approaches against defence mechanisms and systems. The following are areas that could be improved and extended in future research.

− The perfect accuracy in DDoS detection is far from a reality. Leaning the truth and the fact of data is the key to minimising false detection. We can also improve the accuracy of the flow based discrimination strategy with more side information, such as other independent attack features, and extending the experiments on a large scale to observe the performance of the discrimination algorithm. In addition, we are also looking forward to improving the accuracy of discrimination with double checks being used in the decision making module as described in chapter 5. This could provide us with more accuracy however we may not implement this additional module in real-time detection.

− A large number of observing data may cause delay in DDoS detection. This delay increases the chance of the victim to crash due to a large number of incoming packets. The problem was that we used the time slot for observing sample data. Thus, we rely on the time domain rather than the number of packets. Therefore, we need to improve our proposed methods for detecting the repeatable features of other packet information such as packet delay.

− The detection time is so much more important for the DDoS defence system. If the processes of DDoS detection take too long, the stream packets may be delayed at the defence system. This means the legitimated clients may be waiting for a period of time for a response messages.

− We need to explore our methods with some challenges, such as DDoS attacks with spoofing IP address. In chapter 4 and 5, our methods use an individual check for the IP address of a possible attack source. IP spoofing may confuse
our detection system by creating many random IP addresses. Instead of observing behaviour from regular clients, we may confront a number of strange new clients. Therefore, through observation of the changing rate of a port number, a possible solution could be provided for us.

- Extracting the repeatable feature of attack source is not an easy job as complexity is an issue that may cause other problems such as delay and implementation. If the detection mechanism is very complicated, the system may be delayed to achieve a result. On the other hand, the simple method cannot describe the attack phenomenon very well. Finding the balance between delay and complexity is one of the challenges in DDoS detection.

- The DDoS detection system should be able to achieve easy implementation and use. To employ a DDoS defence system in an organisation, it must have a high probability of success. Otherwise, it becomes unnecessarily expensive technology. In addition, our DDoS detection models are not too complex at this stage. However, to cover various methods of DDoS attack, we need to improve the complexity, but not too much that makes implementation difficult. We believe that we can insert complexity in some cases of DDoS attack to achieve accurate results.

- We believe our methods could be implemented on any network equipment such as switches, routers, firewalls, IDS, and so forth. This increases the chance of detecting the DDoS attack at an earlier stage before reaching the victim.

- In Chapter 5, the full learning algorithm must be improved based on the feedback from measurement and making decision modules. This gives us a
smart DDoS detection system with its AI having the capacity to improve its accuracy and configuration process by itself.

− In Chapter 5, we expect to extend the model to multiple user scenarios, and include link padding with multiple uses. Web prefetching algorithms will be integrated into the model, and extensive tests against the data set are expected in the near future. More effort is desired in the process of de-anonymisation at the client side for efficiency and accuracy.

− In real applications on the Internet, it is extraordinarily expensive to achieve perfect anonymity, therefore, alternative solutions are desperately needed. Packet dropping is an interesting strategy to optimise and reduce cost.

− Relative anonymisation is an interesting method to improve web access anonymity. In some cases, perfect anonymity may not be necessary, as users only expect some level of anonymity for their web browsing. Moreover, the adversary may not have complete observation of the monitored users, and therefore, an adaptive method may be introduced to further reduce the cost of anonymisation. We believe Game Theory can play a great role in this direction, for example, finding the boundaries of anonymisation cost against a given anonymity level.

− Link padding has to be considered in the framework, which will reduce the cost significantly. It also poses a great challenge for achieving anonymity of web browsing.

− The trade-off between defence mechanisms of a DDoS attack and traffic analysis needs to be negotiated. Since we improved web access anonymity against traffic analysis attack, the encrypted HTTP communication traffic has also been covered. Perfect anonymity will secure the privacy of user
information from traffic analysis attacks, but this fails to apply for DDoS attack detection. This is because DDoS detection also requires the performance of traffic analysis such as observation of the pattern of traffic requests. We therefore need to establish a scheme of negotiation that allows the traffic analysis performed to detect DDoS attack.
REFERENCE


