NAIST-IS-DD1161207

Doctoral Dissertation

Unsupervised Anomaly Detection in Massive Traffic Using S-transform and Rényi Divergence

Sirikarn Pukkawanna

June 18, 2015

Department of Information Systems
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to the Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Sirikarn Pukkawanna

Thesis Committee:

Professor Suguru YAMAGUCHI	(Supervisor)
Professor Keiichi YASUMOTO	(Co-supervisor)
Professor Kazushi IKEDA	(Co-supervisor)
Professor Hiroyuki SEKI	(Co-supervisor, Nagoya University)
Associate Professor Youki KADOBAYASHI	(Co-Supervisor)

Unsupervised Anomaly Detection in Massive Traffic Using S-transform and Rényi Divergence*

Sirikarn Pukkawanna

Abstract

The detection of network anomalies is an indispensable component of overall security architecture. As sophisticated attacks grow exponentially, preserving security with signature-based Network Intrusion Detection Systems (NIDS) may not be sufficient because they cannot detect new and unknown attacks. Furthermore, in order to obtain good performance from a signature-based NIDS, a network administrator has to essentially keep updating new accurate signatures to the NIDS's signature database. In this dissertation, we propose two novel network anomaly detection methods: S-transform-based and Rényi divergence-based methods. Both methods do not require the pre-defined signatures of targets and are able to detect unknown and new malicious and disruptive traffic with high accuracy and low false positive rates. The methods' targets include malicious traffic caused by Denial-of-Service (DoS) attacks, Internet worms, scannings, and legitimate traffic that is likely to disrupt networks or devices.

This dissertation consists of two main parts. The first part presents the S-transform-based anomaly detection method. Our method uses S-transform to convert a traffic signal (e.g., packet rate) to a time-frequency domain. The method then detects unusual time-frequency behavior caused by anomalies in the time-frequency domain. The major advantage of our method is that it can detect hidden anomalies that cannot be easily seen in the traffic signal (time domain data). We evaluated our method with simulated traffic from the DARPA dataset and real-world backbone traffic from the MAWI dataset. Furthermore, we compared the performance of our method with a popular Wavelet transform-based

*Doctoral Dissertation, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD1161207, June 18, 2015.

anomaly detection method. The results indicated that our method outperformed the Wavelet transform-based method for detecting the anomalies in both datasets in terms of both accuracy and false positive rates. At the end of this part, we propose a new idea to improve the detection performance of anomaly detection methods using sketch technique. We improved our S-transform-based method according to the idea and verified the idea by testing the improved method with a large amount of real-world traffic from the MAWI dataset and real-world traffic from the ISOT dataset. The results indicated that our improved method provided higher accuracy and lower false positive rates.

The second part of this dissertation introduces a new statistical traffic feature, which is the port pair distribution of traffic flows for network anomaly detection. This part then describes the Rényi divergence-based anomaly detection method that observes the port pair distribution of traffic and detects anomalies based on the Rényi divergence of the port pair distributions. We evaluated the performance of our proposed feature by testing it with real-world backbone traffic from the MAWI dataset and by comparing it with four widely-used traffic features, namely the distributions of source IP, destination IP, source port, and destination port. The results indicated that our feature completely outperformed the four widely-used features in terms of both accuracy and false positive rates. Lastly, using the same traffic, we compared the performance of our method with a Kullback-Leibler (KL) divergence-based anomaly detection method. The results indicated that our method could detect anomalies with 96% accuracy and was more effective than the KL divergence-based method.

Keywords:

Anomaly detection, outlier detection, time-frequency analysis, S-transform, sketch, random aggregation, divergence measures, Rényi divergence, α-divergence

Contents

List of Figures

List of Tables

Acknowledgements

First of all, I would like to thank Professor Suguru Yamaguchi, who gave me an important opportunity to study in Japan and be a member of his excellent laboratory. He literally inspired me with the phrase, "work hard, play hard". His suggestions and comments were always clear, direct, and valuable to my research. Furthermore, his simple comment to me in the first year of my study "You are important for our laboratory" kept me warm like being home, made me less homesick, and allowed me to smoothly concentrate on my study. I would like to thank my thesis committee, Professor Keiichi Yasumoto, Professor Kazushi Ikeda, and Professor Hiroyuki Seki for their kind advice and helpful comments for the improvement of this dissertation. Their invaluable questions, especially during my midterm and public defense presentation, are highly appreciated. I would like to especially thank Associate Professor Youki Kadobayashi, who was my closest advisor over the past three years, for all the time, knowledge, and creative ideas he dedicated to me and my research. His suggestions and comments were like diamonds. He also simultaneously gave me the freedom to do my research in my own way. He gave me a tremendous of opportunities to gain relevant research experiences as well as intentionally toughened me up by bringing me out of my comfort zone so that I could grow as a person. I would like to thank Assistant Professor Vasaka Visootitviseth at Mahidol University for kindly suggesting the Monbukagakusho Scholarship to me so that I could partake in the opportunity to study at NAIST. I would like to thank Associate Professor Hiroaki Hazeyama, Associate Professor Takeshi Okuda, and Assistant Professor Shigeru Kashihara for their useful suggestions and comments on my work. I would like to thank all the secretaries in my laboratory, namely Natsue Tanida, Ami Ugou, Yukika Nishitouge, and Naoko Omori, for their always kind support and help. I would like

to thank the Thai students in my university and my international friends in my laboratory, especially Noppawat Chaisamran, Rubwad Mathuranyanon, Onuma Chumsakul, Jane Louie Fresco Zamora, Christopher Michael Yap, Doudou Fall, Marius Liviu Georgescu, Sybile Nivon, Vincent Danché, Ady Wahyudi Paundu, and Jema Devid Ndibwire, for their friendship, support, help, and encouragement. Without a doubt, I would like to thank all of my Japanese friends in my laboratory for being nice friends. I would like to thank Gregory Blanc for taking care of me and being a nice colleague and friend during my internship in France. I would like to thank Kopchai Saisingthong, who is one of my best friends, for his encouraging words over the years. Most importantly, I would like to thank my family, especially my mother Obtip Pukkawanna. She is the one who genuinely brings me to this point and allows me to reach this achievement in my life. Without unconditional love and support from my family members, I can say that I will not be able to succeed in the past three years. Finally, I gratefully acknowledge the financial support of the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan through Monbukagakusho Scholarship.

Chapter 1

Introduction

Today the Internet is one of the most important mediums of communication. Billions of wired and wireless devices are connected to the Internet and communicate with each other, such as mobile phones, personal computers, database servers on enterprise networks, traffic cameras, measurement sensors, and automobiles. This powerful technology truly brings convenience to human beings. At the same time, this technology provides easy access to target devices and networks for intruders and attackers as well.

To protect devices and networks from hazards caused by intruders and attackers, especially from the Internet and untrusted networks, placing a firewall in between local networks and untrusted networks is a primitive method to filter incoming traffic and block unauthorized accesses while permitting authorized communications. Although a firewall provides crucial protection for a network, it is not aimed at detecting intrusions and attacks. Therefore, to reveal intrusions and attacks, a network administrator requires a Network Intrusion Detection System (NIDS).

More specifically, a NIDS is a software or hardware device designed to monitor and analyze traffic data in order to detect intrusion attempts to a monitoring network, such as Denial-of-Service (DoS) attacks, Internet worms, and viruses. When the NIDS faces an attack, it generates an alert to inform the network administrator. In general, NIDSs are categorized into two types: misuse-based NIDS (also traditionally known as signature-based NIDS) and anomaly-based NIDS. The misuse-based NIDS relies on pre-defined attack signatures (e.g., key-

word strings of malicious codes) and uses those pre-defined signatures to catch known attacks. A well-known misuse-based NIDS is Snort [1], which offers network administrators a large set of signatures to detect various kinds of known attacks. A major advantage of misuse-based NIDSs is that they efficiently detect attacks that they are familiar with. However, misuse-based NIDSs still have a limitation in that they are not able to detect new or zero-day attacks. Furthermore, it is critical that signatures are accurate and signature databases need to be up-to-date in order to obtain good detection performance.

Due to the limitations of misuse-based NIDSs, anomaly-based NIDSs were introduced [2]. To detect network anomalies, an anomaly-based NIDS does not use a set of pre-defined signatures. Instead, the anomaly-based NIDS looks for activities (events) that do not conform to an expected pattern [3]. In other words, an anomaly-based NIDS finds outliers. As a result, anomaly-based NIDSs have the ability to detect both known and zero-day attacks.

In this dissertation, we show the existing problems and challenges of network anomaly detection. Existing network anomaly detection methods, including their advantages and disadvantages, are also discussed in this dissertation. Lastly, we introduce our two proposed anomaly detection methods.

The rest of this chapter gives overviews of network anomalies and anomaly detection problems and challenges that motivate us to develop novel anomaly detection methods. We then discuss the key contributions of this dissertation. The end of this chapter gives the outline of the remaining chapters in this dissertation.

1.1 Network Anomalies

As shown in Figure 1.1, network activities can be categorized according to their aims into two types such as legitimate and malicious. A legitimate activity is an activity of a legitimate user or device. For example, an authorized student checks her mailbox via a campus network. This kind of activity is frequently benign, but sometimes it may disrupt or overwhelm a network or device if the traffic of that activity is excessive. Therefore, any legitimate activity that obstructs other legitimate activities should be noticed by network administrators. For example, a network administrator should know if there are simultaneous requests from a

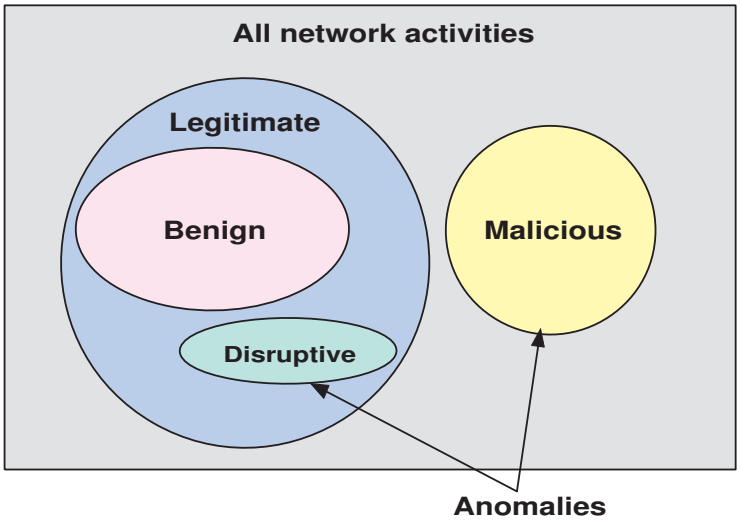

Figure 1.1: Types of network activities

tremendous amount of legitimate users to a server in a very short amount of time. A malicious activity is a harmful action that is done on purpose by an attacker or compromised machine. This kind of activity aims to acquire, destroy, modify, or access a network, machine, or user's data without permission. For example, a bot, which is a compromised machine, scans a network to find vulnerable machines. All malicious activities should be certainly detected.

In 2009, Chandola et al. [3] stated that anomalies are patterns in data that do not conform to a well-defined notion of normal behavior. Malicious and disruptive activities tend to behave differently from normal activities. Therefore, a disruptive activity is also considered a network anomaly, even it is legitimate. In this section, we list network anomalies that are still posing threats and should be detected by NIDSs.

1.1.1 Malicious Activities

DoS/DDoS Attacks A DoS attack is a malicious attempt to make a machine unavailable to its intended users, usually by temporarily interrupting or suspending services of a machine connected to the Internet. The most common type of DoS attack involves flooding a victim with external communication requests.

This overload prevents the victim's resources from responding to legitimate traffic or slows its response. For Distributed DoS (DDoS) attacks, they have the same aim with DoS attacks, but involve a group of attackers to increase the strength of the attack. Even though DoS attack technology is not a new technology, it continues to evolve and remains a serious threat to users, organizations, and infrastructures of the Internet [4]. For example, in late 2014, CNET revealed that anonymous hackers used DDoS attacks to take down Sony's PlayStation network and Microsoft Xbox live [5]. Furthermore, new DoS attacks are continuously invented to evade detection [6]. For example, new volume-based DoS attacks took down many game sites [7].

- *TCP SYN flood* is one of classic DoS attacks that exploits the vulnerability of the TCP three-way handshake. The aim of SYN flood attacks is to occupying the memory resources of a victim by abruptly sending an excessive number of TCP SYN requests with a spoofed source IP. The victim replies with SYN-ACK packets, but never get responses because the source IP is forged. Due to those incomplete three-way handshakes, the victim's memory resources may run out (usually depend on the operating system) and then the victim is unable to accept legitimate connection requests for a while.

- *ICMP flood* is a flooding attack similar to a SYN flood attack. It takes place when an attacker overloads its victim with a huge number of ICMP echo requests with a spoofed source IP. This type of attack still causes a lot of headaches to network administrators.

- *Smurf attack* is a DDoS attack that involves a number of fooled attackers and ICMP packets. In a Smurf attack, a real Smurf attacker with a spoofed IP sends ICMP ECHO request packets to an IP broadcast address of a target network. Technically, all active hosts in that network that receive the ICMP ECHO request packets will reply to the fake IP which is the IP of the actual target victim. This huge packet stream overloads the victim's resources.

- *UDP flood* does not differ from TCP SYN and ICMP floods. The difference

is that the IP packets that the attacker uses against its victim contain UDP datagrams of different sizes.

- *Teardrop* is a type of DoS attack that deals with fragmentation and reassembly of IP packets. Basically, an IP header contains a necessary field named Fragment offset indicating the starting position of each fragment relative to the original packet. To perform a teardrop attack, an attacker starts transmitting fragmented IP packets containing overlapped fragment offsets making a victim unable to reassemble those fragmented packets, exhausting the victims resources, and possibly crashing it.

Probe Attacks In general, a probe attack aims to acquire information about a target network or host by scanning. This attack is usually performed before launching a real and dangerous attack (e.g., DDoS attacks). This attack occurs frequently on the Internet [8] because it is usually an essential step before an actual attack. Furthermore, tools for generating these attacks are easy-to-use and widely available on the Internet.

- *Port scan* is basically used to discover vulnerable services being provided by a host. In other words, a port scan checks which ports can be a hole to access to a target in order to compromise it. A port scan usually involves a wide range of port numbers of a target.

- *Network scan* intentionally investigates a network in order to find hosts running a specific service. For example, if an attacker wants to find a web server in a network, the attacker sends packets to port 80 (HTTP) of all hosts in that network. Technically, hosts that responds with accepting messages are likely to be web servers.

Spreading of Malicious Codes

- *Worms and viruses* are self-replicating programs. A virus is a program that inserts a possibly evolved copy of itself into a host, including operating systems, to propagate. Unlike a virus that requires a human to activate it, a worm propagates without any human intervention by exploiting a software

vulnerability on remote hosts [9]. Because the Internet worms do not need human interaction, the Internet worms spread over many vulnerable hosts in a very short time. For example, the Code Red worm infected 359,000 hosts in less than 14 hours [10]. The Slammer worm infected 90% of vulnerable hosts within 10 minutes. Even though many efforts have been made to prevent Internet worms and viruses, these threats do not seem to diminish as they were discovered continuously. For example, the Sasser was found in 2004, the Storm was found in 2007, the Sluxnet and Duquin were found in 2010, and the Crazy Shellshock was recently found in 2014.

- *E-mail spam* is one of the greatest threats to the use of e-mail. The severity of spams goes far beyond mere inconvenience. E-mail spams are also used for more malicious purposes such as scam distribution, personal information collection, and virus propagation.

R2L Attack A Remote to Local (R2L) attack is used when an attacker wants to gain local access to a machine over network. Specifically, an attacker sends packets to a victim's machine and then exploits some vulnerabilities in the victim's machine. This attack is difficult to detect with anomaly-based NIDSs because the attack generates a small number of packets to the network.

U2R Attack A User to Root (U2R) attack starts with access to a normal user account on a target system. The attacker acts as the normal user to exploit the vulnerabilities in order to gain root access to the system. To obtain the information about a normal user account in a target system, the attacker may perform a password sniffing, dictionary attack, or social engineering attack. This attack is also difficult to detect with anomaly-based NIDSs because the attack generates few packets to the network.

1.1.2 Legitimate and Disruptive Activities

As we mentioned above, legitimate activities can occasionally disrupt or cause trouble to networks or devices. Unfortunately, network administrators rarely have proactive measures against these anomalous activities. In general, a network

administrator will know that an anomaly occurs in her network when the anomaly already affects her network. Below we list some legitimate activities that have negative effects to networks or devices.

- *Flash crowd* [11] is an activity in which a huge number of hosts create excessive connections to unintentionally overwhelm a server. For example, many users try to download the same file from a website at the same time. This legitimate activity can make the website crash.

- *Device misconfiguration and malfunction* are common events that can happen on every network, especially on networks consisting of several heterogeneous devices. These events should be considered and noticed by network administrators because the traffic from these misconfigured or mulfunctioning devices may unintentionally affect the networks where the devices connect to, other devices, or services. For example, a misconfigured router may generate a large number of routing table advertisements to the network. Devices that receive those advertisements may be unaccessible due to over-consumption of memory.

- *Network failure and equipment outage* are events that frequently exist in today's Internet [12]. These events should also be noticed by network administrators as quickly as possible. A failed link or equipment outage (e,g., router reboots) may dramatically impact packet forwarding and services.

1.2 Anomaly Detection, Problems, and Challenges

1.2.1 Anomaly Detection

In data mining, anomaly detection or outlier detection is the identification of items, events or observations which do not conform to an expected pattern or other items in a dataset [3]. In signal and image processing, anomaly detection is characterized by the desire to locate and identify uncommon features in an image [13]. To date, anomaly detection is applied in a variety of domains and

used to analyze many kind of data such as geographic [13], weather , vehicle [14], and network data.

In general, anomaly detection consists of two main steps: (1) finding a baseline representing the normality of data and (2) using the baseline to distinguish anomalies from normal instances in the data. The first step is a very important and challenging task because an obtained baseline genuinely impacts detection performance. If the baseline is precise, it produces good results in detection. On the other hand, if the baseline is inaccurate, it produces a low detection rate. The baseline can be a suitable value, model, or classifier depending on the technique used in an anomaly detection method. Basically, a baseline can be obtained in three approaches. The first approach is by learning a pre-labeled traffic dataset (training dataset) in which all instances were labeled as either normal or attack. The labels can be other terms such as legitimate and malicious. The approach then creates a baseline classifier based on the labeled instances in the dataset. This learning is performed in advance (offline learning) before a real detection. At the time of detection, the classifier is used to designate which traffic instance is normal which traffic instance is anomalous. Anomaly detection methods that use fully-labeled datasets for creating baselines are called supervised anomaly detection methods. The second approach creates a baseline based on a partially-labeled dataset in which only normal instances were labeled. Anomaly detection methods that rely on partially-labeled datasets are called semi-supervised anomaly detection methods [3]. The third approach to obtain a baseline does not use a pre-labeled dataset. The approach creates a baseline based on the assumption that the main behavior of instances in data represents the normal behavior of the data. Therefore, the approach finds instances that do not conform to the main behavior. Anomaly detection methods that create baselines based on unlabeled datasets and this assumption are called unsupervised anomaly detection methods.

Based on the nature of anomaly detection, researchers are unable to strongly specify what types of attacks (e.g., DoS attacks) can be detected by their methods. However, the researchers can intuitively guarantee that all activities having abnormal features will be detected.

Since the potency of anomaly detection does not require prior knowledge

Methods	Advantages	Disadvantages
Statistical methods	- Lightweight (less storage) - Can detect anomalies in real-time	- Performance depends on observed features
Data mining-based methods	- Very good at detecting seen anomalies	- Performance depends on training datasets
TFA-based methods	- Can detect anomalies that cannot be detected by statistical and data-mining-based methods	- Hard to detect anomalies in real-time

Table 1.1: Advantages and disadvantages of anomaly detection methods

of targets, many anomaly detection methods have been proposed for detecting network anomalies such as statistical, data mining-based, and Time-Frequency Analysis (TFA)-based methods. Below we briefly summarize the detection procedures of each method. The advantages and disadvantages of each method are summarized in Table 1.1. We note that the insight survey on existing anomaly detection methods will be described in the next chapter.

Statistical Methods Generally, researchers who develop statistical anomaly detection methods are interested in statistical properties of traffic such as the mean of flow sizes, the ratio of incoming traffic and outgoing traffic, and IP distribution. To detect anomalies, a statistical anomaly detection method observes a statistical property and assumes that a change of the statistical property is caused by an anomaly. The method thus tries to find change points in monitoring statistical data. Many techniques are applied in order to detect change points, namely regression model [15], Kullback-Leibler (KL) divergence [16–21], and Principle Component Analysis (PCA) [22–26]. The major advantage of statistical methods is that they are lightweight in terms of storage. Furthermore, the statistical methods are promising with regard to performing real-time detection. However, the detection performance of statistical methods strongly depends on statistical properties being observed.

Data Mining-based Methods Data mining-based anomaly detection methods are mostly related to learning raw packet or connection attributes (such as source port, destination port, source IP, destination IP, protocol type, TCP flag, and connection size) for creating classifiers. Examples of data mining techniques used for learning such as clustering [27, 28], Neural Network (NN), and Support Vector Machine (SVM) [29] techniques. Most data mining-based methods are supervised or semi-supervised methods that require pre-labeled training datasets. As a result, data mining-based methods have a strong advantage that they efficiently detect anomalies on networks in which training datasets are collected. Unfortunately, the detection performance of supervised and semi-supervised data mining-based methods depends on the quality of used pre-labeled training datasets.

Time-Frequency Analysis-based Methods Time-Frequency Analysis (TFA)-based anomaly detection methods are also associated with traffic statistics like statistical methods, but the TFA-based methods analyze information in the time-frequency domain of a traffic statistical data instead of in a time domain like statistical methods. Specifically, a TFA-based method firstly observes a traffic statistic (e.g., packet rate). The observed traffic statistic is considered a time domain signal. The TFA-based method then transforms the signal into a time-frequency domain and detects abnormalities that are present in the time-frequency domain. A popular Time-Frequency Representation (TFR) technique used to transform a traffic signal to a time-frequency domain is Wavelet transform [30–36]. The advantage of TFA-based anomaly detection methods is that they can detect various types of anomalies that cannot be seen by time domain analysis-based methods. However, TFA-based methods are not probably suitable for real-time anomaly detection due to complexity.

1.2.2 Problems and Challenges in Anomaly Detection

There are the following issues that make anomaly detection very challenging. Table 1.2 describes the problems and challenges of existing anomaly detection methods.

- *The need of pre-labeled datasets of supervised and semi-supervised methods*

Issue	Most statistical methods	Most data mining-based methods	Most TFA-based methods
Pre-labeled datasets	- Not required	- Required - Performance depends on pre-labeled datasets	- Not required
#false alarms	- Higher than data mining-based methods - Depend on used techniques	- Low (frequently)	- Higher than data mining-based methods - Depend on used techniques
Features	- Affect performance	- Affect performance	- Affect performance
Real-time detection	- High possibility - Depend on used techniques	- High possibility with non-complex classifiers	- Possible - Depend on used techniques
Anomaly identification	- Not provided yet - Need root cause analysis	- Easy	- Not provided yet - Require root cause analysis
Mimicry anomalies	- Hard to detect	- Hard to detect	- Hard to detect

Table 1.2: Problems and challenges of anomaly detection methods

Researchers proposed supervised and semi-supervised anomaly detection methods to address the limitation of misuse-based detection methods (used by misuse-based NIDSs) that they are limited to pre-defined attack signatures. Unfortunately, both supervised and semi-supervised methods still require at least a labeled traffic dataset for creating a traffic classifier. The problem is that there are only a few available labeled traffic datasets so far. The reason is that labeling data requires a security expert. Even though researchers can find a labeled dataset deemed adequate, the classifier obtained from that labeled dataset may be not sufficient for detecting anomalies in different network circumstances. Furthermore, if the labeled dataset is an old-fashioned dataset, the obtained classifier is not effective for detecting new anomalies. In other words, the detection performance of supervised and semi-supervised anomaly detection methods depend on pre-labeled training datasets. In summary, supervised and semi-supervised anomaly detection methods have issues with finding a perfect labeled dataset and detecting unfamiliar anomalies in unfamiliar networks. Most existing data mining-based anomaly detection methods operate in supervised or semi-supervised mode. Therefore, most data mining-based methods are facing the same problems and challenges.

- *High number of false alarms* A false alarm is an event occurred when an anomaly detection method incorrectly classifies a benign activity as an

anomalous activity. Supervised and semi-supervised methods suffer less false alarms than unsupervised methods. The reason is that supervised and semi-supervised have some knowledge of benign traffic (from labeled training datasets) to help to correctly distinguish benign activities from anomalous activities. On the other hand, unsupervised methods do not have any idea of benign traffic. The unsupervised methods intuitively detect any traffic behaving differently from the main behavior of data. As a result, unsupervised anomaly detection methods may produce false alarms more frequently than supervised and semi-supervised methods. For example, an unsupervised method may be triggered by benign traffic from a new application or device that just connects to the network. Even though unsupervised anomaly detection methods offer the capability to detect new and unknown anomalies without pre-labeled training datasets, they still face the high false alarm problem. To the best of our knowledge, statistical and TFA-based anomaly detection methods, which mostly operate in unsupervised mode, are also facing this problem.

- *Finding relevant traffic features* One of the most important tasks for developing anomaly detection methods is selecting traffic features, which can be used to distinguish anomalous traffic from normal traffic. Good features should vary significantly between normal and anomalous behaviors. Regarding the effect of features on the detection performance of each type of anomaly detection methods, statistical methods are unable to see any anomalies if the statistical features being observed are not significant. For data mining-based methods, if selected raw features are not relevant, the methods may not be able to find any pattern or the methods may find useless patterns. Similarly, TFA-based methods need a good traffic signal input (statistical feature) because a good signal is a good basis for TFA. A good signal should preserve the behavior of anomalies in both time and time-frequency domains so that a TFA-based method can see and detect anomalies in both domains. To date, there are a number of potential traffic features proposed. Some researchers use up to 200 traffic features. However, there is no single traffic feature that can be used to detect all kinds of anomalies in the world. Therefore, this is a very challenging task for

researchers who have to find relevant traffic features to be able to detect as many as anomalies.

- *Real-time detection capability* Real-time detection is a crucial property that NIDSs must have because it does not matter if an anomaly is not detected at the right time. In other words, if an anomaly is detected at its beginning phase, it is more likely that our networks or devices are still safe or a countermeasure can be deployed in time. In general, a statistical anomaly detection method simple monitors a statistical feature and detects feature changes. The real-time anomaly detection capability of statistical methods depends on the decision making techniques that are used. For example, if a statistical method monitors the packet rate of traffic and detects changes by comparing the current packet rate with a value, this kind of statistical method can detect anomalies in real-time. On the other hand, some statistical methods need to wait for a set of new traffic instances in order to be able to precisely designate anomalous instances. This kind of method detects anomalies with a slight delay. Examples include PCA-based and KL-based methods. A data mining-based method obtains a classifier in advance. The data mining-based method then detects anomalies by simply passing new traffic to the classifier. Therefore, most data mining-based methods with non-complex classifiers can detect anomalies in real-time. A TFA-based method requires a process to transform a signal in order to discover the time-frequency behavior of the signal. Most TFA-based methods require a traffic signal that is large enough so that the methods can see the normal and abnormal patterns of traffic. Therefore, most TFA-based methods report anomalies with some delay similar to some statistical methods. Lately, researchers try to improve the detection performance of their methods by applying or combining sophisticated and effective techniques from several different domains. For example, Fontugne et al. [37] employ an image processing technique to find patterns in a traffic image. Some researchers combine sketch technique with PCA [26]. The sketch technique is also combined with wavelet analysis [38, 39]. These sophisticated techniques make the methods more complicated. Moreover, these sophisticated techniques take more time for calculation and analysis. Therefore, this

is a challenging task for anomaly detection methods to detect anomalies efficiently and quickly at the same time.

- *Anomaly identification capability* Most anomaly detection methods, especially statistical and TFA-based methods, only raise an alarm when a condition is true (e.g., a monitoring value exceeds a threshold). In other words, most statistical and TFA-based methods alert only at times that they think an anomaly exists. Unfortunately, these methods do not provide key information about detected anomalies that are useful for countermeasures such as the port number or IP of detected anomalies. To identify an anomaly, statistical and TFA-base methods require a root cause analysis. For example, after an anomalous time interval is detected, a method investigates all instances in the anomalous time interval to find culprits of the anomaly. Even though some anomaly detection methods provide a set of possible anomalous traffic, but it is still difficult for network administrators to perform precise countermeasures.

- *Mimicry anomalies* Mimicry traffic is anomalous traffic that imitates normal traffic [40]. As anomaly detection methods look for unusual activities that do not conform to expected activities, it is likely to be that anomaly detection methods will ignore mimicry traffic. This is very challenging for all anomaly detection methods and not easy to solve.

1.3 Contributions

This dissertation has four main contributions outlined as follows.

1. The first contribution is the introduction of S-transform to be a valuable TFR tool for network anomaly detection. This contribution also includes the development of an S-transform-based anomaly detection method, which addresses the limitations of misuse-based attack detection methods and supervised and semi-supervised anomaly detection methods. This S-transform-based method does not require prior knowledge of targets, such as attack signatures and pre-labeled training datasets. This method can also detect hidden anomalies that cannot be easily seen by visual inspection

and time domain analysis. Furthermore, this S-transform-based method can better detect anomalies than a popular Wavelet transform-based anomaly detection method. (related to Chapter 3)

2. The second contribution is the improvement of the S-transform-based anomaly detection method using sketch technique. The improved S-transform-based method provides higher accuracy and lower false positive rates than the old S-transform-based method that is not enhanced by the sketch technique. (related to Chapter 3)

3. The third contribution is the introduction of a new statistical traffic feature named port pair distribution for anomaly detection methods. By observing only the port pair distribution, we can better detect anomalies than observing four widely-used statistical traffic features, namely source IP distribution, destination IP distribution, source port distribution, and destination port distribution. Apart from the fact that the port pair distribution feature can provide better detection, it helps to eliminate redundant computation. (related to Chapter 4)

4. The fourth contribution is the development of a Rényi divergence-based anomaly detection method, which uses the port pair distribution feature and Rényi divergence to detect anomalies without requiring of the attack signatures and pre-labeled training datasets. By testing the method with a real-world backbone trace, the method outperforms the Kullback-Leibler divergence-based anomaly detection method. The major advantage of this method is that it reports the 5-tuple information about a detected anomalous flow, namely source IP, destination IP, source port, destination port, and protocol type. Furthermore, the method is a promising method for detecting anomalies in real-time. (related to Chapter 4)

1.4 Dissertation Organization

The rest of this dissertation is organized as follows. Chapter 2 describes general types and desirable properties of anomaly detection methods. Chapter 2 also describes performance evaluation among network anomaly detection methods in

general and historical traffic datasets typically used in intrusion and anomaly detection domains. Furthermore, Chapter 2 describes existing network anomaly detection methods in detail. Chapter 3 describes the proposed S-transform-based anomaly detection method and performance evaluation. Chapter 3 also describes the improvement of the S-transform-based method using the sketch technique and the performance evaluation of the improved S-transform-based method. The effect of the improved method's parameters on detection performance is also presented in Chapter 3. Chapter 4 describes the proposed statistical feature named port pair distribution and Rényi divergence-based anomaly detection method. Furthermore, Chapter 4 shows the performance evaluation of the proposed feature and Rényi divergence-based anomaly detection method. Chapter 5 presents the conclusion of the dissertation and the recommendations for future work.

Chapter 2

A Survey on Network Anomaly Detection Methods

In this chapter, we describe the general types of anomaly detection methods and the advantages and disadvantages of each type. We also describe the important properties that anomaly detection methods must have. Performance comparison among anomaly detection methods, evaluation metrics, and widely-used evaluation traffic datasets are presented in this chapter. In particular, we describe some existing anomaly detection methods in detail, namely statistical, data mining, and TFA-based methods.

2.1 Types of Methods

Based on the characteristics of traffic datasets used to create baselines, anomaly detection methods can be categorized into three types, namely supervised, semi-supervised, and unsupervised.

Supervised A supervised anomaly detection method trusts traffic information on the past. The method uses a set of historical traffic captured at a network in order to learn and create a classifier. The set of the historical traffic is typically called training dataset. Specifically, all instances in the training dataset must be marked as either normal or attack as shown in Figure 2.1. This labeling process can be either manually performed by a security expert or automatically

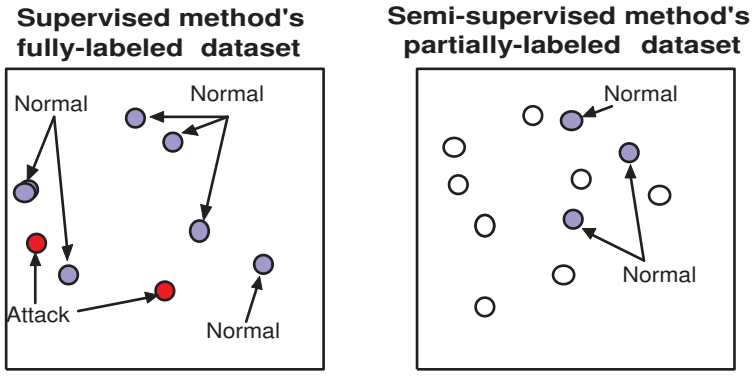

Figure 2.1: Characteristics of pre-labeled training datasets of supervised (left) and semi-supervised (right) methods

performed by a tool. The classifier will be created only one time from the fully-labeled training dataset. This process is operated offline. The method then uses the obtained classifier to analyze and classify new traffic into three classes: normal, attack, and unknown.

An anomaly detection method that falls into this category is suitable to be applied to detect anomalies in the network that the training dataset is collected. If the classifier is used to detect anomalies in a different network, the classifier tends to classify new traffic as unknown because the behavior of normal traffic in the investigated network may be different from the normal traffic in the training dataset. Supervised methods have the following advantages. A supervised method creates a classifier only one time. Therefore, in the time of detection, the method simply passes new traffic to the classifier. The method can also precisely detect expected anomalies. In addition, the method is likely to produce less false alarms in case the training dataset was collected at the same network. The disadvantage of supervised methods is that their detection performances depend on the quality of labeled-training datasets. Furthermore, the supervised methods may miss new and unknown anomalies that have the behaviors differ from anomalies in training datasets.

Semi-supervised A semi-supervised anomaly detection method similarly uses a pre-labeled training dataset for creating a classifier. However, only normal

Method type	Advantages	Disadvantages
Supervised	- Very good to detect familiar anomalies in familiar networks - Low number of false alarms - Create a classifier only once	- Require a fully-labeled dataset - Performance is degraded in different networks
Semi-supervised	- Require only a partially-labeled dataset - Low number of false alarms - Better detect unknown/new anomalies - Create a classifier only once	- Require a partially-labeled dataset - Performance is degraded in different networks
Unsupervised	- Do not require any labeled dataset - Best to detect unknown/new anomalies	- High number of false alarms

Table 2.1: Advantages and disadvantages of supervised, semi-supervised, and unsupervised anomaly detection methods

instances in the dataset are labeled as shown in Figure 2.1. This means that the classifier classifies new traffic into only two classes: normal and abnormal. Any traffic that behaves differently from the normal traffic in the training dataset is basically classified as abnormal traffic.

The semi-supervised method has the advantages similar to a supervised method are that the semi-supervised method learns a partially-labeled dataset only once and then can permanently use the classifier for detecting anomalies. Another advantage of the semi-supervised method is that the semi-supervised method can detect new and unknown anomalies better than a supervised method. It is because the semi-supervised method detects anomalies without any idea about the characteristics of anomalies.

Unsupervised An unsupervised anomaly detection method was proposed to address the problem that pre-labeled training datasets are rarely available. To detect anomalies, an unsupervised method makes the following assumptions.

1. Anomalous instances in data are rare when compared to normal instances

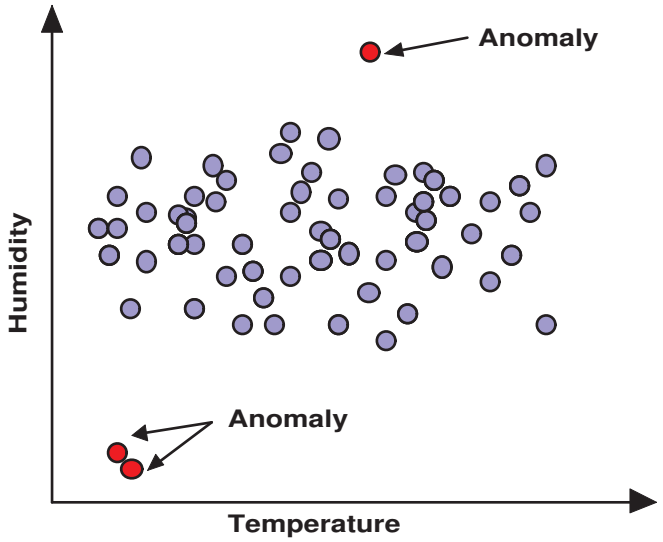

Figure 2.2: Anomalous instances determined based on unsupervised anomaly detection methods' assumptions

[3]. Therefore, the behavior of most instances in the data represents the normal behavior of the data.

2. Instances with behaviors which do not correspond to the normal behavior are anomalous instances as shown in Figure 2.2.

According to the assumptions, the unsupervised method creates the baseline based on instances in the main group containing the largest number of instances. The obtained baseline represents the normal behavior of the data. The method then classifies new traffic that deviates from the baseline as anomalous. The process for finding a baseline from an unlabeled dataset can be either offline or online. An offline unsupervised method typically learns an unlabeled dataset and creates a baseline in advance. While an online unsupervised method will not rely on an old-fashioned (unlabeled) dataset. The online unsupervised method uses traffic being monitored (or recently seen) to create a temporal baseline. They then use the temporal baseline to judge new traffic. A new temporal baseline will be created again at the next time interval based on the traffic being monitored or recently seen at that interval. Based on their assumptions, unsupervised anomaly detection methods tend to produce a number of false alarms. This is a well-

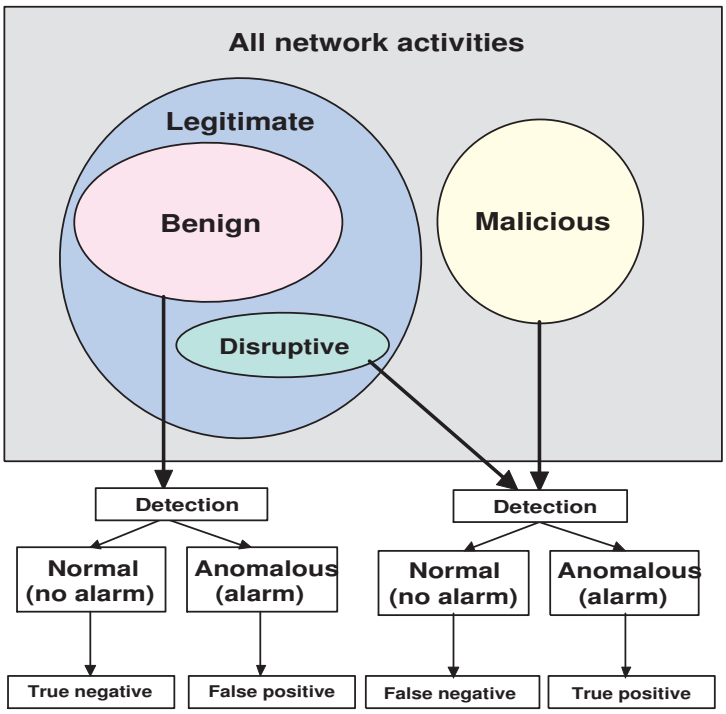

Figure 2.3: Types of network activities and decision process

known problem of unsupervised anomaly detection methods. The advantage of unsupervised methods is that they do not require pre-labeled training datasets. Moreover, they can efficiently detect new and unknown types of anomalies in every network.

2.2 Desirable Properties

Network anomaly detection methods must consider the following properties:

- *Detection rate* The most important property that anomaly detection methods must have is the ability to correctly detect all anomalies that are happening in networks. In other words, the anomaly detection methods should have high detection rates. The detection rate is also known as the true positive rate, hit rate, recall, and sensitivity. As shown in Figure 2.3, the true positive is an event when malicious and disruptive activities trigger an

anomaly-based NIDS to produce an alarm.

Mathematically, the detection rate can be expressed as

$$detection\ rate = \frac{number\ of\ anomalies\ correctly\ detected}{number\ of\ anomalies} \tag{2.1}$$

or

$$detection\ rate = \frac{number\ of\ true\ positives}{number\ of\ true\ positives + number\ of\ false\ negatives}, \tag{2.2}$$

where the false negative is an event when no alarm is raised when an anomaly (malicious and disruptive activities) has taken place.

- *False positive rate* The false positive or false alarm is an event that incorrectly trigger an anomaly-based NIDS to produce an alarm when no attack has taken place. In some domains, the false positive is called fall out. The false positives are not desired and may be considered annoying events. Therefore, an ideal anomaly detection method should produce a small number of false positives.

  The false positive rate is defined as

$$false\ positive\ rate = \frac{number\ of\ normal\ activities\ incorrectly\ detected}{number\ of\ normal\ activities} \tag{2.3}$$

  or

$$false\ positive\ rate = \frac{number\ of\ false\ positives}{number\ of\ false\ positives + number\ of\ true\ negatives}, \tag{2.4}$$

  where the true negative is an event when a normal activity is determined as a normal activity by a NIDS.

- *Speed* Speed (computational time) is a crucial property that NIDSs must have. This is a challenging task for developing a network anomaly detection method as mentioned in Section 1.2.2.

- *Flexibility and adaptation* Good anomaly detection methods are supposed to be able to be deployed in every network environment. In other words, the methods should be adaptive. For example, a method requires a few parameter reconfigurations and tunings when it is deployed in a new network. The method may automatically select suitable values for the method parameters. Furthermore, all types of inputs should be supported, e.g., raw traffic data and NetFlow data [41].

- *Usefulness of outputs* Anomaly detection methods should provide outputs that network administrators can utilize for countermeasures. Examples of countermeasures include blocking, slowing down anomalous traffic, and mitigation. These countermeasures always require important information about anomalies, such as IP, port, and protocol type. For example, to block traffic from an attacker by a Firewall, the network administrator requires at least the IP of the attacker. Therefore, researchers developing anomaly detection methods should consider the usefulness of the outputs of their methods. This is also a challenging task as described in Section 1.2.2.

2.3 Performance Comparison Among Anomaly Detection Methods

2.3.1 Evaluation Approaches, Challenges, and Problems

Performance comparison is a necessary task when a new anomaly detection method is introduced because the comparison result can tell how effective the new method is or how the new method can address old limitations or drawbacks of existing methods. To the best of our knowledge, there are three approaches used to compare the performance among anomaly detection methods.

- *Comparing with a well-known misuse-based NIDS* To confirm that a new method can detect anomalies, some anomaly detection methods are compared with a well-known and reliable misuse-based NIDS. The most popular misuse-based NIDS used in such comparisons is Snort [1]. In general, the

researcher of a method compares her method's result with Snort alerts obtained from testing the Snort with the same traffic dataset. The researcher may enable all Snort signatures or may enable a set of Snort signatures that catch the same types of anomalies.

- *Comparing with a similar anomaly detection method* Most anomaly detection methods usually are compared with one or two famous methods that are based on similar techniques. For example, a statistical anomaly detection method is compared with an existing statistical anomaly detection method. Some researchers may compare their new methods with their old methods that were previously proposed.

- *Comparing with a well-known anomaly detection method* New methods occasionally are compared with one or two well-known and successful methods that apply different techniques.

Each comparison approach has advantages and disadvantages. Comparing with a misuse-based NIDS like Snort may help to guarantee that a new method can actually detect known attacks. However, comparing with a misuse-based NIDS cannot prove that the new method can detect new and unknown anomalies. Comparing with a similar or well-known method can tell that the new method outperforms the competitive method or can address the problems that cannot be solved by the competitive method. The disadvantage of comparing with a similar or well-known method is that the researcher cannot confidently claim that her method is the best method. It is because the selected competitive method may show poor performance during the experiment due to improper parameter setting or the testing dataset.

Performance comparison among anomaly detection methods has several challenges and problems. Each anomaly detection method is specialized for detecting particular categories of anomalies. Some method focuses on detecting scanning activities while some method may focus on detecting flash crowds. This makes a comparison difficult. Apart from different targets, many researches do not provide all the details needed for correct reimplementation and parameter setting.

2.3.2 Evaluation Metrics

There are two metrics that are typically considered with regard to the performance of an anomaly detection method, namely detection rate (true positive rate) and false positive rate (false alarm rate). For the rest of this dissertation, the detection rate or true positive rate is called accuracy rate. The accuracy rate is the probability that anomalies are correctly detected. The accuracy rate can be calculated from Equation 2.1 or 2.2. Conversely, the false positive rate is the probability that non-anomalies are detected. Equations 2.3 and 2.4 present the formulas for calculating the false positive rate.

There are two additional metrics that are considered called true negative (called specificity in some domains) and false negative rate. The true negative rate is the probability that non-anomalies are not detected. The false negative rate is the probability that anomalies are not detected.

The true negative rate can calculated by

$$true\ negative\ rate = \frac{number\ of\ normal\ activities\ correctly\ rejected}{number\ of\ normal\ activities} \tag{2.5}$$

or

$$true\ negative\ rate = \frac{number\ of\ true\ negatives}{number\ of\ true\ negatives + number\ of\ false\ positives} \tag{2.6}$$

or

$$true\ negative\ rate = 100 - false\ positive\ rate. \tag{2.7}$$

For the false negative rate, it is calculated by

$$false\ negative\ rate = \frac{number\ of\ anomalous\ activities\ missed}{number\ of\ anomalies} \tag{2.8}$$

or

$$false\ negative\ rate = \frac{number\ of\ false\ negatives}{number\ of\ false\ negatives + number\ of\ true\ positives} \tag{2.9}$$

or

$$false\ negative\ rate = 100 - detection\ rate. \tag{2.10}$$

2.3.3 Traffic Datasets

One common approach to evaluate an anomaly detection method is testing the method with synthetic anomalies. The synthetic anomalies may be mixed with synthetic or real-world background traffic. The advantage of this approach is that the features of an anomaly (such as packet rate, duration, and the number of IPs) can be completely controlled. Clearly, the disadvantage is that these synthetic anomalies have not happened anywhere and are not new anomalies. Furthermore, this approach may be considered as experimenter bias. The second approach is testing an anomaly detection method with traffic collected at a real-world network, such as a university network, enterprise network, and backbone network. The advantage of using real-world traffic is that the researcher who develops the method can guarantee that the method is able to detect real-world anomalous events. However, there are a few labeled real-world datasets available as mentioned in Section 1.2.2.

From [42], most researchers test their methods with publicly available datasets. To the best of our knowledge, we list the datasets that are widely-used for evaluating anomaly detection methods.

DARPA 1998/1999 The DARPA 1998/1999 dataset [43] is a well-known traffic dataset for the evaluation of NIDSs. This dataset is provided by MIT Lincoln Laboratory. The DARPA 1998/1999 dataset provides five weeks of simulated network traffic collected between the US Air Force-based network and the Internet. The traffic was stored in pcap format. More specifically, the traffic data from weeks one and three are normal traffic data (without attacks). The traffic data from weeks two, four, and five contain both normal and simulated attack traffic. The simulated attacks consist of 177 instances of 59 different types of attacks, such as DoS, U2R, R2L, and probe attacks. The details of the simulated attacks are available in [43].

The DARPA 1998/1999 dataset has several drawbacks. The traffic was obtained from a simulated environment and the simulated attacks are not up-to-date [44]. The methodology used for generating the traffic is inappropriate for simulating actual networks [44]. Furthermore, the synthetic nature of the DARPA dataset is not without criticisms [45, 46].

MAWI The MAWI dataset is publicly-accessible backbone traffic dataset provided by the MAWI Working Group Traffic Archive [47]. This dataset is maintained by the WIDE Project [48]. In fact, the MAWI Working Group Traffic Archive provides real backbone traffic traces collected from several different links, such as samplepoint-A, samplepoint-C, samplepoint-D, and samplepoint-F. The traffic traces from each sample point have different characteristics and sizes. Specifically, the traffic traces from the samplepoint-A contain old traffic captured at a trans-Pacific T1 line in 1999 and 2001. The traffic traces from the samplepoint-D contain IPv6 traffic captured at an IPv6 line to 6Bone. The traffic traces from the samplepoint-F contain backbone traffic collected everyday from a trans-Pacific backbone link for 15 minutes (from 14:00 to 14:15). The capacity of the trans-Pacific backbone link is 150 Mbps and the link connects to an upstream Internet Service Provider (ISP). To date, the traffic traces from the samplepoint-F is the most widely-used traffic traces for researchers who want to test their anomaly detection methods. The reason is that the traffic traces from the samplepoint-F can represent fundamental data for real traffic analysis.

At the beginning, the MAWI dataset were unlabeled traffic dataset. Fortunately, there is a project named MAWILab project [49] that especially provides anomaly labels for the samplepoint-F traces. To label anomalous instances in the samplepoint-F traces, MAWILab combined the outputs of four different anomaly detection methods [50], namely Hough transform, Gamma distribution, KL divergence, and PCA-based methods. There are three types of anomaly labels: anomalous, suspicious, and notice. The traffic labeled as anomalous is real anomalous traffic. The traffic labeled as suspicious is suspicious traffic that is not clearly were identified by MAWILab classification method. The notice label indicates non-anomalous traffic. Since the labels for the samplepoint-F traces are publicly-available and trustworthy, they are becoming increasingly well-known and widely-used.

The advantage of the MAWI dataset, especially regarding samplepoint-F, is that the traffic is very up-to-date. Therefore, the anomalies in the traces tend to be up-to-date or new anomalies. Furthermore, the traffic in the traces is heterogeneous because the traffic is coming from many different institutes.

Abilene The Abilene dataset [51] is a public datasets that is also widely-used in intrusion and anomaly detection domains. More specifically, Abilene is the Internet2 backbone network connecting over 200 US universities and peering with research networks in Europe and Asia. Abilene has 11 Points of Presences (PoP). The Abilene dataset provides flow statistical data describing the byte count of each Original-Destination (OD) flows that are measured at five-minute intervals. The total number of OD flows is 121. The flow statistical data was stored in a matlab file.

CAIDA The Center for Applied Internet Data or CAIDA is a collaborative undertaking among commercial, academic, government, and research organizations that aims to promote cooperation in the engineering and maintenance of a robust and scalable Internet infrastructure. CAIDA also publicly provides several types of traffic datasets [52] for researchers, namely anonymized high-speed backbone datasets (collected during 2002 to 2014), Witty Worm dataset (collected in 2004), DDoS attack dataset (collected in 2007), and telescope datasets. The traffic in the datasets is raw traffic and was stored in pcap format. Therefore, researchers can analyze the traffic without any limitation. Unfortunately, there are no labels available for validation.

KDD Cup 1999 The KDD Cup 1999 dataset [53] contains a wide variety of intrusions simulated in a military network. Specifically, the data in this dataset are not raw traffic data, but are processed data consisting of approximately 4,900,000 records. Each record is a vector of 41 feature values extracted from a connection record. Each connection record was labeled as either normal or exactly one specific kind of attack (namely DoS, R2L, URL, or probe attack). The details of the features are available in [53]. The KDD Cup dataset provides both training data and testing data for researchers. Unfortunately, this dataset is relatively old. Therefore, this dataset may not be a good choice for the evaluation of anomaly detection methods that aim to detect new anomalies. To date, this dataset is popular for researchers who want to evaluate their data mining-based methods.

ISOT The ISOT dataset [54] is a new public traffic dataset provided by the ISOT Research Laboratory at University of Victoria. This dataset is the combi-

nation of several existing publicly available malicious and non-malicious datasets. The malicious traffic in the ISOT dataset consists of traffic from the French chapter of the Honeynet project [55] and Waledac botnets. The non-malicious traffic in the ISOT dataset consists of traffic from the Traffic Laboratory at Ericsson Research in Hungary and from the Lawrence Berkeley National Lab (LBNL) [56]. Fortunately, the ISOT Research Laboratory provides the ISOT Dataset Overview report [57] describing information about malicious and non-malicious IPs. The type of each malicious activity is also described in the report.

The advantage of this dataset is that it comes with labels. However, most malicious traffic in the dataset is botnet traffic. Other types of anomalies are not present in this dataset.

2.4 Statistical Methods

In general, a statistical anomaly detection method consists of two steps: (1) it observes the statistical features of traffic and (2) it detects feature changes. Table 2.2 presents some statistical features and change detection techniques used in statistical anomaly detection methods. Regarding the first step, at the early stage of the development of statistical anomaly detection methods, researchers were interested in simple statistical features, such as packet rate, destination IP rate, the ratio of the number of incoming traffic and outgoing traffic, the mean of flow sizes, and the mean of inter-arrival times. In recent years, researchers started to consider the distributions of fields in a packet header as features, such as the distributions of IPs and ports. This is because these features provide more fine-grained insights than simple statistical features [24]. Regarding the second step, several techniques are applied for detecting changes in the observed statistical features. Examples of applied techniques are forecasting techniques, thresholding techniques, PCA, and KL divergence. Below we describe the details of each existing statistical anomaly detection method. We also discuss the advantages and disadvantages of each method.

Brutlab [58] proposed an automated aberrant behavior detection method based on Holt Winters. This method predicts the future byte rate based on traffic that have been seen previously. More specifically, the future byte rate is

Authors	Aim	Considered statistical features	Applied change detection technique	Type
Brutlab [58]	Detect aberrant changes in a traffic time series	Byte rate	Holt Winters forecasting algorithm	Unsupervised
Kim et al. [59]	Detect anomalous flows	Packet/byte rates of each flow	Thresholding technique	Unsupervised
Soule et al. [60]	Detect anomalous OD flows	Byte rate of each OD flow	Kalman filter	Unsupervised
Wagner et al. [61]	Detect anomalous IPs/ports associated with fast scanning worms	Complexities of source/ destination IPs/ports	Thresholding technique	Unsupervised
Nychis et al. [62]	Detect times containing anomalies	flow size, degree distributions, entropies of source/ destination IP/port	Thresholding technique	Unsupervised
Lakhina et al. [22]	Detect anomalous OD flows containing volume-based anomalies	Packet/byte rates	PCA and heuristics	Unsupervised
Lakhina et al. [24]	Detect a wide set of anomalies	Entropies of source/ destination IP/port distributions	PCA and heuristics	Unsupervised
Li et al. [25]	Detect a wide set of anomalies	Entropies of the source/ destination IP/port distributions of aggregated flows	PCA and heuristics in [24]	Unsupervised
Gu et al. [16]	Detect times containing anomalies	Combined destination port and protocol type distribution	KL divergence	Supervised

Table 2.2: Considered statistical features and applied change detection techniques of existing statistical anomaly detection methods

calculated using the Holt Winters forecasting algorithm, which can predict the next value in a time series. An alarm is raised if the current byte rate is deviant too far from the predicted value. This method can be considered an unsupervised method because the method does not use a labeled dataset for the prediction. Unfortunately, a performance evaluation was not performed in this work. The advantage of this method is that the method is able to detect aberrant behaviors in a real-time context. However, the method cannot identity aberrant traffic.

Kim et al. [59] proposed an unsupervised flow-based abnormal traffic detection method. The method consists of two steps: (1) flow header detection and (2) traffic pattern detection. In the first step, the method calculates the number of packets and the number of bytes of each flow. The method then finds flows

that generate a lot of packets and bytes. Flows that are associated with a lot of packets and bytes are determined as abnormal flows. In the second step, the method identifies the type of each abnormal flow based on pre-defined attack patterns. For example, if a flow consists of small packets, this flow is a scanning flow. They tested their method with two synthetic anomalies such as scanning and TCP SYN flood. The result indicated that the method could detect all synthetic anomalies.

Soule et al. [60] proposed an unsupervised Kalman filter-based method for detecting anomalies in OD flows. The method creates a matrix called a traffic matrix of size $m \times n$, where m represents original nodes and n represents destination nodes. Each matrix element contains a counter (counting the number of bytes) that corresponds to the OD pair. The method then learns a chuck of OD flows and estimates a predicted traffic matrix based on those OD flows using the Kalman filter. At every a certain time interval, a predicted traffic matrix is estimated. The matrix is compared with the actual traffic matrix calculated based on monitoring OD flows. The discrepancy between the predicted traffic matrix and the actual traffic matrix is called innovation. The innovation can be considered as an anomaly score. If the anomaly score is high, the method raises an alert. In this work, they evaluated the method with two kinds of traffic data. The first dataset contains real-world traffic collected from the Abilene Internet2 backbone network. The second dataset contains traffic collected from the Abilene Internet2 backbone network and 500 synthetic anomalies. The results indicated that the method produced a 100% accuracy rate and 7% false positive rate. The advantage of this method is its lightweightness. Furthermore, the method can efficiently detect volume-based anomalies (e.g., floodings) in real-time. This method, however, can only inform network administrators of OD pairs associated with anomalies. The method cannot provide the details of anomalous flows.

Wagner et al. [61] proposed an unsupervised worm detection method. The method detects worms based on the assumption that the complexity of traffic will change when a scanning worm occurs. To detect worms, the method separately observes the complexities of source IPs, destination IPs, source ports, and destination ports of traffic. The complexity of each feature is estimated using a data compression algorithm, such as gzip, bzip2, and lzo. The method then raises

an anomaly alarm when the complexity of the monitoring feature changes. In this work, they tested their assumption with real-world traffic containing Blaster and Witty worms. The results indicated that the complexities of the four features changed (either increased or decreased) when the worms began scanning. The strong advantage of this method is that it can efficiently detect fast random scanning worms.

Nychis et al. [62] apply Shannon entropy [63], which is a powerful information theory technique to measure the uncertainty of data, for detecting traffic anomalies. Firstly, the method observes the entropies of source IP, destination IP, source port, destination port, flow size, and host degree distributions. Secondly, the method detects abnormal changes in those entropies using a heuristic approach. After testing the method with synthetic and real-world traffic captured for a month at Carnegie Mellon University, the results indicated that the entropies of the port and IP distributions could detect the same real-world anomalous activities, such as alpha flows (botnet activities) [24] . By contrast, the entropies of the host degree and flow size distributions could efficiently identify anomalous scanning, DoS attack, and peer-to-peer (p2p) activities that were not detected by the entropies of the port and address distributions. The correlation among those entropies was also studied. The study results indicated that the port and IP distributions were highly correlated. The host degree and flow size distributions were weakly correlated with each other. The host degree and flow size distributions were correlated with the port and IP distributions. The experimental and measurement results in this work confirm that the entropy of a feature distribution can provide more fine-grained insights than traditional statistical traffic features (e.g., byte rate).

Lakhina et al. [22] apply PCA, which is a well-known data dimension reduction technique, to detect anomalous OD flows. The method transforms traffic statistical data (e.g., byte count from Simple Network Management Protocol or SNMP) of all OD flows into a subspace. The subspace describes the principal components of the data and the residual components. The principal components represent the normal components of the data. To detect anomalies, the method uses a threshold-based separation method to measure the distance between the residual component and the principal component. The residual component that

is far from the principal component is defined as an anomalous component. The OD flow associated with the anomalous component is also defined as an anomalous OD flow. They validated the method with two real-world backbone traffic datasets such as Sprint-Europe and Abilene [51] datasets. The results showed that the method could successfully reveal OD flows that volume-based anomalies took place with high accuracy and small false positive rates. However, this method is designed for detecting only abnormal OD flows containing volume-based anomalies.

In the following year, Lakhina et al. proposed a new PCA-based method [24] that can detect a larger set of anomalies (not only volume-based anomalies). The difference between Lakhina et al.'s old method [22] and this new method is that the new method uses the entropies of feature distributions (the entropies of source IP, destination IP, source port, and destination port distributions) as inputs. While the old method uses SNMP data as inputs, this time Lakhina et al. validated the new method with Abilene [51] and Géant datasets. The results showed that the analysis of the entropies of feature distributions could better detect network anomalies than the analysis of volume-based metrics (e.g., packet rate). Examples of anomalies that can be seen in an entropy space are alpha flows, port scans, network scans, network outages, and flash crowds. Even though PCA-based anomaly detection methods [22–24] offer good detection performance, these methods can only identify anomalous OD flows, not the IP flows associated with anomalies.

Many researchers [25,26] tried to improve their anomaly detection methods by applying sketch technique in order to randomly aggregate traffic before detecting anomalies. Li et al. [25] proposed a sketch and PCA-based anomaly detection method that can identify anomalous flows. The Li et al.'s method can address the limitation of the previous PCA-based methods [22–24]. Li et al.'s method randomly aggregates traffic by mapping flows into a fixed-size matrix. The matrix is called a sketch. The mapping process is operated by several different 4-universal family hash functions [64]. The matrix has the size of $m \times n$, in which m represents different hash functions and n represents hash buckets. The method then calculates the entropy of a feature distribution of flows that belong to each matrix element. The obtained entropy is stored in a corresponding element of another

matrix of size $m \times n$. The method analyzes the entropies and detects a set of suspicious flows based on the PCA-based algorithm in [24] To identify anomalous flows, the method finds the suspicious flows that are flagged by all hash functions. The flagged flows are defined as anomalous flows. The evaluation results with traffic data in Abilene and Géant/Dante backbone networks indicated that the method could detect anomalies better than the PCA-based method [24] in both datasets. The advantage of this method is that it can specify anomalous flows. However, the method still has a well-known drawback in that the method is sensitive to PCA parameters [65].

Gu et al. [16] proposed a supervised anomaly detection method based on KL divergence. In the training phase, the method empirically clusters packets into 587 packet classes according to the protocol type and destination port. The method then trains a pre-labeled training dataset (which was collected at the University of Massachusetts Amherst gateway router) in order to find the baseline distribution of each class using Maximum Entropy estimation. In the testing phase, the method compares the empirical distribution of new packets in each class with the corresponding baseline distribution. The KL divergence between the two distributions is calculated. If the KL divergence exceeds a certain threshold, the method raises an alarm. Gu et al. tested the method using six different datasets collected at the same gateway. The instances in the datasets were labeled using human inspection. The results indicated that the method could detect anomalies with a low number of false negatives and false positives. The advantages of this method are that the method requires only a constant amount of memory. The method also consists solely of counting the packets in the traffic, without requiring any flow information. However, the method can only provide clusters containing possible anomalous packets. Other existing divergence measure-based anomaly detection methods are described in Chapter 4.

2.5 Data Mining-based Methods

Data mining is the computational process of discovering patterns in a large amount of collected data. To rapidly and automatically discover patterns in large data, data mining generally uses machine learning in order to help searching and

Authors	Training phase	Testing phase	Type
Moussaid et al. [66]	Cluster instances using K-means	Use the clusters to determine anomalous instances	Supervised
Manuyandi et al. [67] Solanki et al. [68]	Create a classifier using C4.5	Use the classifier to determine anomalous instances	Supervised
Chaturvedi et al. [69]	Create clusters/a classifier using SVM/C4.5	Use the clusters/classifier to determine anomalous instances	Supervised
Mulay et al. [70]	Cluster instances using SVM then create a classifier based on the clusters using a decision tree algorithm	Use the classifier to determine anomalous instances	Supervised
Münz et al. [27]	Cluster instances using K-means	Use the clusters to determine anomalous instances	Unsupervised
Portnoy et al. [28, 71]	Cluster instances using single-linkage clustering algorithm	Use the clusters to determine anomalous instances	Unsupervised
Eskin et al. [29]	Cluster instances using single-linkage clustering algorithm/K-NN/SVM	Use the clusters to determine anomalous instances	Unsupervised

Table 2.3: Training and testing phases of existing data mining-based anomaly detection methods

looking for patterns in the data. Due to the ability of data mining techniques, several effective data mining techniques have been applied for network intrusion and anomaly detection. Well-known data mining techniques include clustering, decision trees, Support Vector Machines (SVM), Genetic Algorithms (GA), and Neural Networks (NN). To the best of our knowledge, most data mining-based anomaly detection methods consist of two phases such as training and testing (detecting) phases. In the training phase, a training feature dataset (either labeled or unlabeled) is learned in order to create clusters or a classifier. In the testing phase, the clusters or classifier will be used to detect anomalous instances in the feature space. Table 2.3 describes the training and testing phases of some data mining-based network anomaly detection methods. In general, data mining-based methods consider raw packet features or flow features, such as source port, destination port, source IP, destination port, protocol type, TCP flag, packet size, and flow size. Some data mining-based methods also consider statistical features, such as the numbers of packets and the number of connections. The major difference between data mining-based methods and statistical methods is that the data mining-based methods may consider both raw and statistical features, but the statistical methods usually consider only statistical features as presented in the previous section. Moreover, most data mining-based methods consider

the correlation among features, but most statistical methods do not. Below we describe the details of some existing anomaly detection methods based on data mining techniques.

Moussaid et al. [66] proposed a supervised anomaly detection method based on K-means clustering algorithm [72]. In the training phase, training instances are clustered into several clusters based on the K-means clustering algorithm. The method then labels each cluster as either normal or anomalous. In the testing phase, the method checks the distance between a new instance and the labeled clusters. The new instance is labeled similar to the label of the nearest cluster. In this work, they compared the performance of the method with two clustering algorithms, namely Fuzzy C-Means (FCM) and Hierarchical algorithms. Using the KDD Cup dataset [53], the results showed that the performance of the K-means-based method was better than the performance of FCM and Hierarchical-based methods.

Manuyandi et al. [67] proposed a supervised anomaly detection system by cascading K-means and C4.5 decision tree algorithm. During the training phase, the method partitions instances in a fully-labeled training dataset into k disjoint clusters. Then, C4.5 decision tree (classifier) is trained with the instances in each cluster. The C4.5 classifier for each cluster is obtained. In the testing phase, the method finds the cluster that is closet with a new instance based on Euclidean distances. The method finally passes the new instance through the corresponding C4.5 classifier in order to determine if the new instance is normal or anomaly. To evaluate the performance of the method, Manuyandi et al. tested the method with training and testing data from the KDD Cup dataset [53]. The method was also compared with several different data mining techniques, such as ID3, Näive Bayes, SVM, and KNearest Neighbor (K-NN). The results indicated that the method could detect anomalies with 99.6 accuracy and 0.1 false positive rate. Furthermore, the method mainly outperformed those different data mining techniques in terms of accuracy and false positive rates. A similar supervised method, which is based on the K-means clustering and C4.5 decision tree algorithms, was recently proposed in 2014 by Solanki et al. [68]

Chaturvedi et al. [69] developed two data mining-based anomaly detection methods for supervised anomaly detection. The first method is based on SVM.

The second method is based on C4.5 decision tree algorithm. During the training phase, the SVM-based method finds the optimal hyperplane that can separate two classes of given samples with a maximal margin. In this case, the classes are normal and anomaly classes. For the testing phase, if a new instance geometrically falls to the normal class, the new instance is a normal instance. On the other hand, if the new instance falls to the anomaly class, the new instance is an anomaly. For the C4.5-based method, the method creates a classifier based on C4.5 decision tree algorithm. In the testing phase, the C4.5-based method uses the classifier to determine new instances. They tested the methods with the KDD Cup dataset [53] and also compared the performance between both methods. The results showed that the accuracy rates of both methods were approximately 85%. The false positive rates of the SVM-based and C4.5-based methods were 1.6% and 0.8%, respectively.

Mulay et al. [70] combine SVM and a decision tree algorithm for supervised anomaly detection. Specifically, the method uses SVM to cluster training instances into several classes. The method then creates a decision tree based on those classes. The evaluation results with the KDD Cup dataset [53] showed that the integration of SVM and decision tree models gave better detection results than an individual model.

Münz et al. [27] present a novel flow-based unsupervised anomaly detection scheme based on K-means clustering algorithm. The training phase deals with preparing unlabeled raw data to be ready for clustering and finding centroids using K-means clustering algorithm. The testing phase deals with comparing new monitoring data with the obtained centroids and identifying anomalous time intervals. In the training phase, flow data (e.g., NetFlow data) are processed and converted to a feature space. Then, the method analyzes and groups the data in the feature space into two clusters based on K-means clustering algorithm, namely normal and anomaly clusters. In the testing phase, when a new flow record arrives, the new record is converted to a feature space. The method calculates the distances between the position of the new record in the feature space with the two centroids (of the normal and anomaly clusters). If the position is closer to the anomaly cluster centroid than to the normal one, or if the distance to the normal cluster centroid is larger than a predefined threshold, the current time

interval is marked as anomalous. The advantage of this method is that it does not require a labeled dataset. However, an essential problem of this method is defining an appropriate number of clusters [27]. This problem is not the problem of only this method, but other K-means-based methods also face this problem. Furthermore, the limitation of this K-means-based method is that it can only identify anomalous time intervals.

Portnoy et al. [28, 71] proposed an unsupervised anomaly detection method using single-linkage clustering algorithm. In the training phase, their method starts with an empty set of clusters and generates the clusters with a single pass through an unlabeled training dataset. For each instance in the training dataset, the method computes the distance between the instance and each of the centroids of the clusters. The instance is assigned to the closest cluster (with the shortest distance). If the instance is not close to any cluster, a new cluster is created for the instance. The clusters are labeled (normal or anomaly) based on the assumption that a normal cluster (cluster which contains normal data) will have a much larger number of instances than an anomalous cluster. Therefore, the cluster with a large number of instances is a normal cluster and vice versa. In the testing phase, a new instance is designated as anomalous or normal based on the distance between the new instance and the centroids of the clusters obtained from the training phase. In this work, the method was evaluated with the KDD Cup dataset [53]. The results showed that the best accuracy rate obtained was 65%. The false positive rate in the same experiment was 1.78%. The results also showed that the number of clusters actually affected the detection performance of the method in terms of both accuracy and false positive rates.

Eskin et al. [29] proposed three different clustering-based methods for unsupervised anomaly detection. The first method clusters data based on the idea of Portnoy [71]. Specifically, the first method creates single-linkage clusters and labels the clusters based on the density of each cluster. To classify a new instance, the first method determines the new instance based on the distances between the new instance and the cluster centroids. The second method clusters data based on K-NN algorithm. In the second method, the K-NN of each data point in the feature space is calculated. If the sum of the distances between the data point to the K-NN is greater than a threshold, the method will consider the data

point an anomaly. The third method clusters data based on SVM. Specifically, all data points in the feature space are mapped into another feature space using a Gaussian kernel. In the second feature space, a hyperplane is drawn to separate the majority of the data from the origin. The remaining points represent anomalies. Two datasets were used for the performance evaluation of the three methods, such as KDD Cup [53] and DARPA [43] datasets. The results indicated that the three methods perfectly detected anomalies without false positives when testing them with the DARPA dataset. For the KDD Cup dataset, the single-linkage clustering-based method could detect anomalies with 93% accuracy. The single-linkage clustering-based method's false positive rate was 10%. For the K-NN-based method, the accuracy rate was 91% and the false positive rate was 8% (the best performance from four experiments). The SVM-based method outperformed the single-linkage-based and K-NN-based methods in terms of accuracy rate. The SVM-based method's accuracy rate was 98%, while the false positive rate was 10%.

2.6 Time-Frequency Analysis-based Methods

Many anomaly detection methods take the advantage of TFR techniques that are powerful to represent a time domain data or time series as a time-frequency domain. Examples of TFR techniques are Short-Time Fourier Transform (STFT) [76] and Wavelet transform [77]. Generally, Time-Frequency Analysis (TFA)-based methods use a TFR technique to discover anomalies in network time series. The ideas behind TFA-based anomaly detection methods are described as follows:

- A time series data that represents network behavior (e.g., bandwidth usage) between two network devices on the Internet can be considered as a traffic signal.
- The traffic signal, which is a time domain data displaying data trend over time, can be represented in a different form called time-frequency domain perspective using a TFR technique.
- A TFA can provide new insights of the traffic signal that may not be available when using a time domain analysis.

Authors	Aim	Considered traffic signals	Used TFR technique
Barford et al. [30]	Detect anomalous times caused by short-lived/long-lived anomalies	SNMP/IP flow data	DWT
Magnaghi et al. [73]	Detect anomalous times caused by misconfigurations	Packet rate	DWT
Dainotti et al. [35]	Detect anomalous times caused by volume-based DoS attacks	Packet rate	CWT
Huang et al. [32]	Detect anomalous times caused by anomalies	Packet rate and byte rate	DWT
Carl et al. [74]	Detect anomalous times caused by flooding-based DoS attacks	CUSUM statistic of a packet rate	DWT
Lu and Ghorbani [75]	Detect anomalous times caused by anomalies	Flow/packets per flow/bytes per flow/ bytes per packet rates, ratio of flow rate to bytes per packet	DWT
Pukkawanna et al. [38]	Detect anomalous times caused by anomalies	Packet rate of randomly aggregated traffic	DWT
Callegari et al. [39]	Detect anomalous flows	Flow rate of randomly aggregated flows	DWT

Table 2.4: Considered traffic signals and used TFR techniques of existing TFA-based anomaly detection methods

- Abnormalities in the time-frequency domain are unusual time-frequency behaviors that can be frequently caused by network anomalies, such as DoS attacks, flash crowds, and scannings.

Below we describe the state-of-the-art TFA-based methods for network intrusion and anomaly detections. Table 2.4 presents a summary of the traffic signals and TFR technique used in each TFA-based method. We note that the TFA-based methods described below are unsupervised methods, with the exception of Lu and Ghorbani's work.

Cheng et al. [78] proposed a spectral-based method to discriminate normal TCP flows from DoS attack flows. Their idea is that TCP flows that do not show periodicity, are likely to be attack flows. Based on the fact that a TCP flow exhibits periodicity, they observe the Power Spectral Density (PSD) of a traffic signal, which is the number of packet arrivals. The PSD shows the periodicity of the signal. If a TCP flow exhibits strong periodicity around its round-trip time in both flow directions, that flow is a normal flow. On the other hand, an attack TCP flow usually does not exhibit periodicity. They validated the effectiveness of their method with real trace analysis. The trace was collected at a 100Mbps link

connected the Harvard Faculty of Arts and Sciences to the rest of their campus. The results indicated that the method correctly identified normal TCP flows with 82% accuracy.

Barford et al. [30] proposed a Discrete Wavelet Transform (DWT)-based methods for detecting four classes of anomalies: outages, flash crowds [11], attacks and measurement failures. They consider SNMP and IP flow data as traffic signals and use DWT to decompose each traffic signal to several frequency bands, namely High (H), Medium (M), and Low (L) frequency bands. To detect short-lived anomalies (e.g., network failures and DoS attacks), the H-part is associated with frequency levels one, two, and three. The M-part is associated with frequency levels four and five. The L-part is associated with the remainder. To detect long-lived anomalies (e.g., flash crowds), only the M-part and L-part are considered. The M-part is associated with frequency levels six, seven, and eight. The L-part is associated with the remainder. After the traffic signal are decomposed, they then use a deviation score to identify anomalies. They tested the method with the SNMP and IP flow data collected at the University of Wisconsin's border router. They found that their DWT-based method could isolate both short-lived and long-lived anomalies. Furthermore, the method additionally revealed some hidden anomalies that are not visually seen. They also compared the detection performance between their deviation score-based thresholding algorithm and the Holt Winters forecasting algorithm. The deviation score-based method could detect anomalies slightly better than the Holt Winters forecasting algorithm. The key advantage of the DWT-based anomaly detection method is that it can detect hidden anomalies in a chaotic traffic signal. Unfortunately, choosing which frequency level should be considered often requires an expert understanding of the performance of wavelet decompositions [30]. In addition, the performance depends on the mother wavelet.

Magnaghi et al. [73] proposed a detection algorithm capable of isolating new misconfigurations and other anomalies, such as IP address duplication, packet filtering misconfiguration, and DoS attacks. The algorithm has the following four steps. In the first step, Management Information Base (MIB) packet counter is periodically collected from a link. This time series is the input signal for the next step. In step two, the signal is decomposed in the wavelet domain using DWT.

In step three, the algorithm analyzes the local minima of the signal energy at resolution levels regulated by the underlying Retransmission Time-Out (RTO) mechanism. Finally, an alarm is raised to warn administrators if the energy coefficients of the signal energy produces a value in a specified range. Magnaghi et al. tested the algorithm with simulated traffic and real traffic from a network. Magnaghi et al. found that the algorithm could successfully detect the presence of the anomalies with 95% to 100% accuracy.

Dainotti et al. [35] combine statistical (time domain analysis) and TFR techniques to create an unsupervised and automated system for detecting volume-based DoS attacks. The system consists of two stages. The first stage is the time domain change detection that aims to detect anomalous time intervals that may contain DoS attack traffic. To detect changes in an input packet rate time series (signal), the system uses CUmulative SUM (CUSUM) and adaptive thresholding algorithms proposed in [79]. If the system finds an anomalous time, the system will trigger stage two. The second stage is associated with Continuous Wavelet Transform (CWT)-based DoS attack detection. The main aim of the second stage is to reduce false alarms caused by the first stage and to identify anomalous time. In the second stage, the system computes the CWT of the same packet rate time series and seeks the zero-crossings in the coefficients of each scale to determine anomalous intervals. The mother wavelet used is Morlet. An anomaly is found, if the maximum coefficient in the anomalous interval is greater than a threshold. They evaluated the system with a big set of real-world traffic from different sources, namely DARPA [43] and UCLA [80] datasets, which contains traffic collected in August 2001 at the border router of Computer Science Department, University of California Los Angeles. They also tested their method with traffic from UNINA dataset containing traffic captured at a router at the University of Napoli Federico II. The results showed that 70% of the start and end times of the anomalies were correctly identified. The advantage of this CWT-based DoS attack detection system is that the CWT coefficient redundancy allows us to identify anomaly points at every scale with the same time-resolution of the input signal.

Huang et al. [32] proposed a framework called Waveman that uses an open source tool called LastWave [81] to provide a real-time analysis of network traffic.

Traffic is captured and the packet and byte count signals are computed. LastWave [81] reads each signal and calculates the level 1, level 2, and level 3 coefficients using the DWT. The level 1 coefficients represent the noise of the signal. The level 2 coefficients represent the bulk of information of the signal. The level 3 coefficients represent the sparse information of the signal. They tested four different mother wavelets with the proposed framework in terms of the ability to reveal anomalies in a traffic signal. The four mother wavelets under consideration are Coiflet, Morlet, Daubechies, and Paul. All mother wavelets were tested with the DARPA dataset [43] consisting of several simulated attack traffic (described in Section 2.3.3). They were also tested with the real-world traffic collected at a domain name service company. This traffic contains real-world port scans. To evaluate the performance of each mother wavelet, they used a percentage deviation algorithm and the Rényi entropy-based method [82]. The evaluation results showed that Coiflet and Paul wavelets had better characteristics when faced the anomalies in the tested traffic.

Carl et al. [74] proposed a flooding-based DoS attack detection method, which combines the CUSUM algorithm with DWT. The way that this method uses the DWT is similar to the Barford et al.'s work [30]. However, Carl et al. use the DWT to detect changes in the CUSUM statistic of a packet rate, not in wavelet coefficients. The method was tested with many kinds of traffic data, such as simulated traffic data from the DARPA dataset [43] and traffic data captured from operational networks. The results indicated that the method had lower false positive rates and shorter reaction times than competing methods.

In [33], Lu and Ghorbani combine DWT and AutoRegressive eXogenous (ARX) model to develop a supervised anomaly detection method. They use ARX model to create a prediction model in the training phase. The training data are the approximations of DWT of a traffic signal (e.g., TCP flow rate). The mother wavelet is Haar. In the testing phase, new monitoring traffic is compared with the prediction model. The unmatched traffic is used to detect outliers using Gaussian Mixture Model (GMM). Lu and Ghorbani also presented fifteen types of traffic signals and evaluated the signals and method with the DARPA dataset [43]. Data from weeks one and three were used to create a prediction model. Data from weeks four and five were used for testing the performance of the signals and method.

The best accuracy rates in terms of attack types and attack instances were 100% and 94.67%, respectively. They also tested the method in a real large-scale ISP network [83]. The results showed that the method successfully detected five our of six attacks, total. Furthermore, they compared the performance with an unsupervised clustering-based anomaly detection method [75] and a nonparametric CUSUM-based SYN flooding attack detection method [84]. The results showed that their DWT and ARX model-based method provided the best performance.

The sketch technique is also employed to improve the performance of TFA-based anomaly detection methods [38,85]. Pukkawanna et al. [38] used the sketch technique to randomly aggregate and separate traffic data into several smaller groups of traffic. Specifically, five different hash functions from [86] were used and the key is the source IP string of each packet. The sketch size is eight. After the traffic is aggregated, they then use the DWT with Daubechies-4 mother wavelet to detect anomalies in the packet rate signal of each traffic group based on the deviation score-based thresholding algorithm in [30]. Finally, to avoid high false positives, they will raise an anomaly alert along with the information (e.g., IP and port) of an anomaly if the anomaly is found in many signals. They evaluated the method with two unlabeled traffic traces from the samplepoint-F MAWI dataset [47]. One of the tested traces was collected in the same day that the Internet Sasser worm was spreading (August 1st, 2004). They compared the method with a port-based method. The best result showed that 71% of the anomalies in the August 1st trace were correctly detected. They also compared the performance with the Barford et al.'s work [30] and an image processing-based method [37]. The results showed that the combined sketch and DWT-based method could detect better the Sasser worm attack traffic than Barford et al.'s work [30]. Furthermore, the combined sketch and DWT-based method was also capable of detecting low-intensity anomalous traffic as well as some types of malicious traffic that could not be identified by [30]. Regarding the comparison with [37], the results showed that the combined sketch and DWT-based method generated a higher number of anomaly alerts. There were some anomalies that were reported by their method and [37].

Callegari et al. [85] proposed an unsupervised anomaly detection method similar to [38], but focus to improve detection time. To detect anomalies, they aggre-

gate NetFlow data and store the results in sketch tables representing time series (traffic signals) of aggregated data. The DTW with Daubechies mother wavelet is applied to each time series to find anomalous time intervals in the same manner with [30, 38]. IPs that are in detected anomalous time intervals of all hash functions are real anomalous IPs. They evaluated their method with the Abilene dataset [51]. The accuracy rates of the method were up to 100%. While, the false positive rates were affected by the threshold. The difference between Callegari et al.'s method [85] and Pukkawanna et al.'s method [38] is that Callegari et al. use a sliding window mechanism to cut the time series and use an adaptive threshold to detect anomalous time intervals. This may help the Callegari et al.'s method detect anomalies better than using a fixed threshold [30, 38].

Chapter 3

S-Transform-based Traffic Anomaly Detection (STAD)

3.1 Introduction and Problem Statement

Unsupervised anomaly detection methods have been proposed to address the following limitations: The limitation of signature-based methods that cannot detect unseen anomalies; The limitation of supervised and semi-supervised anomaly detection methods that rely on labeled training datasets. Time-Frequency Analysis (TFA) is the study of a signal in the time and frequency domains simultaneously. The concept of the TFA motivates research in intrusion and anomaly detection domains to apply Time-Frequency Representation (TFR) techniques for network anomaly detection [30–36, 38, 73, 74, 85]. From Table 2.4, most TFA-based methods employ either DWT or CWT to reveal anomalies in the time-scale domain of traffic signals. The DWT offers lower redundancy in transformations and is easier to implement than the CWT. On the other hand, the CWT offers deep TFA so that the method can see as many as anomalies in the signal. However, the use of both DWT and CWT have the following issues. The justification of mother wavelets, which is an important parameter that impacts detection performance, for Wavelet transform-based anomaly detection methods is still left open for discussion [32]. For example, the authors in [32] claim that Coiflet and Paul wavelets have better characteristics than Daubechies and Morlet wavelets. Meanwhile, many works [30, 38, 85] use Daubechies and those authors archived

satisfactory performance. Furthermore, for the works that employ the DWT, determining the number of decomposition levels still remains a point of contention.

In this chapter, we firstly introduce S-transform [87] in order to address the issues of the use of Wavelet transform for network anomaly detection. In short, S-transform is a TFR technique that also can perform Multi-Resolution Analysis (MRA) similar to the DWT and CWT. To discover time-frequency information of a signal, S-transform simply segments a signal using an adaptive Gaussian window and discovers the time-frequency information of the segmented signal using Fourier transform. This simplicity of S-transform addresses the issues of selecting suitable mother wavelets and decomposition levels of Wavelet transform-based anomaly detection methods.

In this chapter, we also propose an unsupervised anomaly detection method based on S-transform. We name our method STAD. The ideas behind STAD are similar to that of TFA-based anomaly detection methods as described in Section 2.6. In short, to detect anomalies, STAD converts a packet stream to several types of traffic signals. Next, STAD extracts time-frequency information of each traffic signal using S-transform. Finally, STAD detects anomalies in the output of S-transform using heuristics. We evaluated STAD with the MAWI [47] and DARPA [43] datasets described in Section 2.3.3. Furthermore, we compared STAD with Barford et al.'s DWT-based anomaly detection method [30] described in Section 2.6. The results showed that STAD could detect more anomalies than (especially hidden anomalies) the DWT-based method in both datasets.

Some network administrators still hesitate to deploy anomaly-based NIDSs to their networks due to the high false alarm problem. In this chapter, we propose an idea to address the false alarm problem by employing sketch technique. STAD enhanced by the sketch technique provides lower false alarms and higher accuracy.

This chapter is organized as follows. Section 3.2 describes S-transform in detail and demonstrates the ability of S-transform to reveal malicious activities. Section 3.3 explains the details of STAD. The performance evaluation is presented in Section 3.4. Section 3.5 presents how to use the sketch technique to improve the performance of STAD. Performance evaluation of the improved STAD is also presented in Section 3.5.

3.2 S-transform

3.2.1 Definition

The TFR is a view of a signal (taken to be a function of time) describing frequency components of the signal and time in which the frequencies exist. The first TFT is Short-Time Fourier Transform (STFT), which was proposed in 1946 to tackle a problem of Fourier transform that cannot provide time information of frequencies that exist in a signal. In order to obtain time information, the STFT uses a fixed-size sliding window to segment the signal and then conducts a Fourier analysis on the segmented signals. The major drawback of the STFT is that it can either give a good frequency resolution or a good time resolution. In other words, its performance depends upon the window width. In order to address the drawback of the STFT, Wavelet transform was proposed. In Wavelet transform, a signal is broken up into shifted and scaled versions of a mother wavelet which is a continuous function in both time and frequency domains.

Continuous Wavelet Transform (CWT) of a signal $x(t)$ is defined as

$$CWT(\tau, s) = \int_{-\infty}^{+\infty} x(t) \frac{1}{\sqrt{|s|}} w(\frac{\tau - t}{s}) dt, \tag{3.1}$$

where t denotes time. s and τ denote scale and translation of the mother wavelet w, respectively. Equation 3.1 indicates that the CWT analyzes the signal at different frequencies with many different scales. The CWT gives good time resolution and poor frequency resolution at high frequencies (small scale) and good frequency resolution and poor time resolution at low frequencies (large scale). This approach is called Multi-Resolution Analysis (MRA).

For the Discrete Wavelet Transform (DWT), the DWT of a signal $x(t)$ is calculated by passing the $x(t)$ through a series of low-pass and high-pass filters. When the $x(t)$ is initially passed the two filters, the outputs are the detail coefficients (from the high-pass filter $g()$) and approximation coefficients (from the low-pass $h()$). The outputs subsampled by two are

$$y_{low}(t) = \sum_{-\infty}^{+\infty} x(k) h(2t - k), \tag{3.2}$$

$$y_{high}(t) = \sum_{-\infty}^{+\infty} x(k)g(2t-k). \tag{3.3}$$

The decomposition can be performed several times. The outputs obtained from the first decomposition are level 1 coefficients. For the second decomposition, the level 1 $y_{low}(t)$ (approximation coefficients) is decomposed resulting in level 2 detail and approximation coefficients.

Even though the CWT is extremely useful, but the real frequency components of a signal are still important. The CWT considers time-scale region instead of time-frequency region. In order to carry out real TFA as well as address the STFT's drawback, Stockwell et al. [87] proposed S-transform in 1996. S-transform is a special case of the STFT that replaces the fixed-size window with a Gaussian window.

The S-transform of a signal $x(t)$ is defined as

$$ST(\tau, f) = \int_{-\infty}^{+\infty} x(t)g(\tau - t, f) \tag{3.4}$$

where the Gaussian window function $g(\tau - t, f)$ is given by

$$g(\tau, f) = \frac{|f|}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{(\tau-t)^2 f^2}{\sigma^2}} \sigma > 0 \tag{3.5}$$

where t and τ are both time, but τ is used to control the time resolution, f is the Fourier frequency, and σ is the scale parameter that controls the frequency resolution. By default, σ is one. The vector of Fourier frequencies f is: $f = \{0, 1, 2, ... \lfloor L/2 \rfloor\}$, where L is the length of the signal $x(t)$.

Typically, an S-transform's output is stored in a two-dimensional matrix where the rows are frequencies, the columns are the times, and each element is an amplitude of each frequency. In other words, the row vector represents the changes of a certain frequency with time. The column vector represents the frequency distribution of a certain time. To understand S-transform's output more clearly, let us illustrate an output obtained by the S-transform of a multi-frequency component non-stationary signal recorded on 400 points. Figure 3.1 shows the analyzed signal and S-transform's output describing frequency behavior of the signal. From the figure, the S-transform's output (viewed as an image) reveals that the beginning and the end of the signal contains low frequencies around 45Hz with low

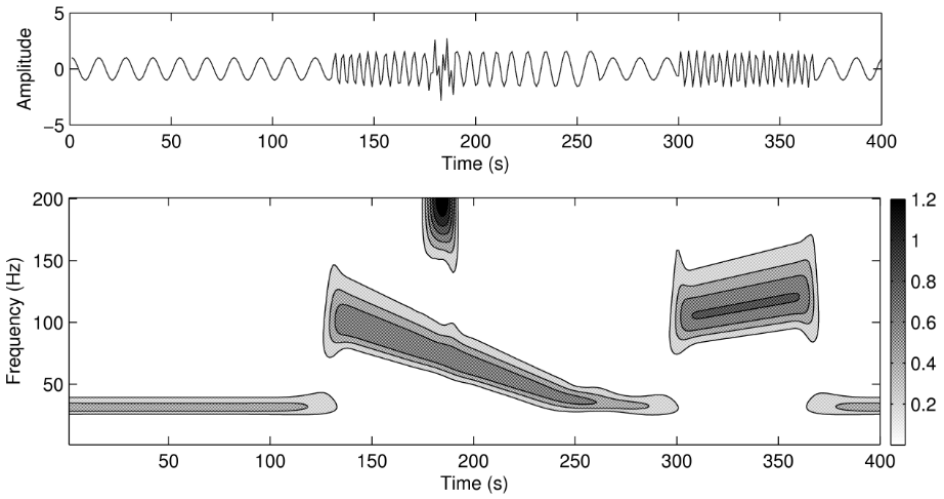

Figure 3.1: Multi-component non-stationary signal and S-transform's output describing frequency behavior of the signal

amplitudes. The signal from 130 seconds to 260 seconds contains frequencies decreasing from 100Hz to 50Hz. The highest frequency exists at the center of the signal in a short time. A middle-frequency ranging from 100Hz to 150Hz is located between 300 seconds to 360 seconds. From Figure 3.1, we can see that S-transform can correctly and precisely express the time-frequency behavior of the signal. Because of this performance, S-transform has been used in various fields of research [88–91].

S-transform has the following advantages.

- It can observe how frequency components of the signal change over time.
- It provides Multi-Resolution Analysis (MRA).
- It produces a time-frequency plot that is easier to visually analyze time-frequency behavior than a CWT's output. The reason is that the CWT produces a time-scale plot.
- It uses the Fourier kernel to provide the absolute phase information of each frequency component. This phase information is referenced to the time origin. As a result, S-transform provides supplementary information about

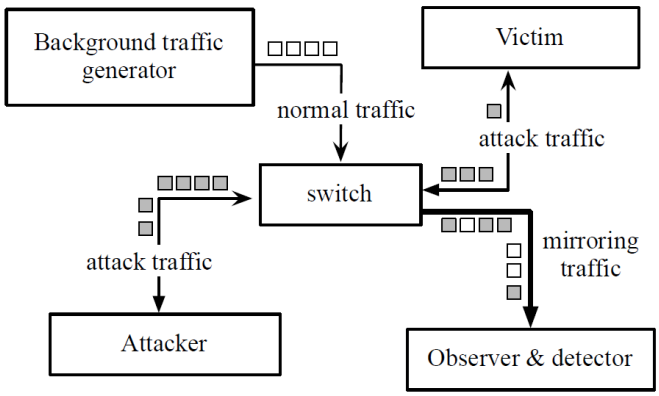

Figure 3.2: Experimental test-bed consisting of four actors: background traffic generator, attacker, victim, and detector

spectra which is not available from locally referenced phase information obtained by the CWT [87].

3.2.2 Capability of Revealing Attacks

We would like to investigate if S-transform is capable of revealing unusual frequency behavior in a traffic signal caused by general malicious activities, such as scannings and floodings. We thus performed experiments in a test-bed. The test-bed consists of four interconnected computers with the following setup: one computer acts as attacker, one computer acts as victim, one computer acts as background traffic generator, and one computer acts as observer. The test-bed architecture is shown in Figure 3.2. All computers in the test-bed network were installed with Ubuntu version 10.04. The background traffic generator was responsible to generate clean background traffic by using Tcpreplay [92]. Specifically, the traffic trace used as the clean background traffic is a trace from the MAWI dataset [47]. Specifically, the trace was captured on January 2nd, 2011 at samplepoint-F. More than 50% of the traffic in the trace was web traffic. About 15% of the traffic was associated with other applications. About 20% of the traffic was UDP traffic. About 2% of the traffic was ICMP traffic. The background traffic generator replayed the traffic for three minutes without modifying packets. The background traffic generator also preserved original inter-packet time. The

attacker was responsible for generating malicious traffic to the victim, namely SYN flood, ICMP flood and port scan traffic. The detector collected raw traffic and processed the traffic with the following two steps:

1. The detector counted the number of packets every second. It then stored the values in an array. The first and last elements of the array are the total numbers of all packets found within the first and last minutes, respectively. For example, if the traffic is collected for one minute, the first element's value is the number of packets seen within the first second and the last elements value is the number of packets seen within the 60th second. The array represents a traffic signal. The detector also removed the DC component of the traffic signal by subtracting its mean value.

2. The detector converted the traffic signal to a time-frequency domain using S-transform with default parameter values (see Section 3.2). The output from the detector was a two-dimensional matrix or an image with a width of 60 and a height of 30. The image illustrates time-frequency behavior of the traffic signal created from the mixed malicious and background traffic.

In these experiments, five malicious activities were generated by the attacker, namely a TCP SYN flood, three types of port scans, and an ICMP flood. Below we describe how the attacker generated each malicious traffic and show the S-transform's outputs obtained from the detector.

A SYN Flood by Neptune The attacker used the Neptune tool [93] to continuously send 10,000 TCP SYN packets at the second minute (14:01) to the victim. Figure 3.3 shows the S-transform of the traffic signal containing the SYN flood. The figure shows that the SYN flood traffic caused explicit changes in medium frequencies of the traffic signal.

A Port Scan by portscan.c After the traffic generator initially added the clean traffic to the test-bed network for one minute, the attacker launched a port scan by portscan.c [94] to probe vulnerability ports of the victim host. All packets that passed the detector host's interface were collected and converted into a traffic signal. Figure 3.4 shows the output derived from transforming the

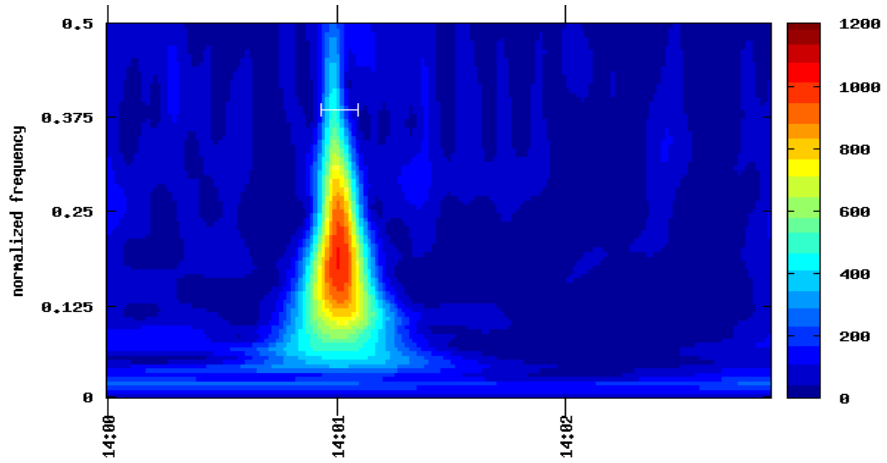

Figure 3.3: Time-frequency domain (by S-transform) of a traffic signal containing a SYN flood

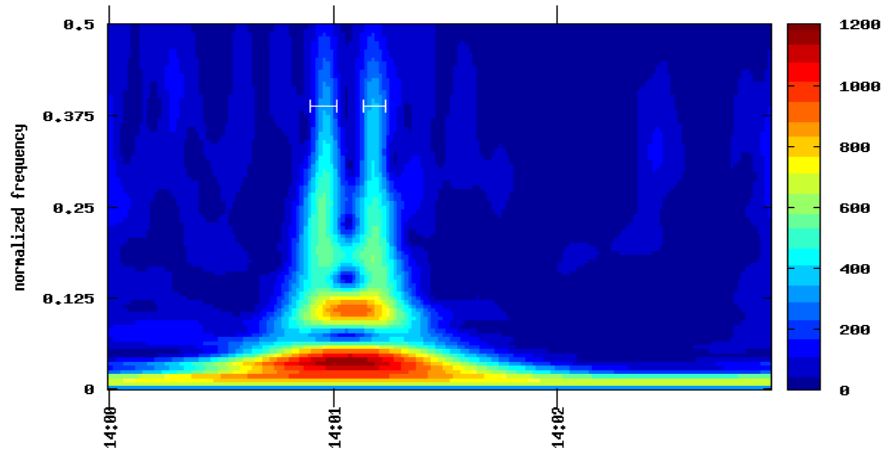

Figure 3.4: Time-frequency domain (by S-transform) of a traffic signal containing a port scan

traffic signal by S-transform. The figure shows that this kind of port scan caused high amplitude (high energy with red coloration) at around low-frequency area.

A Default Nmap Scan Nmap [95] is a tool used for creating a port scan. It offer several types of scans such as fast scan, default scan, and full scan. The numbers of target ports among them are different. In this experiment, the attacker generated a default Nmap scan, which scanned ports of the victim between

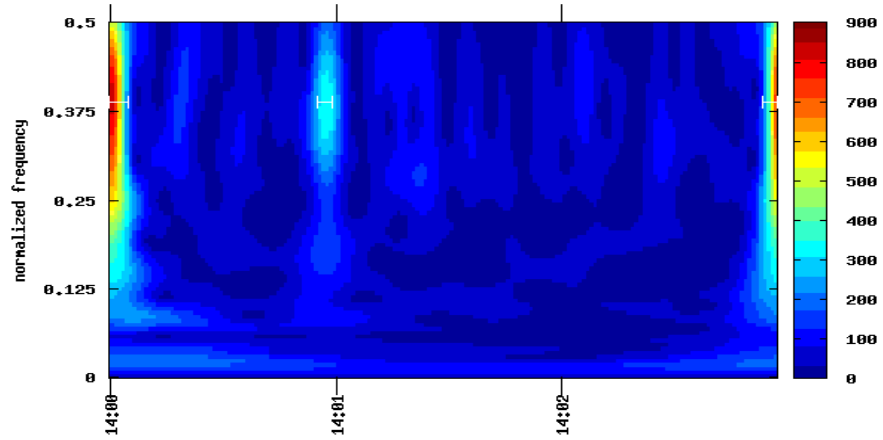

Figure 3.5: Time-frequency domain (by S-transform) of a traffic signal containing a default Nmap scan

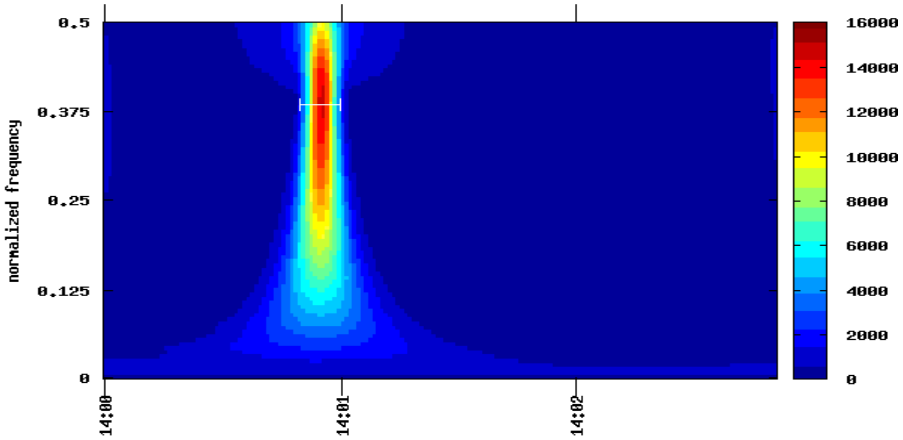

Figure 3.6: Time-frequency domain (by S-transform) of a traffic signal containing a full Nmap scan

1 and 1,024 as well as any ports listed in the services file which came with Nmap. Figure 3.5 shows the S-transform of the traffic signal containing the default Nmap scan. The figure shows that the default scan caused high amplitude at high frequencies in time-frequency domain. The figure also shows that S-transform could precisely reveal the scanning at the correct time.

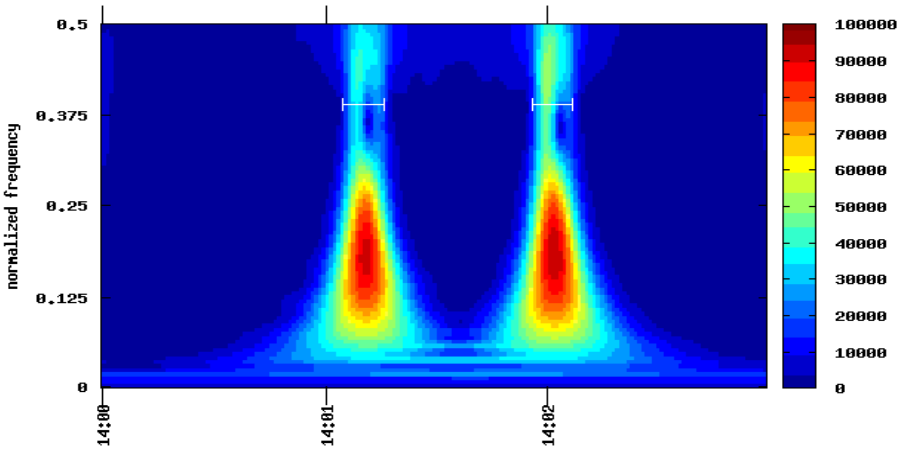

Figure 3.7: Time-frequency domain (by S-transform) of a traffic signal containing two instances of Jping

A Full Nmap Scan The full Nmap scan completely scanned ports 0 to 65,535 of the victim and started from the second minute (14:01) of the experiment. Figure 3.6 shows the S-transform of the traffic signal containing the full Nmap scan. The full Nmap scan caused high amplitude at high frequencies in time-frequency domain similar to the default Nmap scan. The full Nmap scan caused a higher amplitude than the default Nmap scan because the number of scanned ports of the full Nmap scan was larger than the number of scanned ports of the default Nmap scan.

Multiple Ping-of-Deaths by Jping Jping [96] is an ICMP flood attack that attempts to crash the operating system of a target host by sending many ICMP echo request packets to the target. In this experiment, Jping sent 1,000,000 ping packets to the victim at the second and third minutes (14:01 and 14:02). The S-transform of the traffic signal is shown in Figure 3.7. The figure shows that S-transform could reveal both instances of the ping-of-death and could precisely identify attack times.

In summary, according to the experimental results, S-transform efficiently differentiated between normal time-frequency behavior and abnormal time-frequency behavior caused by the generated malicious activities. The experimental results also help to confirm that a malicious activity affects traffic behavior and makes

the amplitude (energy) of some frequency component fluctuate. For example, the generated SYN flood and ping-of-death affected medium frequencies. The generated Nmap scans affected high frequencies. In the experiments, S-transform was tested with only malicious traffic. Even though we have not yet tested S-transform with disruptive traffic, but we believe that S-transform is able to reveal abnormal time-frequency behavior caused by disruptive activities. As a result, S-transform is a promising tool for developing an unsupervised network anomaly detection method.

3.3 STAD

This section describes our proposed S-transform-based unsupervised anomaly detection method named STAD. STAD works based on the same ideas of all TFA-based anomaly detection methods described in Section 2.6. STAD consists of three main steps: (1) Conversion of traffic to signal, (2) S-transform, and (3) Detection of intense and hidden anomalies. The details of each step are described below.

3.3.1 Conversion of Traffic to Signal

In the first step, STAD calculates six statistics of traffic data being analyzed, namely packet rate, bit rate, source IP rate, destination IP rate, flow rate, and average flow size rate. The packet rate is calculated by counting the number of all packets of all protocol types at each second. The bit rate is the number of bits per second. The source IP rate is the number of distinct source IPs that are seen per second. The destination IP rate is the number of distinct destination IPs that are seen per second. The flow rate is calculated by adding the number of distinct TCP flows, UDP flows, and ICMP flows together. We empirically define an ICMP flow as a pair of source and destination IPs. For the average flow size rate, it is calculated by dividing the number of packets by the total number of flows that are seen every second. Each statistical data obtained during this step is called traffic signal. Finally, this step is completed by individually subtracting its mean value from each element of the traffic signal in order to remove the DC component.

3.3.2 S-transform

In this step, the time-frequency characteristics of each traffic signal are revealed using S-transform. S-transform with default parameter values analyzes the traffic signal. S-transform's output is stored in an S-transform matrix (ST matrix) of size $m \times n$, where m is the signal's time intervals and n is analyzed frequencies from 0 to $\lfloor m/2 \rfloor$. Each element is an amplitude (energy). For example, if the duration of the packet stream is one minute, the ST matrix has dimension of 60×30. At this step, the frequency behavior over time of the six traffic signals is revealed in six *ST matrices*.

3.3.3 Detection of Intense and Hidden Anomalies

The aim of this final step is to detect anomalous times that contain anomalies. The pseudo-code of this step is described in Algorithm 1. Firstly, STAD focuses on detecting intense anomalies by relying on Time Maximum Amplitude (TMA), Time Amplitude (TA), and Time Variance Amplitude (TVA). The TMA is a vector of the maximum value in each ST matrix column. The TA is a vector of the sum of all values in each ST matrix column. The TVA is a vector of the variance of all values in each ST matrix column. In order to detect suspicious time intervals, a threshold $thres - \alpha$ is applied to the TMA, TA, and TVA. An interval's value that exceeds the $thres - \alpha$ will be labeled as suspicious. Secondly, STAD detects hidden anomalies by using Frequency Amplitude (FA) computed by adding all values of each row of the ST matrix. Next, the FA is divided equally into three parts. Then, each slope angle of each pair of maximum points among the three parts is computed. If the angle exceeds a threshold $thres - \beta$, the ST matrix row vector corresponding to the row number that is equal to the higher maximum value of each part will be selected for detecting suspicious time intervals. Finally, the time intervals which have been labeled as suspicious at least twice are anomalous time intervals. There are two parameters in this step: $thres - \alpha$ for detecting intense anomalies and $thres - \beta$ for detecting hidden anomalies. In this work, the $thres - \alpha$ is empirically set to 1.8×mean of analyzing vector. The $thres - \beta$ is 30.

```
Data: 6 ST matrices
Result: anomalous time intervals carrying intense and hidden anomalies
set all elements of labelCount array to be 0
foreach STmatrix do
    for STmatrix column t ← 1 to m do                    /* initial to detect intense anomalies */
        TMA[t] = max(STmatrix, t);
        TA[t] = sum(STmatrix, t);
        TVA[t] = variance(ST matrix, t);
    end
    labelSuspiciousTimes(TMA); labelSuspiciousTimes(TA); labelSuspiciousTimes(TVA);
    for STmatrix row f ← 1 to n do                       /* initial to detect hidden anomalies */
        FA[f] = sum(STmatrix, f);
    end
    equally split FA into FA_L, FA_M, and FA_H;
    degreeLM = slope(max(FA_L), max(FA_M)); degreeMH = slope(max(FA_M), max(FA_H));
    if (degreeLM≥ thres − β) and (max(FA_L>FA_M)) then
        labelSuspiciousTimes(STmatrix row vector at row number max(FA_L));
    end
    if (degreeLM≥ thres − β) and (max(FA_L<FA_M)) then
        labelSuspiciousTimes(STmatrix row vector at row number max(FA_M));
    end
    if (degreeMH≥ thres − β) and (max(FA_M>FA_H)) then
        labelSuspiciousTimes(STmatrix row vector at row number max(FA_M));
    end
    if (degreeMH≥ thres − β) and (max(FA_M<FA_H)) then
        labelSuspiciousTimes(STmatrix row vector at row number max(FA_H));
    end
end
for i ← 1 to m do                                        /* determine anomalous time intervals */
    if (labelCount[i]≥2) then time interval t is an anomalous time interval;
end
function labelSuspiciousTimes(vector) (   /* function for labeling suspicious time intervals */)
for t ← 1 to m do
    if (vector[t]≥ thres − α) then labelCount[t]++;
end
```

Algorithm 1: Algorithm for detecting intense and hidden anomalies

3.4 Performance Evaluation

We verified the detection performance of STAD by testing it with simulated traffic from the DARPA dataset [43] and real-world backbone traffic from the MAWI dataset [47] described in detail in Section 2.3.3. Furthermore, we compared the results of STAD with an unsupervised DWT-based anomaly detection method proposed by Barford et al. [30]. The details of Barford et al.'s method are described in Section 2.6. We call Barford et al.'s method WTAD. Based on the algorithm in [30], the parameters in WTAD are set as follows. The mother

Table 3.1: Total number of anomalous time intervals detected by STAD and WTAD and the number of overlapping results between STAD and WTAD with DARPA report

Date	STAD		WTAD	
	#alerts	#alerts overlapped with DARPA report	#alerts	#alerts overlapped with DARPA report
Monday	13,490	2,525	18,950	4,382
Wednesday	14,610	2,518	18,500	2,221
Thursday	14,350	3,580	20,110	1,721
Friday	12,860	1,600	15,460	1,338

wavelet used was Daubechies-4. The V-part was constructed by combining level 1 wavelet coefficients and level 3 wavelet coefficients. The anomaly detection threshold was set to 1.8×mean of the V-part. At the end of this section, we investigated the detection time of STAD.

3.4.1 Simulated Attack-present Traffic

We tested STAD with the 4th week traffic data of the DARPA dataset [43] consisting simulated attacks and normal traffic. The details of the DARPA dataset are described in Section 2.3.3. Table 3.1 shows the total number of anomalous time intervals detected by STAD and WTAD. Table 3.1 also shows the total number of overlapping results between STAD and WTAD with the DARPA report. The table indicates that STAD could mostly detect more attack time intervals than WTAD. By looking at the results, we found that STAD could detect simulated DoS and probe attacks which occurred in the traffic traces more precisely than WTAD. Table 3.1 also indicates that both methods reported significantly more time intervals as anomalous. The reasonable causes are shown in Figure 3.8. The first sub-figure of Figure 3.8 is the flow rate signal of Monday traffic. The simulated time intervals are highlighted. The second and third sub-figures present anomalous time intervals reported by STAD and WTAD, respectively. Both of them labeled time intervals during 11:00 to 15:00 as anomalous. By using manual inspection, we found that there was flash crowd activity which caused a large number of connections between internal hosts and external web servers. The sec-

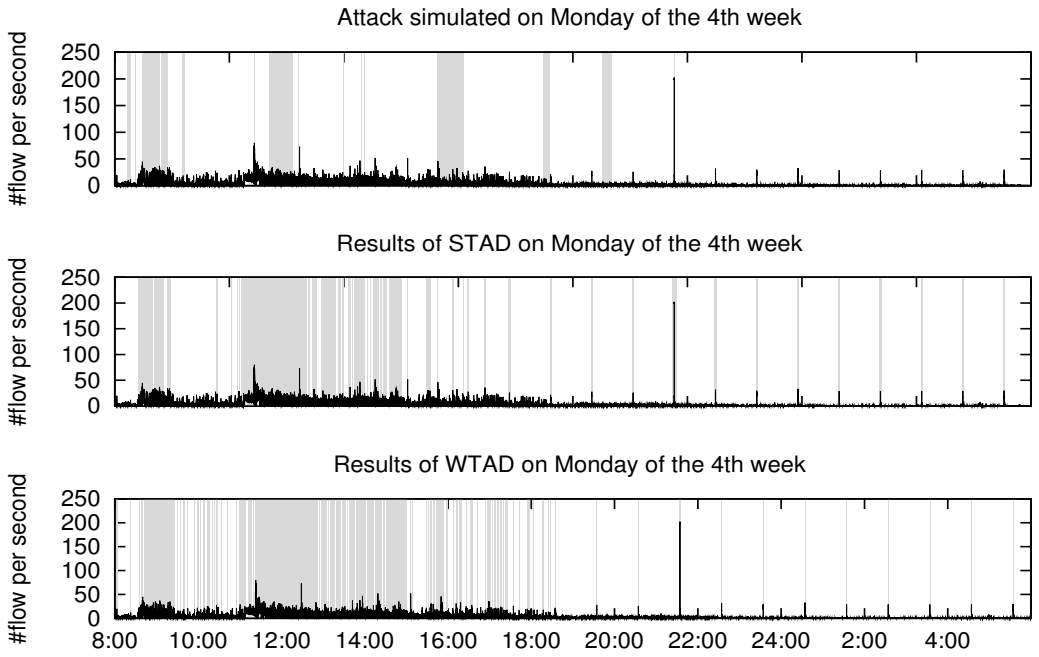

Figure 3.8: Actual simulated attack times in the 4th week's Monday trace in the DARPA dataset (top), anomalous time intervals detected by STAD (middle) and WTAD (bottom)

ond and third sub-figures of Figure 3.8 also show that from about 16:00, both methods labeled few time intervals, which did not carry simulated attacks, as anomalous at every constant interval. Manual analysis confirmed that there was a communication between two hosts at constant intervals in the traffic trace. One of the hosts used DNS port (port number 53) to create a large number of connections compared to the typical behavior which occurred during 22 hours. Furthermore, both STAD and WTAD report that a heavy ICMP packet stream originated from the external network at around 21:30 (a significant spike) as an anomaly. Traffic data from Monday, Thursday, and Friday also contain flash crowds and anomalous DNS traffic that made both methods reported many alerts as shown in Table 3.1. From these results, we can conclude that both STAD and WTAD could detect flash crowds and some traffic that had deviant behavior. However, STAD was better at detecting more attacks, especially DoS and probe attacks, in the DARPA dataset than WTAD.

Table 3.2: The details of the traffic in the four MAWI traces

Date	#flows	#pkts	#bytes	#IPs
1/Aug/2004	107M	819M	2,668M	51M
6/Feb/2009	88M	2,143M	14,253M	47M
6/Apr/2009	67M	1702M	11,192M	44M
2/Jan/2010	75M	1464M	9797M	45M

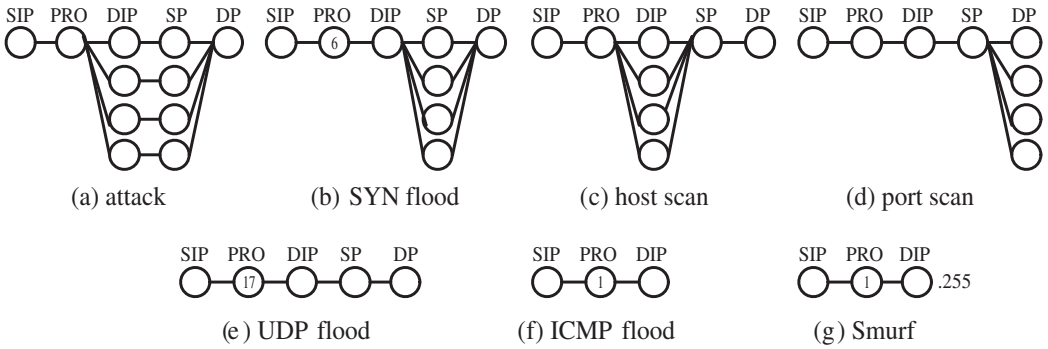

Figure 3.9: Graph models presenting communication behavior of malicious activities

3.4.2 Real-world Traffic

Four backbone traffic traces from MAWI dataset [47] were used to evaluate performance among STAD and WTAD, namely traffic data collected on August 1st, 2004, February 6th, 2009, April 6th, 2009, and January 2nd, 2010. Table 3.2 shows the details of the traffic in the four traces.

Unfortunately, the MAWI dataset lacked anomaly labels of those traffic traces during our testing, therefore we used manual analysis and two signature-based methods from [97, 98] to help to validate the results from STAD and WTAD. The signature-based methods use seven graph models as signatures to detect anomalies. Figure 3.9 shows their pre-defined graph models. Graph model (a) represents the communication behavior of common attacks. Graph model (b) represents SYN flood communication behavior. Graph models (c) and (d) represent host scan and port scan behavior, respectively. Graph models (f) and (g) represent the behavior of ICMP flood and Smurf attacks, respectively. Specifically, the graph model needs a certain detection threshold, therefore we empirically set the

Table 3.3: Total number of anomalous time intervals detected by STAD and WTAD, and the total number of overlapping results between STAD and WTAD with signature-based methods

Date	STAD		WTAD	
	#alert	#alert overlapped with signature-based	#alert	#alert overlapped with signature-based
1/Aug/2004	119	116	200	195
6/Feb/2009	449	443	90	87
6/Apr/2009	431	174	180	65
2/Jan/2010	425	160	100	41

thresholds for the graph models (a), (b), (c), and (d) are 20% of average number of flows per host per second. For graph models (e), (f), and (g), the threshold was 20% of average number of packets per host per second.

Table 3.3 shows the total number of anomalous time intervals detected by STAD and WTAD. Table 3.3 also shows the total number of overlapping results between STAD and WTAD with signature-based methods. Mostly, STAD generated much more alerts than WTAD except on August 1st, 2004. Borgnat et al. [99] reported the August 1st, 2004 trace contained Sasser worm traffic and the signature-based methods classified 772 time intervals (from 900 time intervals) as malicious. On the other hand, STAD and WTAD similarly reported small numbers of alerts. This is due to both methods considering the large Sasser traffic as normal behavior. This is a weakness of unsupervised anomaly detection. For the results on February 6th, 2009, STAD gave more alerts than WTAD. Manual analysis found that from the beginning of the trace until about 14:08, a host intermittently opened many connections to the Virtual Network Computing (VNC) port of many hosts.

Figure 3.10 shows that STAD could detect those malicious activities similar to the signature-based methods, while WTAD missed them. The spikes shown in Figure 3.10 are caused by the Dasher worm. Both STAD and WTAD could precisely detect that significant changes occurred. Figure 3.10 also shows that the signature-based methods reported almost all time intervals as anomalous. This is because there were two anomalies (Figure 3.9(a) and (e)) happening continu-

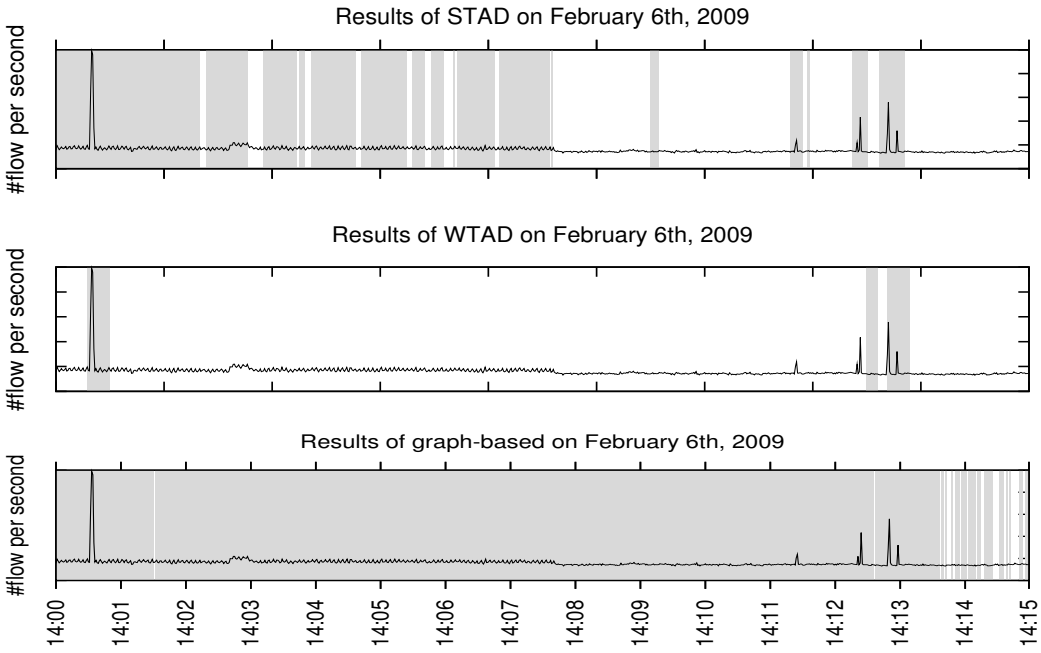

Figure 3.10: Anomalous time intervals in MAWI February 6th, 2009 trace detected by STAD (top), WTAD (middle), and signature-based methods (bottom)

ously throughout the trace. Moreover, the signature-based methods determined anomalies based on pre-defined models without analysis of behavioral change like unsupervised anomaly detection methods. Figure 3.11 shows the results on April 6th, 2009, which consist of several anomalies such as heavy ICMP floods at around 14:01 and a long-lived anomaly displaying behavior like the graph model (a) from about 14:08. The results show that WTAD missed some anomalous instances. By contrast, STAD missed only some instances at the beginning and end of the long-lived anomaly. Figure 3.11 also shows that the signature-based methods did not report that all time intervals from 14:08 are anomalous. This is because the connection numbers associated with the long-lived anomaly were lower than the threshold in some time intervals. This is a drawback of the signature-based methods. More specifically, if a host communication is matched against a graph model but the monitoring value (e.g., number of ports) does not exceed the threshold, the signature-based methods will designate those hosts as legitimate. The results on January 2nd, 2012 are similar to the results on April 6th, 2009. The better

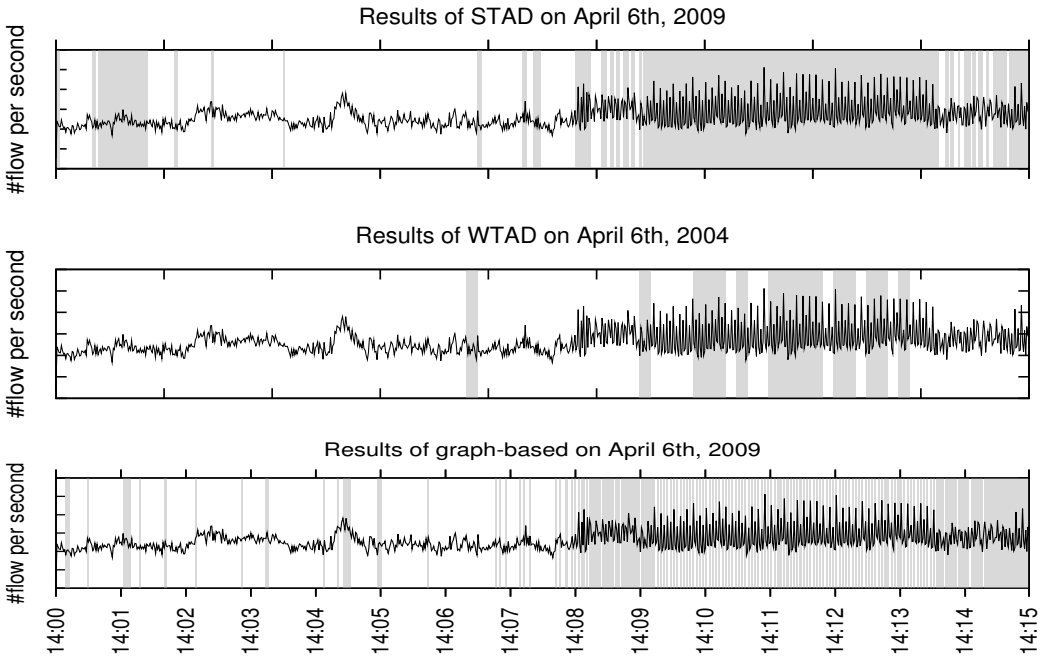

Figure 3.11: Anomalous time intervals in MAWI April 6th, 2009 trace detected by STAD (top), WTAD (middle), and signature-based methods (bottom)

detection performance of STAD comes from the ability to detect hidden brute force SSH attacks which occurred on January 2nd, 2012.

In summary, STAD utilizes S-transform to reveal abnormal time-frequency behavior which is caused by an anomaly in a traffic signal. STAD offers network administrators do not need to be concerned with selecting suitable mother wavelets which actually affect detection performance [32]. Moreover, network operators do not need to be concerned with wavelet coefficients that should be considered. STAD has two simple parameters ($thres - \alpha$ and $thres - \beta$) that are easy to tune. If network administrators want to make STAD more sensitive to anomalies, the network administrators can decrease the $thres - \alpha$ and $thres - \beta$. On the other hand, if $thres - \alpha$ and $thres - \beta$ are increased, STAD becomes less sensitive. According to the experimental results, STAD also offers better detection performance than the DWT-based anomaly detection method [30].

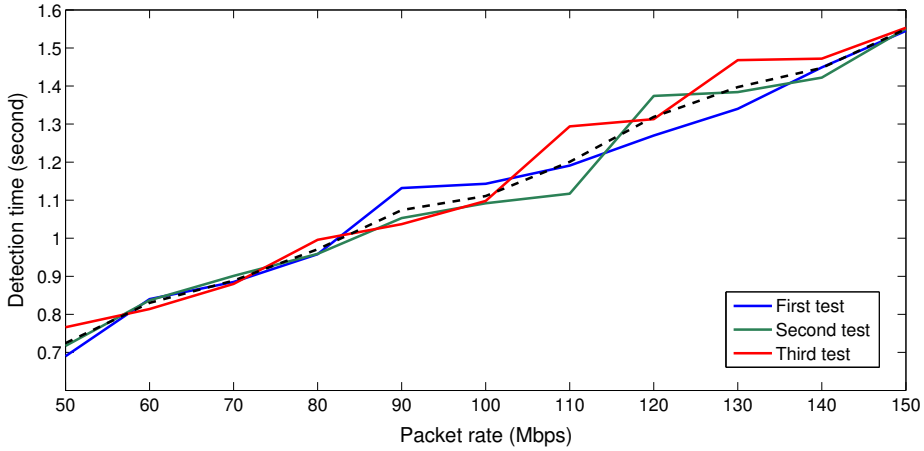

Figure 3.12: Detection time as a function of traffic packet rate. Blue, green, and red lines represent the results obtained from the first, second, and third tests, respectively. Dashed line in black represents the mean of the detection time of the three tests

3.4.3 Detection Time

As speed is a desirable property of network anomaly detection methods as we mentioned in Section 2.2, we performed experiments to measure the detection time of STAD.

In fact, the detection time of STAD mainly depends on the time that is spent in the first step, which is the Conversion of traffic to signal. In the Conversion of traffic to signal step, traffic is converted to six statistics, namely packet, bit, source IP, destination IP, flow, average flow size rates. Calculating the packet, bit, source IP, and destination IP rates of the traffic do not take much time. On the other hand, calculating the flow and average flow size rates takes more time because the traffic contains raw packets, not flow records (processed data). Therefore, STAD has to aggregate the raw packets into flows in order to be able to calculate the flow and average flow size rates. Generally, the aggregation will take a longer time if there are a large number of distinct flows. For the second and last steps of STAD, each step is likely to take a certain time.

In order to control possible factors that may affect the detection time of STAD, we used Iperf [100] to generate traffic with a fixed packet rate. The traffic was generated for three minutes and was collected and stored in a trace by

tcpdump [101]. In the experiment, traffic was separately generated at different packet rates between 50 to 150 Mbps. We applied STAD to each trace and measured the detection time (when the first step starts working to the third step reports anomalies). We note that we modified STAD to convert the traffic to only packet rate. The remaining five statistics were not considered in this experiment because we wanted to know the detection time used for detecting anomalies in a traffic signal. We tested three times for each trace. Figure 3.12 shows the detection time as a function on packet rate of the three tests. The figure also shows the mean of the detection times, represented by the black line. The results indicate that the detection time of STAD increased when the packet rate increased. More specifically, STAD took 0.7 seconds in total to detect anomalies in a three-minute trace containing 50 Mbps traffic. For 150 Mbps traffic, STAD took 1.5 seconds.

3.5 Improving STAD with Sketch Technique

Sketch technique has been applied in intrusion and anomaly detection domains because it provides scalability and helps to improve detection performance [102]. Sketch technique is used to randomly aggregate traffic data into several groups of traffic and then the traffic in each group is investigated. Dividing a group of traffic into smaller groups of traffic and performing anomaly detection analysis of each traffic group helps to increase chances of seeing more anomalies. Moreover, it makes an anomaly detection method able to handle big traffic data. The sketch technique is described in detail in the next section. The authors in [103] utilize the sketch technique to randomly aggregate traffic flows and then use CUSUM to identify change points. The sketch technique was also applied together with PCA [25] and Wavelet transform [38,39,85] to detect unusual events. Specifically, they enhance their old methods by adding traffic sketching before performing anomaly detection.

Previously, we have introduced S-transform as a new efficient TFR tool for revealing anomalies in a traffic signal. In this section, we try to combine the sketch technique with STAD in order to improve the detection performance. In this section, we also evaluated the detection performance of the improved STAD

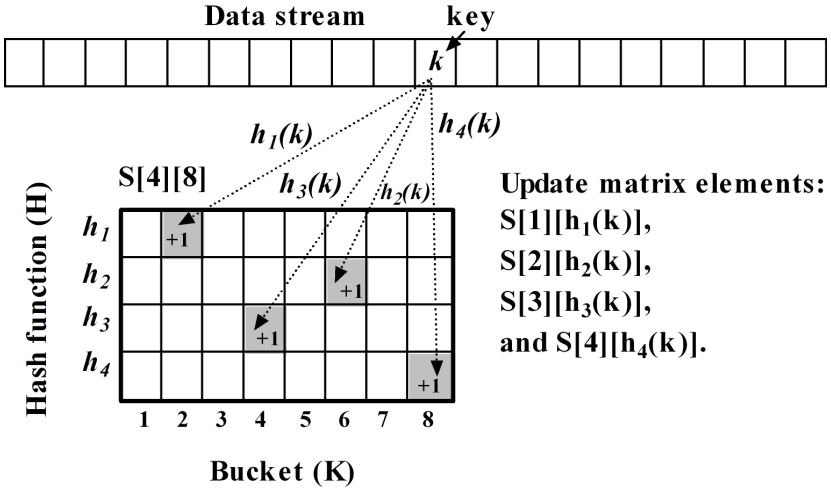

Figure 3.13: A sketch and updating the sketch (H=4 and K=8)

with several traffic traces from the MAWI dataset [47] in terms of accuracy and false positive rates. We also evaluated our improved STAD with traffic containing real-world botnets from the ISOT dataset [54]. Furthermore, we explored the improved STAD's parameters and explored their effects on detection performance. Lastly, we compared the detection performance between the improved STAD with two anomaly detection methods: (1) the stand-alone STAD (our old STAD described in Section 3.3) and (2) WTAD [30] enhanced by the sketch technique.

3.5.1 Background

Sketch

A sketch is a data structure used to summarize a data stream. Technically, a sketch is a two-dimensional $H \times K$ array $S[H][K]$, where each row $(1, 2, .., H)$ is associated with different hash functions, and the columns $(1, 2, .., K)$ are the hash buckets (see Figure 3.13). The matrix element $S[i][j]$ contains the counter associated with the hash bucket j of the hash function i.

To summarize a data stream, an empty sketch $S[H][K]$ is created in which all elements are set to zero. Then, hash functions $h_1, h_2, .., h_H$ linearly hash each key k (e.g., source IP) in the stream. The counters of corresponding matrix elements are updated. Figure 3.13 depicts the summarization of a data stream and storing

the summarized data in a sketch constructed by four hash functions. Each hash table has eight buckets.

Summarizing a data stream using sketch technique has two main advantages: it touches original data only once and uses a fixed amount of memory to store the summarized data. This leads to its application for analyzing and detecting the changes in massive data streams.

Technically, to detect changes in a data stream using the sketch technique, a method constructs a sketch and then continuously updates the sketch when an input key arrives. Until a counter in the sketch reaches a specified value, the method raises an alarm and assumes that a change occurs.

To improve the stand-alone STAD, we apply this sketch technique, but instead of detecting heavy buckets in one sketch, we construct several sketches with the same size at different times. We can then observe the transformation of the sketches. Finally, we detect culprits who made big transformations.

Shannon Entropy

Shannon entropy [63] is a measure of the randomness (uncertainty) of a set of data. Technically, the Shannon entropy of a set of random variable X with possible values $x_1, x_2, ..x_n$ is conventionally defined as:

$$E(X) = -\sum_{i=0}^{n} p_i log_2 p_i, \tag{3.6}$$

where p_i is the probability of value x_i that occurs in the data. The p_i is calculated by the frequency of the value x_i divided by the frequency of all possible n values. The *E(X)* is in the range of zero to $log_2(n)$. The *E(X)* is zero when there is absolutely no randomness. The *E(X)* is close to zero when the data contains a few values. Conversely, the *E(X)* is $log_2(n)$ or close to $log_2(n)$ when every value equally participates in the data.

To apply the entropy concept for unsupervised anomaly detection, researchers typically observe the degree of randomness of a traffic feature distribution (e.g., destination port distribution). Then, they detect large variations of the randomness. For example, in an enterprise network where the IP entropy is normally

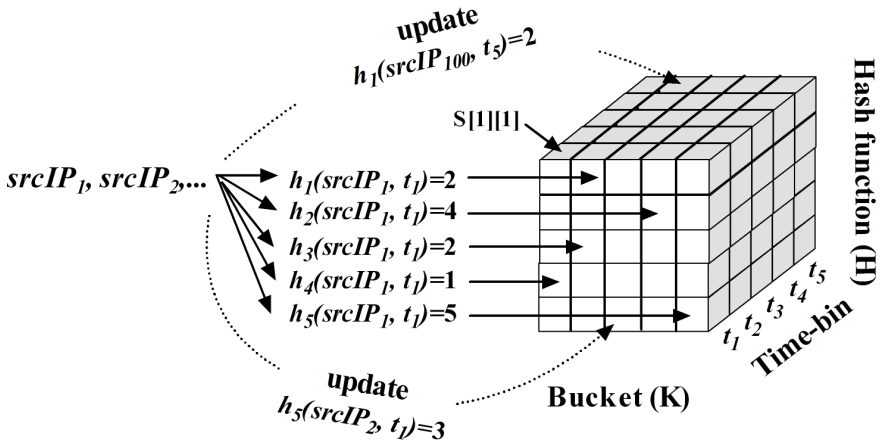

Figure 3.14: Summarizing a traffic stream at different five time intervals (H=5, K=5, and key is the source IP)

high, if an attacker abruptly generates a huge number of packets, an entropy-based detector raises an alert because the entropy decreases.

3.5.2 STAD Improved using Sketch Technique

In this section, we describe the improved STAD that uses the sketch technique to summarize input traffic and then uses S-transform to reveal unusual frequency components of the traffic signals of the summarized traffic. The improved STAD consists of three main steps: (1) Summarizing the traffic with sketch, (2) Detecting suspicious time intervals with S-transform, and (3) Finding the intrinsic culprits of anomalies. Below we describe each step in detail.

Summarizing the Traffic Stream with Sketch Technique

In this step, we summarize the packet stream based on the sketch technique and keep the summarized data in sketches. Specifically, in every constant time interval we perform the following operations. Firstly, we create an empty sketch $S[H][K]$, where H is the number of hash functions used to summarize the traffic and K is the number of buckets per hash table (called sketch size). All elements in $S[H][K]$ are initially set to zero. Secondly, we use the H hash functions to

group the keys (e.g., source IPs in packets) in the time interval to K buckets. h_1 hashes the keys to its buckets, h_2 hashes the same keys to its buckets, and so on. The keys that have the same property (same hash value) are grouped into the same buckets. This means that similar packets are grouped to the same bucket. Thirdly, we compute the entropy of the keys in each bucket using Equation 3.6. Then, we store the entropy value in the associated sketch element. Now we know the entropy of each packet group. For example, the entropy value of the keys that were hashed by h_1 and dropped to its first bucket is stored in $S[1][1]$. To completely discover various kinds of anomalies, we individually use the source IPs, destination IPs, source ports, and destination ports of packets as keys.

The formula to compute p_i (in Equation 3.6) when the keys are source IPs is

$$p_i = \frac{number\ of\ packets\ associated\ with\ source\ IP\ i}{total\ number\ of\ packets}. \tag{3.7}$$

The formula to compute p_i when the keys are destination IPs is

$$p_i = \frac{number\ of\ packets\ associated\ with\ destination\ IP\ i}{total\ number\ of\ packets}. \tag{3.8}$$

The formula to compute p_i when the keys are source ports is

$$p_i = \frac{number\ of\ packets\ associated\ with\ source\ port\ i}{total\ number\ of\ packets}. \tag{3.9}$$

The formula to compute p_i when the keys are destination ports is

$$p_i = \frac{number\ of\ packets\ associated\ with\ destination\ port\ i}{total\ number\ of\ packets}. \tag{3.10}$$

Figure 3.14 illustrates the summarization of a traffic stream at time intervals $t_1, t_2, ..,$ and t_5 using five hash functions. The sketch size is five and the keys are source IPs in the stream. At t_1, the entropy value in $S[1][2]$ is updated because $h_1(srcIP_1)$ is two, $S[2][4]$ is updated because $h_2(srcIP_1)$ is four, $S[3][2]$ is updated because $h_3(srcIP_1)$ is two, and so on. At t_2 to t_5, new four sketches with the same size are created and updated based on new keys in the particular time intervals. After constructing and updating the sketches, we can see how the entropy of keys grouped to the same bucket fluctuates at every time interval as an entropy signal. According to Figure 3.14, we obtain 25 entropy signal which have a length of five.

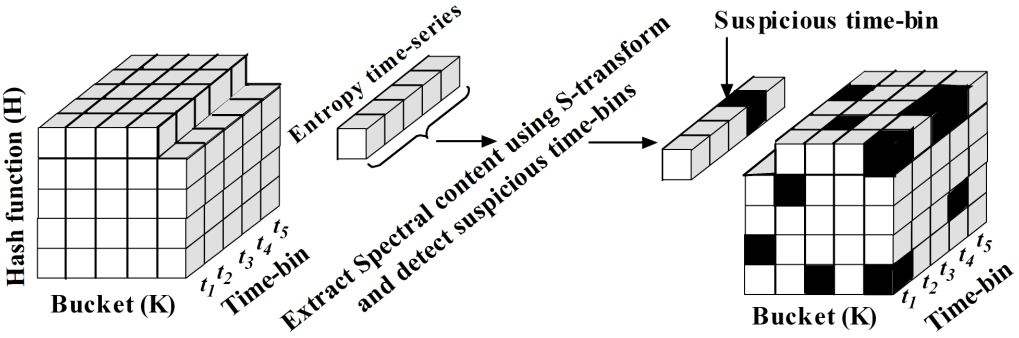

Figure 3.15: Detecting suspicious time intervals in an entropy signal

Detecting Suspicious Time Intervals with S-transform

The goal of this step is to investigate the entropy signal obtained from the previous step and detect suspicious time intervals containing anomalies. Firstly, we remove the DC component of the entropy signal $x(t)$ by using the following equation.

$$x'(t) = x(t) - MEAN \tag{3.11}$$

where $MEAN$ is the mean value of the whole entropy signal $x(t)$. All points in the $x(t)$ are subtracted by its mean. The purpose of this process is to remove the constant values that are added to the signal. These values distort the frequency components in the signal. Secondly, S-transform (with default parameter values described in Section 3.2) converts the $x'(t)$ to a time-frequency domain and produces a matrix (image) like the bottom sub-figure of Figure 3.1. Thirdly, we produce two additional time series from the heap map:

1. a time series that is obtained by vertically summing all matrix elements in the upper half and

2. a time series that is obtained by vertically summing all matrix elements in the lower half of the matrix.

The first time series shows the amplitude variation of the high frequency components and the latter shows the amplitude variation of the low frequency components over time. In this work, we consider time intervals that hold deviant

amplitudes as suspicious time intervals. Thus, in the first time series, we find time intervals that hold amplitude values higher than a threshold *upper_thres* or lower than a *lower_thres*. For the second time series, we find time intervals in the same manner. Figure 3.15 illustrates the processes of detecting suspicious time intervals in the entropy signal. The detected suspicious time intervals are highlighted in black.

Finding the Intrinsic Culprits of Anomalies

In the previous steps, the H summarizers (hash functions) aggregated the traffic according to their own scheme and produced $H \times K$ entropy signals. Then, the method analyzed the signal individually to identify suspicious time intervals which tend to contain anomalies.

Instinctively, all keys that exist in the suspicious time intervals of one summarizer are potential culprits who cause changes. In order to reduce the false positives, we combine the suspicious keys in the detected suspicious time intervals obtained from all summarizers by taking the intersection. The keys in the intersection result are considered intrinsic culprits of anomalies in the input traffic.

3.5.3 Performance Evaluation

Detection Performance

We evaluated the detection performance of the improved STAD with the MAWI [47] and ISOT [54] datasets. The details of both datasets were previously described in Section 2.3.3.

- *MAWI dataset* We evaluated the detection performance of the improved STAD in terms of accuracy and false positive rates with 114 MAWI traces collected from January 1st to April 30th, 2010. Note that six traces during this period are unavailable in MAWI dataset. We also investigated the effect of the method parameters on detection performance. Furthermore, we performed a performance comparison between the improved STAD and our old stand-alone STAD (without sketch) from Section 3.3.

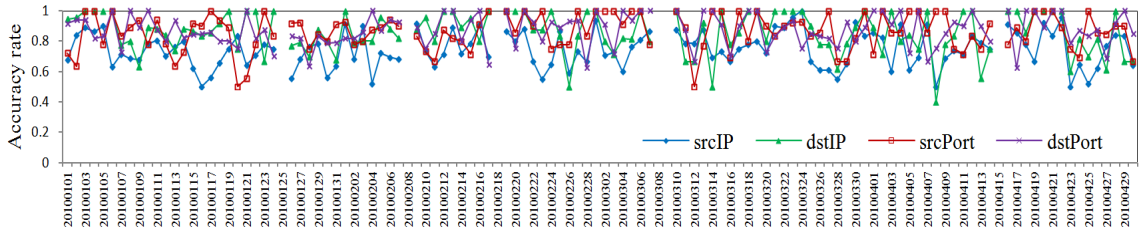

Figure 3.16: Accuracy rate in detecting anomalies in MAWI traces collected from January 1th to April 30th, 2010

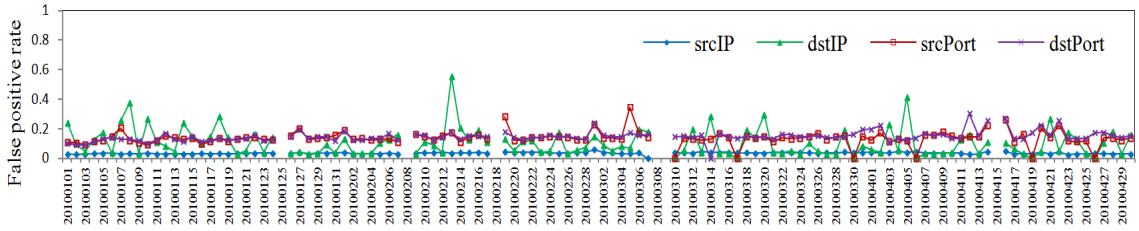

Figure 3.17: False positive in detecting anomalies in MAWI traces collected from January 1th to April 30th, 2010

To verify detection results, we used anomaly labels from MAWILab [50] [49] as a benchmark. The details about MAWILab and MAWILab's anomaly labels are described in Section 2.3.3. We only considered the anomalous labels for verification. The reason we selected the traces from 2010 instead of newer traces is that MAWILab provides complete labels only until April, 2010.

In the experiment, the parameters were set as follows. The H was three. The three hash functions are general purpose hash functions from [86], namely RSHash, PJWHash, and ELFHash. The K was 64 and the time interval size was one second. The keys were source IPs, destination IPs, source ports and destination ports. For the $upper\_thres$ and $lower\_thres$, we set them based on a three-sigma rule. The $upper\_thres$ was $2SD$ and the $lower\_thres$ was $-2SD$. This means that we detected time intervals containing 5% of the amplitude values that were more than $2SD$ or less than $-2SD$.

In this experiment, accuracy and false positive rates of detection were mea-

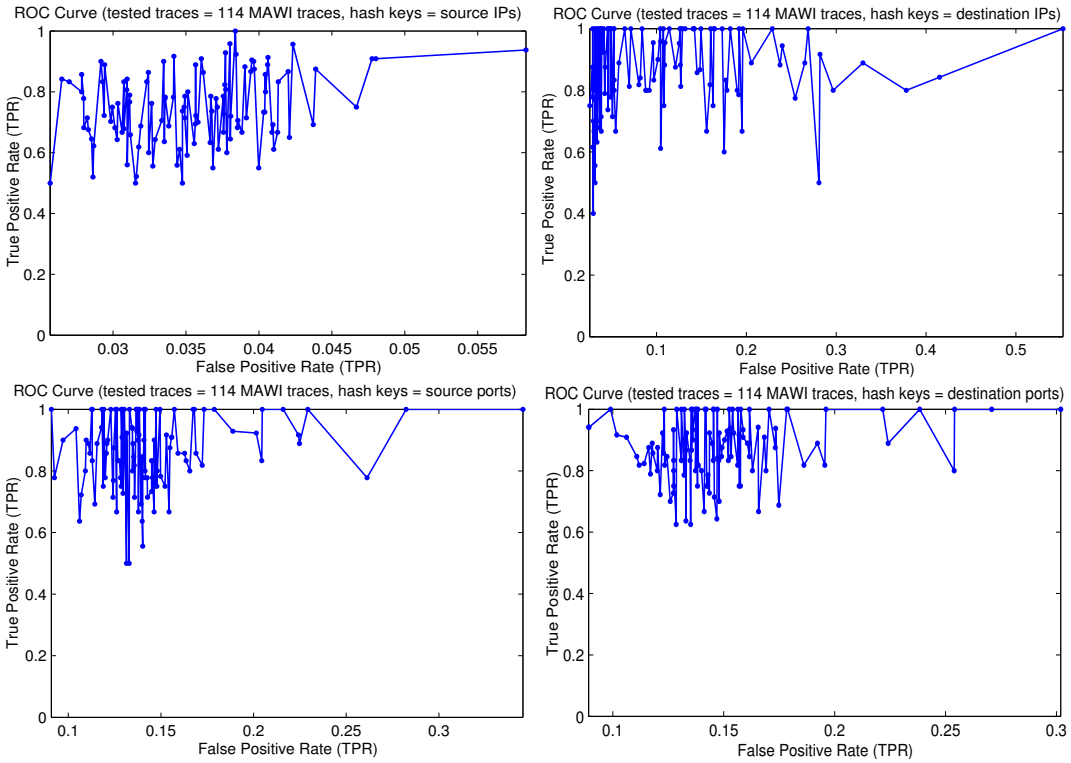

Figure 3.18: Four ROC curves when hash keys are source IPs (left above), destination IPs (right above), source ports (left below), destination ports (right below), respectively

sured. The accuracy rate was computed as the total number of anomalies that were correctly detected by the method divided by the total number of anomalous labels from MAWILab. The false positive rate was computed as the total number of normal instances that were incorrectly detected as anomalies by our method divided by the total number of normal instances in the trace.

Figure 3.16 and 3.17 show the accuracy and false positive rates in detecting anomalous source IPs, destination IPs, source ports, and destination ports in 114 MAWI traces. Figure 3.18 shows the Receive Operating Characteristic (ROC) curves of our improved STAD. The figures indicate that the overall accuracy rate was above 60% and in some traces the improved STAD succeeded in detecting anomalies with 100% accuracy. The average accuracy rates of detected anomalous source IPs, destination IPs, source

Key	172.16.0.11	172.16.0.12	#other detected IPs
Source IP	Detected	Not detected	468
Destination IP	Not detected	Detected	519

Table 3.4: Detection results with the ISOT dataset

ports, and destination ports were 75%, 86%, 86%, and 88%, respectively. The false positive rate of anomalous source IP detection was rather stable at about 3%. The false positive rates in detecting anomalous source and destination ports were about 12%. While the false positive rate of anomalous destination IP detection was not stable. In summary, the improved STAD could moderately detect anomalous source IPs with low false positive rates. For anomalous destination IPs, source ports, and destination ports, the improved STAD could detect anomalies with more precision, but with increased false positive rates.

We also measured the detection time for one analysis. We randomly tested the improved STAD with five MAWI traces. Our detector (that detects anomalies based on the improved STAD) ran on a 10.04 Ubuntu virtual machine with 3.4GHz CPU and 8GB of RAM. We found that it took approximately two minutes to automatically perform the three steps (until printing out intrinsic culprits) described in Section 3.5.2. For example, it took 1.5 minutes to detect anomalies in the trace collected on January 1st, 2011. The January 1st trace size is 1.6 GB containing 22 million packets and the traffic rate is 152 Mbps. Note that the improved STAD's detection time depends on many factors such as traffic characteristics, the number of hash functions, and sketch size.

- *ISOT dataset* The ISOT dataset consists of a large traffic trace containing legitimate and malicious traffic [54, 57]. In this experiment, we cut the traffic that is present in the first 15 minutes of the original trace. By manual inspection, we found that the 15-minute long trace contains two malicious IPs, namely 172.16.0.11 and 172.6.0.12. Both IPs are associated to spam activities as reported in the ISOT Dataset Overview report [57] by the ISOT Research Laboratory. After we obtained the 15-minute long trace,

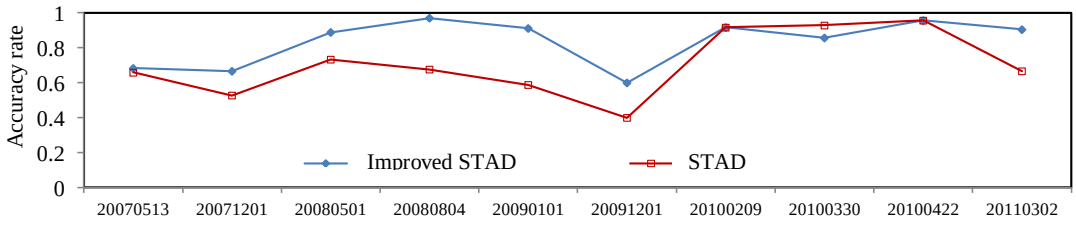

Figure 3.19: Accuracy rate in detecting anomalous source IPs of the improved STAD and stand-alone STAD

we then applied our improved STAD to detect anomalies in the trace. All parameters were set to have the same values similar to the MAWI dataset experiment. In this experiment, we consider only two keys to summarize the traffic in the first step such as source and destination IPs.

Table 3.4 shows the detection results of our improved STAD. The table shows that when we used the source IPs as keys, our improved STAD reported 172.6.0.11 as a malicious IP, but did not mark 172.6.0.12 as malicious. On the other hand, when the keys were destination IPs, STAD determined that 172.16.0.12 was a malicious IP, but missed 172.16.0.11. The table also shows that our improved STAD additionally detected a number of IPs as malicious IPs. More specifically, there were additional 468 IPs that were detected when the used keys were source IPs. There were additional 519 IPs that were detected when the used keys were destination IPs. By manual inspection, we found that all IPs (987 IPs) communicated with either 172.6.0.11 or 172.6.0.12. This means that those 987 IPs were also associated with malicious activities. In summary, with this ISOT dataset, our improved STAD could detect anomalies with 100% accuracy and 0% false positive rates.

Sketch vs. Non-sketch

The improved STAD utilized the sketch technique for improving detection performance and scalability. This section presents the comparison results obtained by comparing the detection performance between the improved STAD with the stand-alone STAD in order to confirm that the sketch technique genuinely en-

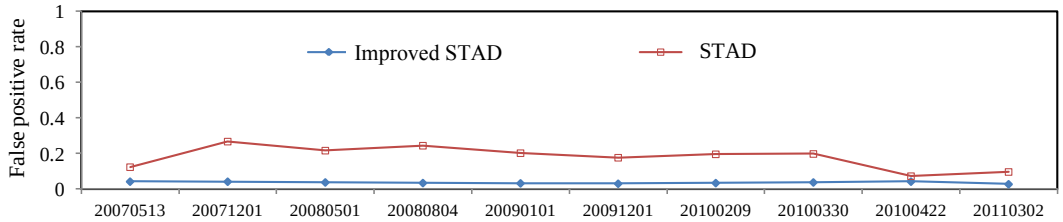

Figure 3.20: False positive rate in detecting anomalous source IPs of the improved STAD and stand-alone STAD

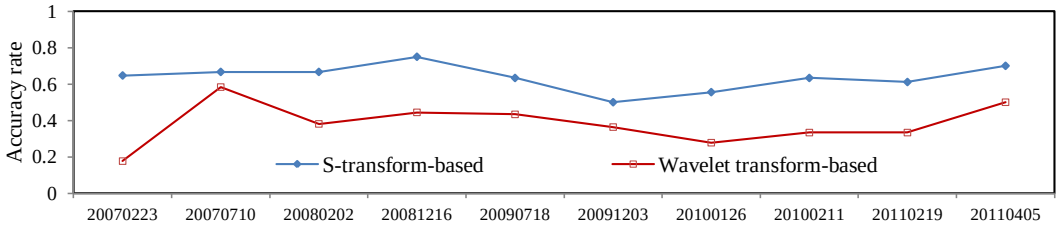

Figure 3.21: Accuracy rate in detecting anomalous source IPs of the improved STAD and Wavelet transform-based method

hances the performance in terms of both accuracy and false positive rates. In this experiment, the improved STAD's parameters were set to the same values described in Section 3.5.3. The keys were source IPs. For the stand-alone STAD, it read a traffic trace and converted the traffic to an entropy signal. The stand-alone STAD then detected suspicious time intervals in the entropy signals based *upper_thres* and *lower_thres*. All source IPs that were present in the detected suspicious time intervals were culprits. The size of time interval was one second. Both methods considered the same frequencies (0 Hz to $\frac{length(x)}{2}$ Hz). Ten MAWI traces collected between 2007 and 2011 were investigated in this experiment. Figure 3.19 shows the accuracy rate comparison between the improved STAD and stand-alone STAD. Figure 3.20 shows the false positive rate comparison of the two methods. The results indicate that the improved STAD mostly outperformed the stand-alone STAD in terms of both accuracy and false positive rates.

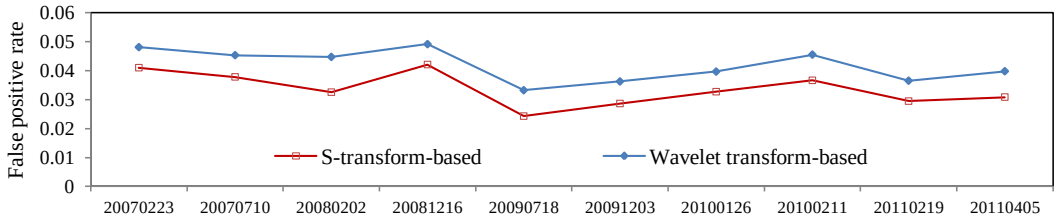

Figure 3.22: False positive rate in detecting anomalous source IPs of the improved STAD and Wavelet transform-based method

S-transform vs. Wavelet Transform

We also compared the results of the improved STAD with the results of the unsupervised sketches and wavelet analysis-based anomaly detection method proposed by Callegari et al. [85]. More specifically, Callegari et al. applied a DWT-based method proposed by Barford et al. [30] in order to detect anomalies in time series given by the temporal evolution of a sketch bucket. The details of Callegari et al.'s method are described in Section 2.6. For the Callegari et al.'s method, we set the number of hash functions, the sketch size, and the time interval size equivalent to the values in the improved STAD. The mother wavelet was Daubechies-4 and the maximum decomposition level was three. The detection threshold was four. In this experiment, both methods investigated the same ten MAWI traces. The accuracy and false positive rates of both detectors are shown in Figures 3.21 and 3.22, respectively. Figure 3.21 shows that our method detected about 64% of anomalies, while the Callegari et al.'s method detected about 37% of anomalies. About the false positive rate (see Figure 3.22), the Callegari et al.'s method generated more false positive alarms than our method.

Parameter Effect

This section describes how each of the improved STAD's parameters affects its detection performance. We explored three method parameters: the number of hash functions, sketch size, and time interval size. Note that we randomly tested the improved STAD with several MAWI traces and the results were consistent. All figures referred to in this section illustrate the results derived from testing on the trace collected on January 1st, 2010.

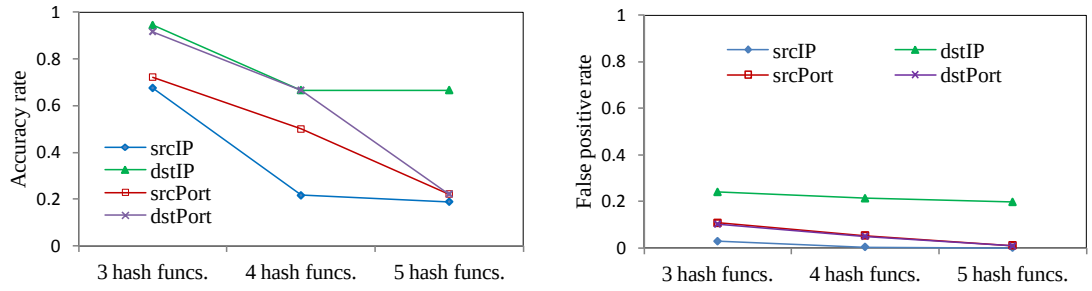

Figure 3.23: Accuracy rates (left) and false positive rates (right) as a function of the number of hash functions

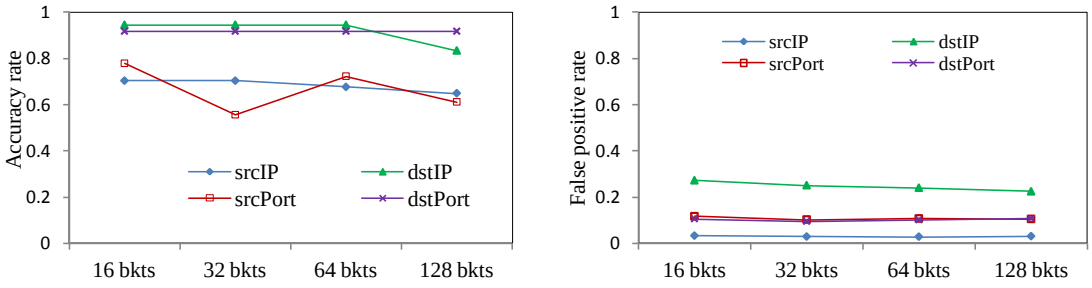

Figure 3.24: Accuracy rates (left) and false positive rates (right) as a function of sketch size

- *Number of hash functions* Figure 3.23 depicts the accuracy (left) and false positive (right) rates as a function of the number of hash functions. We found that using smaller numbers of hash functions provided higher accuracy rates, while the false positive rates slightly decreased. The accuracy rates decreased when we used more hash functions because the method detected the keys in the intersection result.

- *Sketch size* We tested the method with various values of K, such as 16, 32, 64, and 128, and measured accuracy and false positive rates. Figure 3.24 depicts the effect of the sketch size. The results show that the sketch size slightly affected the performance. Moreover, we found that 16 was the best sketch size for detecting anomalies in the MAWI traces because it used the lowest amount of memory and gave the performance similar to the remaining sketch sizes

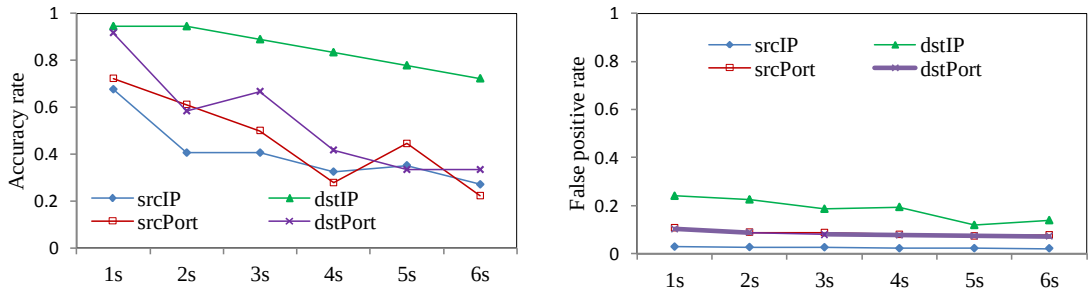

Figure 3.25: Accuracy rates (left) and false positive rates (right) as a function of time interval size

- *Time interval size* Figure 3.25 plots the performance as a function of time interval size. We observed the consequences of adjusting the time interval size from one second to six seconds. The results show that the smaller time interval sizes increased accuracy as well as false positive rates. Thus, to get the best performance from our method, a small time interval size should be taken into account.

In summary, the sketch technique could help to improve the detection performance of our older version of STAD that only performs S-transform analysis. The experimental results also confirm that S-transform analysis outperforms wavelet analysis for detecting network anomalies without prior knowledge of anomalies. Regarding the parameter effect, the improved STAD was slightly sensitive to the number of hash functions and time interval size in terms of accuracy rate. Smaller numbers of hash functions and time interval sizes provided better detection performance.

3.6 Limitations and Discussion

Our improved STAD cannot provide the complete information about an anomalous packet. The complete information includes source IP, destination IP, source port, destination port, and protocol type. Our improved STAD can provide anomalous source IPs if the keys are the source IPs of packets. The method can provide anomalous destination IP if the keys are the destination IPs of packets. This is because our improved method considers only one field in a packet header

(from four fields such as source IP, destination IP, source port, and destination port) as hash hey. This is a limitation of the improved STAD. The solution to address this limitation is described in Section 5.2 in Chapter 5.

Our proposed S-transform-based anomaly detection methods (both stand-alone STAD and improved STAD) are not suited for detecting bulk anomalies similar to other unsupervised anomaly detection methods [3]. A bulk anomaly is an anomaly that occurs on a large scale. Based on the unsupervised anomaly detection's assumptions, our S-transform-based methods and other unsupervised methods interpret that large-scale anomaly as a normal instance. Anomaly detection methods operating in supervised and semi-supervised modes are better suited for detecting bulk anomalies.

Our S-transform-based methods do not reveal the type of a detected anomaly. The reason our methods do not try to identify anomalies is that our methods detect anomalies without any knowledge about anomalies. We do not define the pattern of a DoS attack, or the spread pattern of a worm. We look for changes and assume that those changes were caused by malicious and disruptive traffic. As a result, our methods cannot inform the type of a detected anomaly. However, our methods can inform network administrators of the cause of a detected anomaly. For example, our methods can inform network administrators that a detected anomalous destination IP is associated with a tremendous number of packets.

Sketches theoretically only contain counters and do not preserve original keys in a data stream. Thus, in our improved STAD, to identify the culprits, the original keys are kept temporarily. For real-world applications that must avoid high memory consumption to maintain the original keys, reversible sketches [104] can be efficiently applied. As the focus of our current work is detection performance, we did not utilize or implement the reversible sketches.

We tested our methods with up-to-date traffic from the MAWI dataset because we want to ensure that our methods will be able to detect modern and real-world anomalies. Even though MAWI traffic can be representative of the current state of Internet traffic, the obtained evaluation results in this chapter are still specific to the MAWI traffic and the parameter values which were set during the experiments.

Parameter tuning may be required in order to apply our methods in other

networks and to get expected detection performance.

3.7 Chapter Summary

TFA provides new insights into traffic signals that cannot be obtained from time domain and visual analyses. In this chapter, we proposed S-transform as a new and better tool for unsupervised network anomaly detection. We also developed an S-transform-based anomaly detection method called STAD, which uses S-transform to convert a traffic signal to a time-frequency domain so that we can see anomalous time intervals. STAD then determines anomalous times based on TMA, TA, and TVA vectors. We verified the effectiveness of STAD by testing STAD with real and simulated traffic from MAWI and DARPA datasets, and by comparing the results with Barford et al.'s DWT-based anomaly detection method [30]. The results indicated that STAD outperformed the DWT-based method in terms of detection ability with both datasets. Finally, we measured the detection time of STAD. We found that STAD took 0.7 seconds to detect anomalies in a three-minute trace containing 50 Mbps traffic. For 150 Mbps traffic, STAD took seconds.

Due to the concerns of false alarms and the inability to identify actual anomalies of STAD, we employed the sketch technique to improve our old STAD. More specifically, at every constant time interval, several traffic summarizers, which are technically hash functions, describe the entropy of traffic in their own way. Then, the improved STAD transforms the entropy signal to a time-frequency domain in order to reveal suspicious activities. Finally, the improved STAD determines a suspicious culprit as a real culprit when the suspicious culprit is reported by every summarizer.

Our improved STAD has the following advantages.

1. Our improved STAD operates on aggregate traffic data without deep-packet inspection which enables us to analyze encrypted and massive traffic.

2. Our improved method does not require pre-defined signatures of prospective targets and pre-labeled traffic datasets.

3. Our improved method employs the sketch technique for traffic summarization. These three advantages make our method scalable.

4. As our improved method looks for abnormalities in time-frequency domains, our method can detect extra anomalies apart from time domain analysis-based methods.

In this chapter, we also evaluated the detection performance of the improved STAD with real-world traffic: (1) four months of backbone traffic from the MAWI dataset and (2) traffic from the ISOT dataset. With MAWI traces, we found that the improved STAD could detect anomalies with 60% to 100% accuracy. The false positive rates were between 3% and 12%. With traffic from the ISOT dataset, our improved STAD precisely detected two real malicious IPs with a 0% false positive rate. We also analyzed the effect of the method parameters and the results indicated that our improved STAD was not highly sensitive to parameter tunings. Furthermore, we compared the performance of our improved STAD with two unsupervised anomaly detection methods: (1) our old STAD and (2) sketch and DWT-based methods. The results indicated that our improved STAD outperformed both methods in terms of accuracy and false positives. Lastly, we discussed the limitation of the improved STAD, that is unable to simultaneously report the four fields (namely source IP, destination IP, source port, and destination port) of an anomaly. Furthermore, both old STAD and improved STAD are not suitable to detect bulk anomalies due to the nature of unsupervised detection methods. Another limitation is that both methods cannot currently identify the type of a detected anomalies, but they can tell the abnormal characteristics of a detected anomaly.

Chapter 4

Rényi Divergence-based Anomaly Detection

4.1 Introduction

In statistical anomaly detection methods, the features truly play an important role in detection performance. To the best of our knowledge, most statistical methods mainly consider four widely-used feature distributions, namely the distributions of source IPs, destination IPs, source ports, and destination ports. Some methods consider only one feature or combine them for better detection performance because some features are effective for detecting only certain types of anomalies. This may bring redundant analysis and unnecessary computation.

In this chapter, we propose a new statistical traffic feature and unsupervised statistical anomaly detection method based on the new feature and Rényi divergence [105]. The proposed feature is the port pair distribution that is able to catch various kinds of anomalies which is equivalent in effectiveness to the combination of the four widely-used features mentioned above. Furthermore, the proposed Rényi divergence-based method does not require pre-labeled datasets to find a baseline. Firstly, the method observes the port pair distribution of distinct traffic flows. Secondly, the method computes the Rényi divergence between the port pair distributions of the current and previous time intervals and considers the divergence as the anomaly score of the current time interval. If the score deviates from a dynamic threshold that is calculated based on historical

anomaly scores, the method starts to find anomalous flows. To evaluate the performance of the proposed method, we tested the proposed Rényi divergence-based method with a MAWI trace captured recently in 2014 and compared the results with the results from a statistical anomaly detection method based on the original KL divergence. Furthermore, we compared the performance of the proposed feature with four widely-used features, namely source IP, destination IP, source port, and destination port distributions. The results show that the proposed Rényi divergence-based method outperformed the original KL divergence-based method in terms of accuracy rate while the false positive rates were analogous. Furthermore, the results show that using the port pair distribution could detect more anomalies with a lower false positive rate than the four widely-used features. Finally, we tested the method in terms of potentiality in decreasing the number of false positives. The results indicated that by adjusting only one of the method parameters the false positive rate could be decreased by up to 3 times without affecting the accuracy of detection.

The rest of the chapter is organized as follows. We discuss related work in Section 4.2. Section 4.3 describes the new proposed statistical traffic feature and how to calculate the feature. We present our proposed Rényi divergence-based anomaly detection method in detail in Section 4.5. Section 4.6 explains the evaluation including the traffic trace used, labels, and results. We discuss several issues regarding the proposed method in Section 4.7. We conclude the chapter in Section 4.8.

4.2 Related work

In this section, we describe existing statistical anomaly detection methods relying on the KL divergence or other divergence measures. The KL divergence or relative entropy [108] has been preferred by researchers in the intrusion detection domain due to its lower complexity and superior performance [18]. Basically, KL divergence-based intrusion and anomaly detection methods involve the following four steps.

1. Selecting a statistical traffic feature which is relevant for differentiating targets from other normal traffic.

Considered features	Divergence measures	Targets	Approach
Combined destination port and protocol type distribution [16]	KL divergence	Anomalies	Supervised
Flow duration/octets per flow/packets per flow/IP /port/TCP flag/layer-2 protocol ID distributions [17]	KL divergence	Anomalies	Unsupervised
Source/destination port distributions [19]	KL divergence	Worms	Unsupervised
Source IP/destination IP/source port/ destination port distributions [20]	KL divergence	Scanning events	Unsupervised
Signal power level distribution [18]	KL divergence	Anomalies	Supervised
SIP address distribution [106]	Hellinger divergence	SIP flooding attacks	Supervised
Destination IP distribution [107]	Hellinger and Chi-square divergences	Flooding attacks	Unsupervised
Connection size distribution [21]	KL divergence	DDoS attacks	Unsupervised

Table 4.1: Considered statistical traffic features of existing divergence measure-based methods

2. Calculating two probability distributions: the baseline probability distribution of the feature of normal packets and the probability distribution of the feature of packets being tested (empirical distribution). The baseline probability distribution can be calculated based on a pre-labeled training dataset (supervised) or an unlabeled dataset (unsupervised).

3. Calculating the KL divergence between the two probability distributions and defining it as anomaly score.

4. Detecting targets based on the obtained anomaly score.

In 2005, Gu et al. [16] proposed a supervised anomaly detection method based on KL divergence. In the training phase, they empirically classify packets into 587 packet classes according to the protocol type and destination port. They then train a pre-labeled training dataset to find the baseline distribution of those classes using maximum entropy estimation. In the testing phase, they compare the empirical distribution of new monitoring packets with the baseline distribution using KL divergence. Anomalous packets increase when the KL divergence is larger than a certain threshold.

In the next year, Stoecklin et al. [17] proposed a supervised KL divergence-based method that provides more accuracy than the work proposed by Gu et al. in [16]. They consider several flow features that are likely to be affected by most

anomalies and observe their distributions over time, namely the distributions of flow duration, octets per flow, packets per flow, IPs, ports, TCP flags, and layer-2 protocol IDs. To detect anomalies, they compare the empirical distribution with the baseline distribution and compute a deviation score between the two distributions based on KL divergence. If the score is larger than a threshold, anomalies are identified and reported. Network worms are considered as one of the most critical network anomalies. The authors in [19] use the KL divergence of histograms of source and destination ports to detect attempts of worm propagation in a network. To detect scanning activities, Houerbi et al. [20] use the KL divergence to track changes in the four widely-used features. In [18], the authors measure the power level distribution of wireless signals and use the KL divergence to determine anomalies. Another divergence measure named Hellinger divergence is also used in this domain. In [106], the authors proposed a supervised Hellinger divergence-based method for detecting Session Initiation Protocol (SIP) flooding attacks in a Voice over IP (VoIP) network. They utilize sketch technique in order to improve the detection performance of the original Hellinger divergence-based detection systems. For constructing a sketch, the keys are SIP addresses of the senders and the counters are associated with the number of SIP messages. The baseline distribution (which represents the probability of the SIP messages hashed to each hash bucket) is obtained from a training dataset containing the SIP message history. The empirical distribution is calculated from a testing dataset. An alarm is raised if the Hellinger divergence between the baseline and empirical distributions exceeds a certain threshold, which is estimated based on the Exponentially Weighted Moving Average (EWMA) algorithm. Tajer et al. [107] integrate two divergence measures over sketch data structure in order to detect flooding attacks in any protocols, namely Hellinger and Chi-square divergences. They apply sketch technique to summarize traffic packets according to destination IP addresses. The counters are associated with the number of SYN segments. They then calculate the probability distribution from the sketch data. The Hellinger or Chi-square divergence between two probability distributions of adjacent time intervals is used to determine if a flooding activity occurs. Recently, Shi proposed an unsupervised KL divergence-based method for DDoS attack detection [21]. Her proposed system monitors the connection size distribution of

traffic flows per fixed time interval. The system calculates the KL divergence between the two distributions of the connection size of connections in two successive time intervals. Table 4.1 lists the existing types of distribution used as features for detecting anomalies of existing divergence measure-based methods. The table also shows the divergence measures used for dissimilarity comparison and the approaches for finding baseline distribution.

The difference between our proposed Rényi divergence-based method and the existing divergence measure-based methods is that our method considers a different kind of statistical traffic feature which is the port pair distribution of traffic flows. Our method uses the Rényi divergence, which is a generalization of KL divergence, for anomaly detection process. This allows network administrators to be able to tune their detectors in order to decrease the number of false positives.

4.3 New Feature: Port Pair Distribution

A probability distribution is a function that describes how likely it is to obtain the different possible values of a random variable. In this chapter, we propose the probability distribution of port pairs as a new statistical feature for detecting anomalous flows in a network. The proposed feature is not associated with either the source or destination port of a packet like the source port or destination port distribution. The proposed feature is associated with the combined source and destination ports of a flow. The flow is generally defined as unique 5-tuple information, namely source IP, destination IP, source port, and destination port and protocol type.

Let us define a random variable X as the pair of source and destination ports of a flow. The random variable X can take on classes from domain D ($X \subset D$). The domain D contains 3,034 classes associated with the port pairs of a flow. Table 4.2 shows the defined 3,034 classes and corresponding port pairs. The first 10 classes are associated with flows using system ports (ports 0 to 1,023). We call these flows system-to-system flows. Specifically, the 1st class represents "0-255 & 0-255". It means that a flow with source and destination ports in the range of 0 to 255 will belong to the 1st class. The 2nd represents "0-255 & 256-511", which means that a flow with the source port in the range of 0 to 255 and the

Table 4.2: 3,034 classes of port pairs

Class number	Port pair of a flow	
	Port #1	Port #2
1	0 - 255	0 - 255
2	0 - 255	256 - 511
3	0 - 255	512 - 767
4	0 - 255	768 - 1,023
5	256 - 511	256 - 511
6	256 - 511	512 - 767
*	*	*
*	*	*
11	0 - 255	1024 - 1279
12	0 - 255	1280 - 1535
*	*	*
*	*	*
1019	1024 - 2047	1024 - 2047
1020	1024 - 2047	2048 - 3071
1021	1024 - 2047	3072 - 4095
*	*	*
*	*	*
3034	64512 - 65535	64512 - 65535

destination port in the range of 256 to 511 will belong to the 2nd class. A flow with the source port in the range of 256 to 511 and destination port in the range of 0 to 255 also belongs to this class. The 3rd class represents "0-255 & 512-767", the 4th class represents "0-255 & 768-1023", the 5th class represents "256-511 & 256-511", the 6th class represents "256-511 & 512-767" and so on. Classes 11 to 1,018 are associated with flows using system, user (ports 1,024 to 49,151), and private (ports 49,152 to 65,535) ports. We call these flows client-to-server flows. Class 11 represents "0-255 & 1024-1279", class 12 represents "0-255 & 1280-1535", class 13 represents "0-255 & 1035-1290", and so on. Class 1,019 represents "1024-2047 & 1024-2047", class 1,020 represents "1024-2047 & 2048-3071", class 1,021 represents "1024-2047 & 3072-4095", and so on. The remaining classes are associated with flows using only user and private ports. These flows

are called peer-to-peer flows.

Equation 4.1 describes the formula for finding which class number a system-to-system flow belongs to. Equation 4.2 describes the formula for finding which class number a client-to-server flow belongs to. Equation 4.3 describes the formula for finding which class number a peer-to-peer flow belongs to.

$$class_{s2s} = 3\left\lfloor \frac{lp}{256} \right\rfloor + \left\lfloor \frac{hp}{256} \right\rfloor - (\frac{(\lfloor \frac{lp}{256} \rfloor - 1)(\lfloor \frac{lp}{256} \rfloor)}{2}) + 1, \tag{4.1}$$

$$class_{c2s} = 252(\left\lfloor \frac{lp}{256} \right\rfloor) + \left\lfloor \frac{hp}{256} \right\rfloor + 7, \tag{4.2}$$

$$\begin{aligned} class_{p2p} \quad = \quad & 63(\lfloor \tfrac{lp}{1024} \rfloor - 1) + (\lfloor \tfrac{hp}{1024} \rfloor - 1) - \\ & (\tfrac{(\lfloor \frac{lp}{1024} \rfloor - 1)(\lfloor \frac{lp}{1024} \rfloor)}{2}) + 1019, \end{aligned} \tag{4.3}$$

where $class_{s2s}$ is the class number of a server-to-server flow, $class_{c2s}$ is the class number of a client-to-server flow, and $class_{p2p}$ is the class number of a peer-to-peer flow. lp is the lower port number of the flow which can be either the source or destination port. lh is the higher port number that the flow uses. For example, if a client-to-server flow has the source port 80 and the destination port is 45,000, the lp and hp are 80 and 45,000, respectively.

To calculate the probability distribution of port pairs of a number of flows, $P(X = x)$ or $P(x)$ is defined as the total number of unique flows that use the source and destination ports belonging to the class x divided by the total number of unique flows. For example, the total number of unique flows is 1,000 and the total number of unique flows that their ports belong to class 10 is 200, therefore $P(X = 10) = \frac{200}{1000} = 0.5$.

The benefit of the proposed feature is that it can completely catch unusual activities involving several distinct source or destination ports, such as port scans, flash crowds, and heavy hitters.

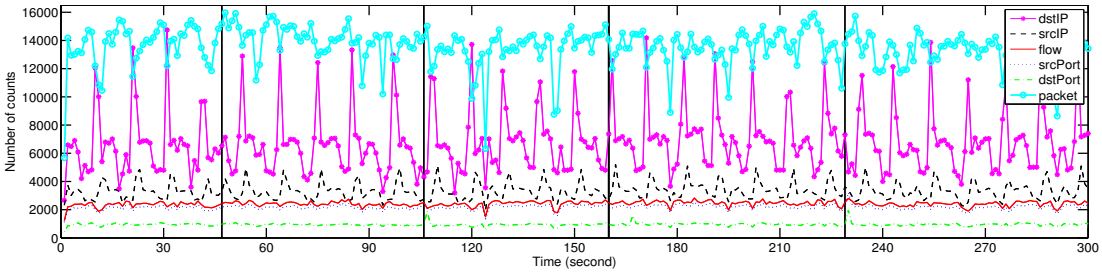

Figure 4.1: Six statistics of the testing traffic consisting of real backbone traffic and four synthetic mimicry anomalies. Top to bottom: the number of packets, the number of destination IPs, the number of source IP, the number of flows, the number of source ports, and the number of destination ports per second. Vertical black lines represent launch times of the four synthetic anomalies which we generated

4.4 Background

4.4.1 Kullback-Leibler Divergence

Definition

Kullback-Leibler (KL) divergence [108] is a measure of dissimilarity between two probability distributions. It is also known as information divergence or relative entropy. The KL divergence of the discrete probability distribution Q from the discrete probability distribution P is

$$D(P||Q) = \sum_{i=1}^{n} p_i log_2 \frac{p_i}{q_i}. \tag{4.4}$$

The KL divergence has the following properties. The KL divergence is non-negative, $D(P||Q) \geq 0$. The KL divergence is not symmetric, $D(P||Q) \neq D(Q||P)$.

Capability of Revealing Hidden Anomalies

In this section, we want to confirm that the KL divergence can be used to reveal anomalies, especially mimicry anomalies that are hard to detect. We expected that KL divergence increases when an anomaly occurs. To test, we generated the

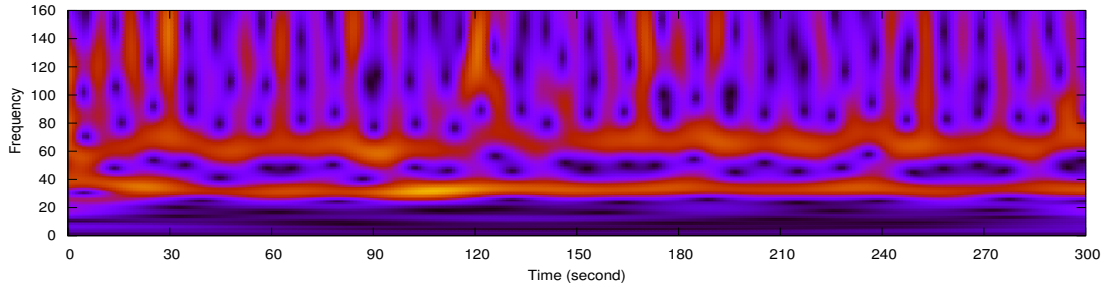

Figure 4.2: S-transform domain of the number of flows per second of the testing traffic shown in Figure 4.1. Black, purple, red, and yellow colors represent very low, low, medium, and very high amplitudes, respectively

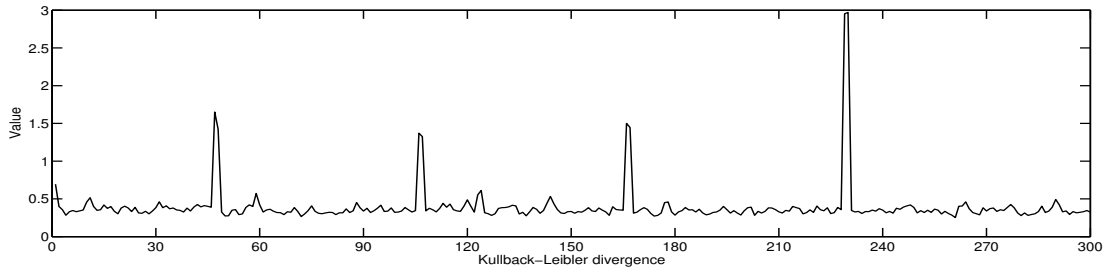

Figure 4.3: KL divergence of the tested traffic

following four synthetic anomalies over MAWI backbone traffic.

At every second, we calculate the KL divergence measure between two probability distributions. The first probability distribution is the port pair distribution of the traffic flow in the previous second. The second probability distribution is the port pair distribution of the traffic flow in current second. In this experiment, we were only interested client-to-server flows, therefore, all client-to-server flows were mapped to class numbers using Equation 4.2.

1. *Mimicry Anomaly I* A single host uses a single well-known source port to communicate to a single destination port (either an user or private port) of a group of hosts. In the experiment, the destination IPs were set randomly using $rand()$ [109]. More specifically, a destination IP consists of four sets of numbers separated by three dots. Each set was computed by $rand() mod 256$. We generated 355 different flows per second which is 5% of the flow rate of the backbone traffic. Anomalies that have behavior similar to mimicry

anomaly I can be DoS attacks using a single port pair.

2. *Mimicry Anomaly II* This anomaly engages with several random destination IPs and destination ports. The number of destination IPs was 355 and they were generated in the same way as mimicry anomaly I was generated. The destination ports were 355 sequential ports. The source IP was a single value and the source port was a well-known port. The behavior of this mimicry anomaly is similar to some DoS attacks and flash crowds.

3. *Mimicry Anomaly III* This type of anomaly is similar to mimicry anomaly II but the destination ports were generated randomly by $rand()mod65536$.

4. *Mimicry Anomaly IV* The last mimicry anomaly is associated with multiple source and destination ports. A source IP was fixed. The destination IPs were generated randomly. The source ports were 355 random well-known ports. The destination ports were random values as well, but with very high port numbers. Anomalies behaving like mimicry anomaly IV include some scannings and attacks.

The four mimicry anomalies were generated at different times over background backbone traffic. The used backbone traffic was a MAWI samplepoint-F traffic captured on March 12th, 2014. The backbone traffic also contains real anomalies. The number of flows per second is 7,000 on average. After we started to release the backbone traffic, we launched the first mimicry anomaly at the 47-second mark. The second, third and fourth anomalies were launched at seconds 106, 160, and 229, respectively. Figure 4.1 shows six statistics of the backbone traffic plus the four synthetic anomalies, namely the number of source IPs, the number of destination IPs, the number of source ports, the number of destination ports, the number of packets, and the number of flows per second. Each synthetic anomaly is explained in detail below.

Figure 4.1 indicates that the behavior of the four synthetic anomalies conformed to the behavior of backbone traffic in terms of the number of source IPs, the number of destination IPs, the number of source ports, the number of destination ports, the number of packets, and the number of flows per second. To double confirm that the four synthetic anomalies carefully mimicked normal traffic in terms of both time and time-frequency domains, we applied S-transform to

discover the time-frequency behavior of the six traffic statistics shown in Figure 4.1. Figure 4.2 shows the time-frequency behavior of the number of flows per second time series discovered by S-transform. Specifically, the black, purple, red, and yellow colors shown in the figure represent very low, low, medium, and very high amplitudes, respectively. From the figure, it can be seen that there are no unusual frequency behaviors present at the launch times of the four synthetic anomalies. We also found that there are no unusual frequency components in the S-transform domain of the remaining statistics which means that these synthetic anomalies are actually unseen.

Figure 4.3 shows the KL divergence calculated at every second for five minutes based on the testing traffic. From the figure, it shows that the KL divergence increases conspicuously when the synthetic anomalies were launched. This confirms that the KL divergence can be used for detecting anomalies.

Another interesting part of the experiment is that the KL divergence also revealed six real anomalies that came with the real-world backbone traffic. We describe the details of the six anomalies below.

1. *Real Anomaly I* The behavior of this anomaly is shown in Figure 4.4(a). The red node in the first column is the principle character of this community. Nodes in the second column are source ports that the principle character used. The third column represents destination ports. Nodes in the last column represent remote hosts that the principle character communicated with. As shown in Figure 4.4(a), the principle character was mainly associated with three well-known ports, namely port 80 (HTTP), 443 (HTTPS), and 21 (SSH). It was also associated with several destination ports and hosts. After doing a deep packet inspection, we found that throughout the duration of the experiment, this principle character communicated with more than 2,000 hosts from different networks with source port 80 and 7,000 different destination ports. It also used ports 443 and 21 to communicate with some hosts. Apart from client-server flows, there were some flows associated with very high port numbers. Most of the connections in this community were uni-directional connections. We assume that this community is an anomaly because most attacks and scannings usually had a uni-directional stream of packets.

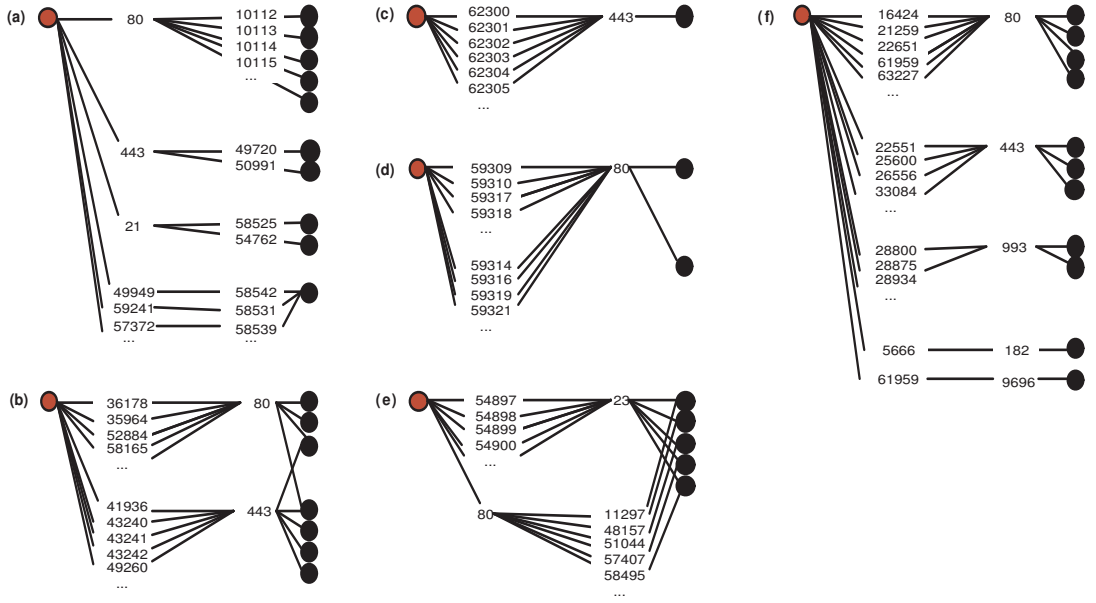

Figure 4.4: Six additional mimicry anomalies with real information about used port numbers. The red node is the principle in the anomaly community. The 2nd, 3rd, and 4th columns are source port(s), destination port(s), and remote host(s), respectively

2. *Real Anomaly II* The hosts in this community exchanged web traffic: HTTP and SSL/TLS traffic. Their communication behavior is shown in Figure 4.4(b). There was a host randomly using more than 30 distinct source ports to communicate to 21 web servers from different networks. This community showed up at different times in the experiment, namely at the 12th, 45th, 63rd, 95th, 123rd, and 250th seconds. This anomaly can be a communication between an attacker and compromised web servers. It is less likely that this community is a genuine flash crowd because the traffic in this community is mostly uni-directional traffic from the principle character to the web servers.

3. *Real Anomaly III* This suspicious anomaly had a behavior similar to real anomaly II but was only associated with HTTPS as shown in Figure 4.4(c). The principle character in this community could be a web user and the remote host could be a web server. The interesting thing is that the 293 source ports used by the user were sequential (from ports 62300 to 62593).

As a result, this event can be an anomaly (e.g.,a DoS attack) because the source ports were ordered.

4. *Real Anomaly IV* We also found a suspicious community at the 152nd second of the experiment and their behavior is shown in Figure 4.4(d). A single host randomly used high source ports to create several connections with two web servers. More specifically, it used 368 different ports with a web server and 381 ports with the other server. This is apparently a flooding attack because all connections in this community were completely half-open connections and this anomaly appeared for a short time.

5. *Real Anomaly V* This suspicious event appeared for 90 seconds from seconds 97 to 187. We found that a host was trying to connect to Telnet servers (providing the service on port 23) as shown in Figure 4.4(e). We inspected the traffic trace exhaustively and found that the host connected to all 256 hosts in a class C network. It used 259 different high source ports and most connections were one-way. In addition, the principal character also acted as a web server and connected to the same 256 hosts. Based on our knowledge, this community is likely abnormal.

6. *Real Anomaly VI* The principle character in this community interacted with several kinds of servers as shown in Figure 4.4(f). It used more than 700 source ports to talk with 193 web servers. Simultaneously, it had encrypted communication with 126 other web servers with 300 different source ports. Ten IMAP servers from the same network were also connected to this community. In addition, we found that the principle character also talked with a remote host via port 9696 which is a port used by many threats so far.

4.4.2 Rényi Divergence

The Rényi divergence [105] is a measure of two probability distributions. The Rényi divergence is similar to the KL divergence. The Rényi divergence between P and Q is defined as

$$D_\alpha(P||Q) = \frac{1}{\alpha - 1} log_2(\sum_{i=1}^{n} p_i^\alpha q_i^{1-\alpha}), \tag{4.5}$$

where α is called the order of the divergence and $\alpha \geq 0$. The n is the number of classes. Theoretically, $D_\alpha(P||Q)$ is the KL divergence ($D_{KL}(P||Q)$), if α is one. The Rényi divergence is also associated with several distance measures depending on the value of α, e.g., associated with Bhattacharyya distance (also known as Hellinger distance) when $\alpha = \frac{1}{2}$ and X^2-divergence when $\alpha = 2$ [110]. The Rényi divergence (except with $\alpha = \frac{1}{2}$) does not theoretically have the property of symmetry.

4.5 Rényi Divergence-based Anomaly Detection Method

This section describes the details of the proposed unsupervised statistical anomaly detection method that is based on the port pair distribution and Rényi divergence. The method consists of three main steps, namely Calculating the port pair distribution, Detecting a Suspicious time, and Identifying anomalous flows.

4.5.1 Calculating the Port Pair Distribution

For every fixed time interval of b, we calculate the probability distribution of port pairs of flows that are seen in such intervals. Section 4.3 explains the definition of the port pair distribution and how to calculate the probability distribution. In this work, we consider only IPv4, TCP, and UDP flows and ignore ICMP packets because we are interested in the ports being used by flows.

4.5.2 Detecting a Suspicious Time

The aim of this step is to investigate whether the current time interval is a suspicious time interval that may containing anomalies. We determine a suspicious time interval based on the distance between the distribution behavior of port pairs of flows in the current time interval and the previous time interval. To calculate the distance between the two distributions, we use the Rényi divergence [105], which is a measure of the difference between two probability distributions P and Q. The Rényi divergence between P and Q is defined by Equation 4.5 described

in Section 4.4.2. This time n in Equation 4.5 is 3,034.

The Rényi divergence (except with $\alpha = \frac{1}{2}$) does not theoretically have the property of symmetry, but we need a proper and symmetric metric (anomaly score) in order to understand how suspicious the current time interval is. Therefore, we define the anomaly score of the current time interval as

$$Anomaly\_score = D_\alpha(P||Q) + D_\alpha(Q||P), \tag{4.6}$$

where P is the probability distribution of the port pairs of the previous time interval. Q is the probability distribution of the port pairs of the current time interval. In other words, the anomaly score is the sum of the distance from Q to P and the distance from P to Q. To finally determine if the current time interval is suspicious or not, we use a dynamic threshold which is freshly calculated every time interval based on the history of anomaly scores. If the current time interval's anomaly score exceeds the current threshold, the current time interval is considered suspicious and then anomalous flows will be identified. In this work, the threshold is decided based on the three-sigma rule and calculated by

$$Threshold = \mu + 2\sigma, \tag{4.7}$$

where μ and σ are the mean and standard deviation of the former B anomaly scores, respectively.

4.5.3 Identifying Anomalous Flows

The aim of the final step is to find anomalous flows that have made the anomaly score greater than the threshold. According to Equation 4.5, $D_\alpha(P||Q)$ can be written as

$$D_\alpha(P||Q) = \frac{1}{\alpha - 1} log_2(p_1^\alpha q_1^{1-\alpha} + p_2^\alpha q_2^{1-\alpha} + ... + p_{3034}^\alpha q_{3034}^{1-\alpha}). \tag{4.8}$$

Since we want to detect flows that made the high anomaly score, we find $topN$ terms (from 3,034 terms) in Equation 4.8 that contain the highest values. We will consider those classes suspicious classes that may contain suspicious flows. For example, if the 2nd term (which is associated with the 2nd class and is $p_2^\alpha q_2^{1-\alpha}$)

Data: an interval number of the current time interval determined as suspicious
Result: a list of anomalous flows
$topN \leftarrow$ the number of considered suspicious classes
for $i \leftarrow 1$ **to** 3034 **do**
    $p_i \leftarrow P(X = i)$ of the current time interval
    $q_i \leftarrow P(X = i)$ of the previous time interval
    $array[i] = p_i^{\alpha} q_i^{1-\alpha}$
end
$suspiciousClasses[] = getIndices(sort(array[], 'highToLow'), topN)$
for $i \leftarrow 1$ **to** $topN$ **do**
    $A \leftarrow$ flows belonging to class $suspiciousClasses[i]$ in the current interval
    $B \leftarrow$ flows belonging to class $suspiciousClasses[i]$ in the previous interval
    $C = C \cup ((A - B) \cup (B - A))$
end
return C

Algorithm 2: Finding anomalous flows after the current time interval was determined as suspicious

produces a value which is one of top $topN$ highest values, the 2nd class and flows belonging to the 2nd class at the current interval are considered suspicious.

Let us define suspicious flows in suspicious classes that are found in the current time interval as set A. Set B contains suspicious flows in the same classes that are found in the previous time interval. Flows that are in $A - B$ as well as in $B - A$ are determined as anomalous flows. Algorithm 2 describes how to find suspicious classes and flows after the current time interval is determined as suspicious.

From Equation 4.6, the anomaly score is derived by the sum of $D_\alpha(P||Q)$ and $D_\alpha(Q||P)$, therefore we also find $topN$ terms in the equation below.

$$D_\alpha(Q||P) = \frac{1}{\alpha - 1} log_2(q_1^{\alpha} p_1^{1-\alpha} + q_2^{\alpha} p_2^{1-\alpha} + ... + q_{3034}^{\alpha} p_{3034}^{1-\alpha}). \tag{4.9}$$

We then also find additional anomalous flows according to Algorithm 2, but replace $array[i] = p_i^{\alpha} q_i^{1-\alpha}$ in Algorithm 2 with $array[i] = q_i^{\alpha} p_i^{1-\alpha}$.

4.6 Performance Evaluation

In this section, we describe the traffic trace and reliable anomaly labels used to evaluate our detection results. Furthermore, we show the results obtained from four experiments. The first sub-experiment is to measure the accuracy and false positive rates in detecting real anomalies of the method. Furthermore, we

compared the performance between the proposed method and a method based on the original KL divergence. In the second sub-experiment, we compared the performance of our proposed feature with the four widely-used features in terms of accuracy and false positive rates. The third sub-experiment is associated with the order parameter (α) and investigates how adjusting α helps to decrease the false positive rate. The last sub-experiment is studying the effect of the threshold on our detection performance.

4.6.1 Used Traffic Trace

The traffic trace used to test the performance of our proposed Rényi divergence-based anomaly detection method is a trace from the MAWI dataset. We selected a samplepoint-F trace captured on June 29th, 2014. Specifically, the trace contains real-world legitimate and malicious traffic generated from several institutes: 46% of packets are TCP packets, 4% are UDP packets, and the rest are ICMP packets. The average number of flows per second is 4,185. The majority of the traffic is web traffic, which is about 40% of all packets in the trace [47]. As reported by MAWILab [49], this trace contains several real-world anomalies such as DoS attacks, network scans, and alpha flows. We describe the details of the anomalies in the trace reported by MAWILab in the next section.

4.6.2 Used Anomaly Labels

We used reliable anomaly labels provided by MAWILab [49] which is a successive project that is especially providing anomaly labels for the MAWI traces. We previously described the method to label the MAWI traffic in Section 2.3.3. In this experiment, we consider only traffic labeled as anomalous and suspicious for detection performance evaluation. Furthermore, we do not considered labels associated with ICMP because our current method do not support yet.

According to MAWILab label file available in [49], the trace totally contains 97 anomalies with 25 different types of anomalies. 30 of them are network scans, namely three *ntscACK*, 16 *ntscSYNt*, one *sntscSYNt* , two *ntscTCPRSTACKrp*, three *ntscTCPICdurp*, one *ntscUDP*, and four *ntscUDPUDPrp* instances. More details about the definition of each anomaly type can be found in [111]. Further-

Divergence measure	Accuracy rate	False positive rate
KL divergence	63.9%	1.6%
Rényi divergence	96.9%	2.1%

Table 4.3: Performance comparison between the original KL divergence and Rényi divergence (b=1, B=30, and order=2)

more, the trace contains two instances of DoS attacks, namely *DDoSSYN* and *ptpDoSSYN*. The trace also contains 10 heavy hitters, namely four *alphfl* and six *alphfHTTP* instances. There are also 28 point-to-multipoint abnormal instances, namely three *mptp*, six *mptpHTTP*, two *mptpla*, one *mptplaHTTP*, four *ptmp*, four *ptmpHTTP*, two *ptmpla*, and six *ptmplaHTTP* instances. Furthermore, there are 27 other anomalous activities, namely 16 *mptmp*, one *point_to_point*, one *ptpposcaUDP*, three *salphafl*, four *ipv46tun*, and two *ipv4gretun* instances.

4.6.3 Experimental Results

This section describes the results of each of the sub-experiments.

We tested the method with the traffic trace described in Section 4.6.1 and used MAWILab labels described in Section 4.6.2 to calculate the accuracy and false positive rates. The accuracy rate is defined as the total number of anomalous flows that were correctly detected by the method, divided by the total number of anomalous flows associated with the 97 anomalies reported by MAWILab. The false positive rate is the total number of normal flows that were incorrectly detected as anomalies by the method, divided by the total number of normal flows in the trace.

Regarding parameter settings, the size of time interval b is one second, therefore at every one second the port pair distribution will be calculated based on the flows in such time intervals. The Rényi divergence and the threshold will also be calculated at every one second. B is 30. This means that the threshold is determined based on the past 30 anomaly scores. $topN$, which is the number of considered suspicious classes, is 10. α in Equation 4.5 is set to two.

In this sub-experiment, we also compared the results of the method with the results of the original KL divergence-based method. The original KL divergence-

based method considers the port pair distribution similar to the method and detected suspicious times and identified anomalous flows in the same manner with the method. Table 4.3 shows the accuracy and false positive rates of our proposed Rényi divergence-based and the original KL divergence-based methods. The results show that the method's accuracy rate was 96.9%, while the accuracy rate of the original KL divergence-based method was 63.9%. For the false positive rate, the method had the false positive rate of 2.1% and the original KL divergence-based method had the false positive rate of 1.6%. The reason the method provided the higher performance is that the Rényi divergence with $\alpha = 2$ theoretically has a lower convexity property than the KL divergence (or Rényi divergence with $\alpha = 1$) [110]. Therefore, with the two same distributions (P and Q), the Rényi divergence is likely to be higher than the KL divergence. This makes the anomaly score of an anomalous time interval more significant when $\alpha = 2$ than when $\alpha = 1$. In summary, with this parameter setting the method could detect anomalies with about 97% accuracy and 2% false positive rates. Furthermore, it outperformed the original KL divergence-based method in terms of accuracy rate, while our false positive rate is slightly more than the original KL divergence-based method.

Port pair vs. Four Widely-Used Feature Distributions

Four statistical traffic features that are widely-used in intrusion and anomaly detection domain are the distributions of source IPs, destination IPs, source ports, and destination ports. To investigate if the use of the port pair distribution as a feature can better detect anomalies than the four widely-used features, we tested the same trace with those four widely-used features again and measured their accuracy and false positive rates. All parameters were set with the same values, namely $b = 1, B = 30$, and $\alpha = 2$. Table 4.4 shows the numbers of anomalies detected by each feature. From 97 anomalies, one proposed feature (port pair distribution) could detect 94 anomalies. The source IP, destination IP, source port, and destination port distributions could detect the anomalies less than the proposed feature, namely 90, 93, 87, and 78 anomalies, respectively. Specifically, there is only the proposed feature that could detect all instances of scanning activities in the trace. The port pairs of the flows could reveal those

		Port pair	Src. IP	Dst. IP	Src. port	Dst. port
Scannings (30)	ntscACK (3)	3	3	3	3	2
	ntscSYNt (16)	16	13	15	12	14
	sntscSYNt (1)	1	0	0	0	1
	ntscTCPRSTACKrp (2)	2	2	2	2	2
	ntscTCPICdurp (3)	3	2	2	2	3
	ntscUDP (1)	1	1	1	1	1
	ntscUDPUDPrp (4)	4	4	4	4	4
	#detected scannings	30	25	27	24	27
DoS attacks (2)	DDoSSYN	1	1	1	1	1
	ptpDoSSYN (1)	0	1	0	1	0
	#detected DoS attacks	1	2	1	2	1
Heavy hitters (10)	alphfl (4)	4	4	4	4	4
	alphfHTTP (6)	6	6	6	6	5
	#detected heavy hitters	10	10	10	10	9
Point-to-multipoint activities (28)	mptp (3)	3	3	3	3	3
	mptpHTTP (6)	6	6	6	6	5
	mptpla (2)	2	2	2	2	2
	mptplaHTTP (1)	0	1	1	1	0
	ptmp (4)	4	4	4	4	3
	ptmpHTTP (4)	4	4	4	4	2
	ptmpla (2)	2	2	2	2	2
	ptmplaHTTP (6)	6	5	6	6	4
	#detected activities	27	27	28	28	21
Others (27)	mptmp (16)	15	15	16	12	10
	point_to_point (1)	1	1	1	1	1
	ptpposcaUDP (1)	1	1	1	1	0
	salphafl (3)	3	3	3	3	3
	ipv46tun (4)	4	4	4	4	4
	ipv4gretun (2)	2	2	2	2	2
	#detected others	26	26	27	23	20
#detected anomalies (from 97)		94	90	93	87	78
Accuracy rate (%)		96.9	92.8	95.9	89.7	80.4
#detected normal flows (from 943k)		19k	92k	147k	71k	135k
False positive rate (%)		2.1	7.6	14.3	9.8	15.6

Table 4.4: The number of anomalies detected by our proposed feature and the four widely-used features including the accuracy and false positive rates of each feature (b=1, B=30, and order=2)

scannings better than the source and destination port distributions because of the characteristic of the MAWI trace. The traffic in the trace was captured at a high-speed backbone network that consists of traffic of various applications and from different sources. In other words, the traffic is rather complex. Therefore, when a scanning occurs the number of ports involved may not be significant enough to skew the distribution. In this case, the scanning will be not seen by the source or

destination port distributions, but it will be seen by the port pair distribution. However, the method missed one DoS attack, one point-to-multipoint abnormal, and one unknown abnormal instances. The source IP and port distributions could slightly detect the DoS attack traffic better than the proposed feature, but they missed many scanning traffic that the proposed feature could detect. From the table, it seems that the destination IP distribution outperformed the other widely-used features and could efficiently detect other anomalous activities. This is due to the fact that most of the anomalies in the trace were associated with many distinct destination IPs. For the destination port distribution, with this trace, the destination port distribution showed the worst performance among the four widely-used features. It could only detect 78 anomalies and missed 19 anomalies. The accuracy rates of the port pair, source IP, destination IP, source port, and destination port distributions were 96.9%, 92.8%, 95.9%, 89.7,%, and 80.4%, respectively.

Through our inspection, we found that there are 943,618 normal flows that are not associated with the 97 anomalies reported by MAWILab. Table 4.4 also shows the total number of normal flows incorrectly detected and considered anomalous by each feature. Based on those numbers, the port pair, source IP, destination IP, source port, and destination port distribution features had the false positive rates of 2.1%, 7.6%, 14.3%, 9.8%, and 15.6%, respectively. In summary, the port pair distribution feature completely outperformed the four widely-used features in detecting anomalies, especially scannings, in the trace in terms of both accuracy and false positive rates.

Tuning the Order for False Positive Reduction

In this sub-experiment, we studied the effect of α of the Rényi divergence on the performance of the method and demonstrate that tuning the order α of the Rényi divergence (Equation 4.5) can help to decrease mistaken detection. Figure 4.5 shows the accuracy and false positive rates as a function of α. The Rényi divergence with $\alpha = 1$ corresponds to the KL divergence. The above sub-figure of Figure 4.5 shows that the α did not affect the accuracy of detection of the method. Specifically, setting α to two provided the highest accuracy rate, which is 96.9%. When α was set to three and four, the accuracy rate slightly decreased

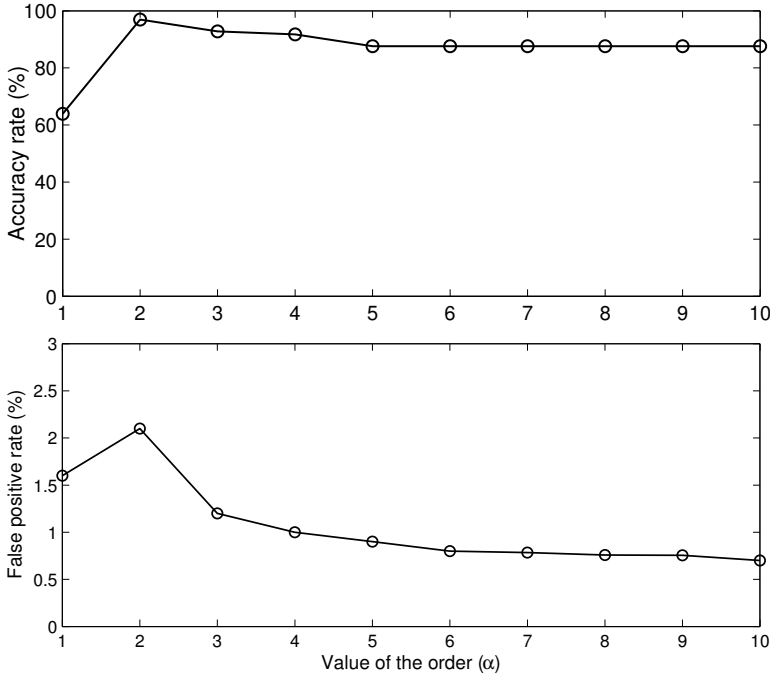

Figure 4.5: Accuracy (above) and false positive (below) rates as a function of the order (b=1 and B=30)

to 92.8% and 91.8%, respectively. The accuracy rate became stable at 87.6% when α was five. On the other hand, the α obviously affected the false positive rate of the method, especially when α was changed from two to three, the false positive rate decreased from 2.1% to 1.2%. The false positive rate continued to slightly decrease when α was set to four. The false positive rate decreased to 0.7% when α was 10. The reason the larger α produced less false positives is that the Rényi divergence with bigger values of α (e.g., $\alpha = 10$) are less convex [110] and increase/decrease more constantly than the Rényi divergence with lower numbers of α. On the other hand, the Rényi divergence with a small α will give more weight to the high probabilities. As a result, the low α may be sensitive and produces more false positives. In summary, the method's accuracy rate is not significantly affected by the α and setting the α of the Rényi divergence to two can allow us see enough difference in port pair distribution. On the other hand, the method's false positive rate is affected by the α. Increasing the value of α could decrease the false positive rate to 0.7%, which is 3 times the false positive rate of $\alpha = 2$,

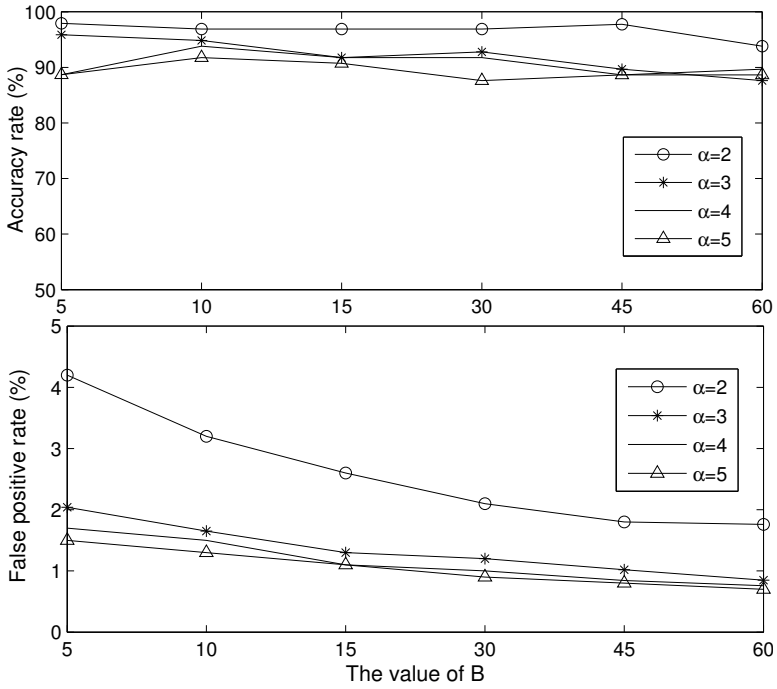

Figure 4.6: Accuracy (above) and false positive (below) rates as a function of B (b=1)

while the accuracy rate was still rather stable.

The Effect of the Threshold

In the Detecting a suspicious time step (see Section 4.5.2), the three-sigma rule-based threshold is determined based on the number of historical anomaly scores. Previously, we empirically set B to 30 as the default value of the method. In this sub-experiment, we want to study the effect of B on the accuracy and false positive rates of the method. We tested the method with the same traffic trace used in the previous sub-experiments and adjusted B with several values, namely 5, 10, 15, 20, 30, 45, and 60, respectively. The study results are shown in Figure 4.6 indicating the accuracy (above sub-figure) and false positive (below sub-figure) rates as a function of the number of considered past anomaly scores (B). The results indicate that with any value of α changing the values of B did not significantly affect the accuracy rate of the method. On the other hand, higher B generated lower false positives. Therefore, to avoid large false positives the

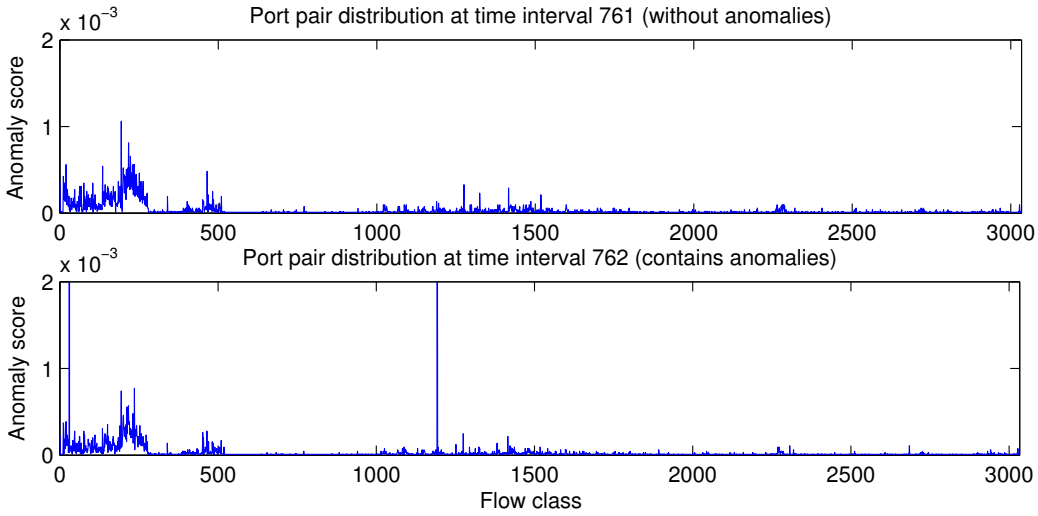

Figure 4.7: Port pair distributions of flows at a non-anomalous (top) and anomalous (bottom) time intervals, respectively

threshold should be determined based on a high number of past anomaly scores.

4.7 Limitations and Discussion

We have suggested the port pair distribution can be a relevant feature for intrusion and anomaly detection, especially detecting scannings in massive and complex traffic. In this chapter, we empirically classify the port pairs into 3,034 classes and raise an anomaly alarm when new monitoring flows belong to unpopular class(s) or many flows belong to the same classes in a short period of time. Therefore, if a new anomaly generates a small number of flows and those flows belong to the same classes with normal traffic in the network, the proposed feature will not be able to detect the anomalous flows. This is a well-known phenomenon of anomaly-based detectors because they detect unusual behavior.

The Rényi divergence boosts the performance of the method in terms of accuracy and false positive compared to the KL divergence-based method. According to the experimental results, $\alpha = 2$ is the best choice in terms of the accuracy rate and $\alpha = 10$ is the best choice in case we want to obtain the minimum false positive rate. Even though $\alpha = 10$ gave an acceptable rate of accuracy, the value

of α must be selected suitably because the Rényi divergence with a low α affects the nature of the distribution of features in traffic.

The experimental results have indicated that the proposed feature and method outperformed the four widely-used features to detect the anomalies in the trace. Some of the widely-used features missed some anomalies (e.g., scannings) because the distribution of the features did not change when those anomalies occurred. For the same reason, the method missed some anomalies, but with a lower number because the tested trace contained diverse types of traffic. Furthermore, most anomalies are scannings, which are anomalies that the proposed feature can efficiently detect. In some different network environments with different behaviors of traffic, the source and destination port distribution may detect the same kinds of anomalies that they have missed in the experiments. The detection performance of statistical methods typically depends on background traffic.

Similar to our proposed S-transform-based anomaly detection method, this Rényi divergence-based method is also not suitable for bulk anomaly detection. In case an anomaly continuously takes place for a long period of time, e.g., for five minutes, the port pair distributions of the traffic flows in those times will look similar. As a result, the Rényi divergence-based anomaly scores will not fluctuate, rendering our method ineffective for bulk anomaly detection.

The output of our statistical method proposed in this chapter provides the anomalies' 5-tuple information, namely source IP, destination IP, source port, destination port, and protocol type (UDP or TCP). This information is basically useful and relevant for technical countermeasures. However, our method cannot give the information about the type of a detected anomaly, e,g., DoS attacks or worms, similar to other unsupervised anomaly detection methods.

In general, statistic methods can be categorized into two types: parametric and non-parametric methods. A parametric method makes an assumption about a statistical property of data such as the mean, variance, or probability distribution of the data. The parametric method detects anomalies based on the assumed statistical property. For example, if the probability distribution does not resemble the assumed probability distribution (e.g., normal distribution), the parametric method will consider that an anomaly takes place. On the other hand, a non-parametric method makes no assumption about the statistical property of

data. The non-parametric method detects anomalies based on observations (e.g., sample data). Our proposed Rényi divergence-based method is a non-parametric statistical method. As a result, our proposed Rényi divergence-based method uses the characteristics of the port pair distribution of flows in previous time intervals as baseline. Our method then detects anomalies when the current port pair distribution significantly differs from the baseline. An example of changes in the port pair distribution caused by anomalies is shown in Figure 4.7. The figure shows the port pair distribution of flows in two different time intervals: 761 and 762 seconds. The time interval 761 does not contain any anomalous flow. The time interval 762 contains anomalous flows that make the distribution differs from the baseline. More specifically, the probabilities of the occurrence of some flow class change because of anomalies. As a result, the Rényi divergence-based anomaly score increases at the time interval 762 and our method starts to identity anomalous flows afterward.

4.8 Chapter Summary

Statistical methods have been proposed for network traffic anomaly detection due to the capacity of real-time detecting of both seen and unseen attacks without attack signatures. In this chapter, we proposed the port pair distribution as a new feature for traffic anomaly detection and introduced an unsupervised statistical anomaly detection method based on the new proposed feature and Rényi divergence. We tested the performance of the proposed feature and method with a real-world backbone traffic trace containing several kinds of anomalies from the MAWI Working Group Traffic Archive. The results indicated that the proposed feature could detect more anomalies than the four widely-used features in terms of accuracy and false positive rates, namely source IP, destination IP, source port, and destination port distributions. The results also show that the method had an accuracy rate of about 97% and significantly outperformed the original KL divergence-based method in terms of accuracy rate while both methods generated similar false positive rates, which were about 2%. We also tested the effectiveness of the method in decreasing the false positive rate and studied the effect of threshold. The results show that the method potentially decreased 3 times of the

false positive rate. By setting α to 10, the method yielded a false positive rate of only 0.7%.

The advantage of the method is that it does not require training as well as a pre-labeled traffic training dataset to find a baseline profile for determining anomalies. However, the method is still effective due to the low number of false positives. By observing only the port pair distribution of flows, the method is able to detect diverse types of anomalies better than the four widely-used features. Therefore, the method helps to eliminate redundant computation. Furthermore, the method gives network administrators the freedom to tune the proposed method to eliminate a number of false positives by simply tuning one parameter. Furthermore, our method provides the anomalies' 5-tuple information (such as source IP, destination IP, source port, and destination port) which is useful and relevant for technical countermeasures. Finally, our method can potentially be improved to be a real-time detection method so that the detection results can be used to employ countermeasures in a timely fashion.

Chapter 5

Conclusion and Future Work

5.1 Dissertation Summary

This dissertation exposed the problems and challenges in network anomaly detection. The dissertation also explored techniques that are employed for network anomaly detection, namely statistical, data mining, and TFR techniques. The dissertation also discussed the strong and weak points of each explored anomaly detection method.

The dissertation proposed an S-transform-based unsupervised anomaly detection method named STAD. STAD utilizes S-transform to reveal anomalies in a traffic signal. Unfortunately, STAD can only identify times that anomalies take place and STAD cannot specify which packet is anomalous. The dissertation thus presents an idea to address those limitations as well as to improve the detection performance of STAD in terms of accuracy and false positive rates. The idea is that input traffic is randomly aggregated into several traffic groups using sketch technique. Then, the traffic in each group is investigated in order to find suspicious activities using S-transform. Finally, the suspicious activities found in each traffic group are combined and treated as anomalies if the suspicious activities are found in every traffic group. Performance evaluations were conducted using simulated traffic data from the DARPA dataset and real-world traffic data from the MAWI and ISOT datasets. The old STAD was compared with a DWT-based unsupervised anomaly detection method [30]. The improved STAD was compared with a sketch and DWT-based unsupervised anomaly detection method [39]. The

results showed that both the old STAD and the improved STAD were superior to the DWT-based methods in terms of both accuracy and false positive rates. The reason that is possible is because S-transform is able to completely discover the behavior of every frequency component of a traffic signal. On the other hand, the DWT summarizes a signal into two scales at a level. The first scale represents low-frequency behavior and the second scale represents high-frequency behavior. This enables our S-transform-based methods to detect anomalies better than DWT-based methods.

In addition, a new statistical feature was proposed for statistical anomaly detection methods. The feature is named port pair distribution. The proposed feature is associated with the source and destination ports of a traffic flow. The proposed feature is different from widely-used features that are associated with the port or IP of a packet. This dissertation presented a performance comparison between our proposed feature and the four widely-used features when they are applied for unsupervised anomaly detection. The four widely-used features are the distributions of source IPs, destination IPs, source ports, and destination ports. The results indicated that the proposed port pair distribution feature obtained the maximum accuracy rate, which is 97%. Furthermore, the proposed port pair distribution feature obtained the minimum false positive rate, which is 2.1%.

Apart from the proposed feature, a non-parametric statistical anomaly detection method was proposed. The proposed method is based on Rényi divergence and the port pair distribution feature. The method monitors the port pair distribution of traffic flows without any assumption about the distribution. The method then starts to seek anomalous flows when the Rényi divergence-based anomaly score exceeds an adaptive threshold. Performance evaluation with a MAWI trace was performed. The results showed that the Rényi divergence-based method could detect anomalies with 96% accuracy and 2% false positive rates. Furthermore, the Rényi divergence-based method outperformed a KL divergence-based method in terms of accuracy rate. By simply adjusting the order parameter α of the Rényi divergence, the false positive rate decreased to 0.7% without affecting the accuracy of detection. This Rényi divergence-based method is more promising for real-time anomaly detection than the proposed S-transform-based

methods. The reason that our method is more promising is because our method requires less computation. However, the old STAD, improved STAD, and Rényi divergence-based methods have some limitations. The aforementioned methods cannot be used for the detection of large-scale anomalies. The reason that the methods cannot be used for the detection of large-scale anomalies is because the methods intuitively designate a large-scale anomaly as a normal event. This limitation is similar to that of other unsupervised anomaly detection methods. Another limitation of our methods is that they cannot specify the type of a detected anomaly. However, they can provide the reason that a detected anomaly triggered an alarm.

In summary, the stand-alone STAD, improved STAD, and Rényi divergence-based methods proposed in this dissertation offer promising ways to detect network anomalies in case network administrators do not have prior knowledge about targets. All methods do not require pre-defined signatures to find attacks like signature-based methods. Furthermore, the methods do not require and rely on pre-labeled training datasets like supervised and semi-supervised methods. As a result, the proposed methods can genuinely detect new and unknown anomalies.

5.2 Recommendations for Future Work

5.2.1 STAD

Online Detection

The improved STAD proposed in this dissertation was designed only for single-shot and offline anomaly detection. In order to implement the improved STAD to detect anomalies online in the real world, the method can be implemented as shown in Figure 5.1. Traffic (packet stream) passes the Sketcher. The Sketcher randomly aggregates the traffic into several sub-traffics using several different hash functions (Sketch technique). The hash functions should be carefully selected because the hash functions affect detection performance [38]. This time the Sketcher can apply a reversible sketch technique (e.g., proposed in [104]). The reversible sketch technique will help to reduce the amount of time to find real culprits in the last step. This technique is suitable for real-time detection. Moreover,

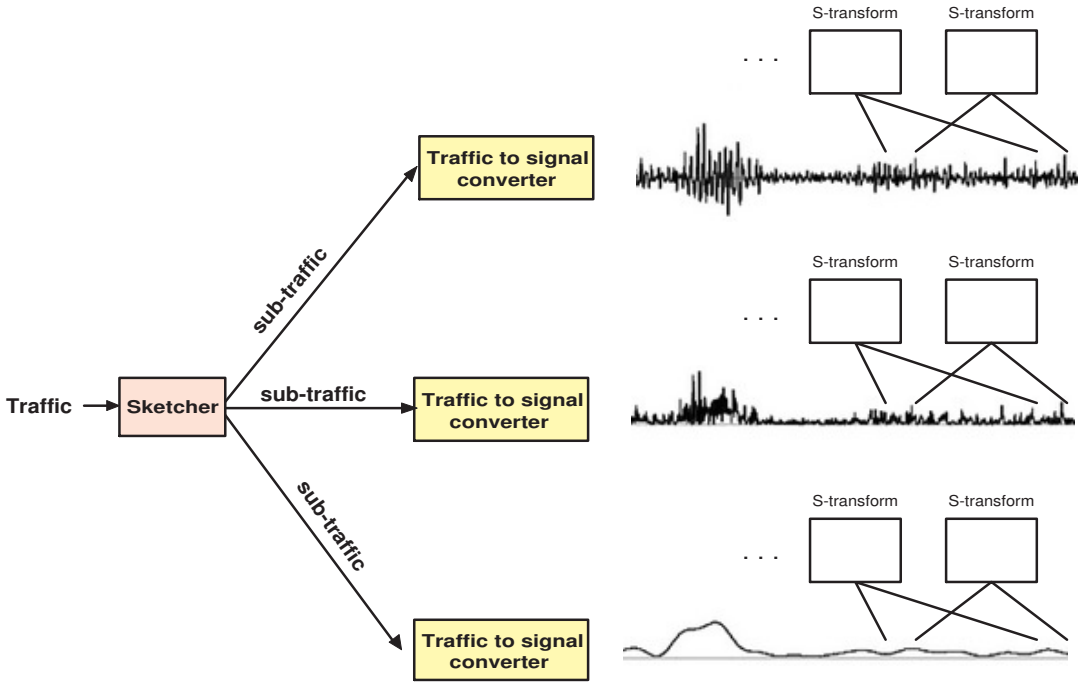

Figure 5.1: Future STAD for online anomaly detection

the reversible sketch technique reduces memory consumption to maintain original keys. Traffic sampling may be applied before the traffic passes the Sketcher. The Traffic to signal converter is responsible for calculating an expected value over time based on the sub-traffic. The output from the Traffic to signal converter is a continuous time-series and can be seen as a continuous signal. In fact, there are many types of time series that can be calculated from the sub-traffic, such as the number of packets per second, the average packet size per minute, the number of connections per second, and the entropy of port distributions per second. Our improved STAD calculates only one entropy signal from a sub-traffic. For example, if the keys used to group traffic are source IPs, our improved STAD generates the entropy signal based on source IPs. If the keys are source ports, the improved STAD generates the entropy signal based on source ports. For the future STAD, we suggest that four types of entropies should be generated from a sub-traffic, namely the entropy of source IP distribution, the entropy of destination IP distribution, the entropy of source port distribution, and the entropy of destination

port distribution. It is because each type of entropy is relevant to detect specific types of anomalies as shown in many works [24, 61, 62]. If we consider only one type, we may miss anomalies that can be seen in other types of entropy signals. We would like to note that Figure 5.1 shows only one signal generated from a sub-traffic for easy understanding. After the Traffic to signal generates a continuous signal, the method segments the signal with a suitable size and transforms the segmented signal to reveal usual frequency behavior using S-transform. The method then detects suspicious time intervals and identifies anomalous source ports, destination ports, source IPs, and destination IPs according to the Detecting suspicious time intervals with S-transform and Finding the intrinsic culprits of anomalies steps described in Section 3.5.2. The signal segmentation will be controlled by a sliding window so that the method can track anomalies and can find the right times to detect the anomalies.

Relevant Anomaly Report

As the improved STAD cannot simultaneously provide all five relevant fields of a detected anomaly, we propose a solution to address this limitation. The future method may combine the five fields in a packet header (source IP, destination IP, source port, and destination port, and protocol type) to a value, which can be either a string or other type of value. The method then uses the value as a hash key. For example, the method may concatenate the source IP and destination IP to a string and the method then uses the string as hash key. As a result, the method knows the five fields of a detected anomaly and informs network administrators of the relevant fields. Before this method is implemented in the real world, more research efforts is needed. For example, the effect of S-transform's parameters, e.g., the number of considered frequencies (f in Equations 3.4 and 3.5) should be studied. After the study, the method may automatically and suitably adjust the parameters.

Change Detection Techniques

We takes advantages of S-transform in order to discover the unusual characteristics of frequencies in a traffic signal. Technically, S-transform's output is a two-dimensional matrix describing the correlation between time and frequency

components. Our STAD (the old STAD and the improved STAD) searches unusual time points by aggregating the data in the matrix and detecting time intervals that are holding unusual frequency characteristics based on a threshold. In other words, we summarize S-transform's output through a time-series and then simply use a threshold in order to detect change points in the time-series. In fact, our simple change point detection technique can be replaced by other sophisticated change point detection techniques such as a relative density ratio estimation-based technique [112] and Bayesian techniques [113, 114]. Moreover, S-transform's output can be considered as an image. Intuitively, the background of the image reflects the typical frequency behavior of the traffic signal. The foreground objects reflect abnormal frequency behavior that can be caused by anomalies. To detect objects in the image, image processing techniques can be applied such as GA-based object selection method [115], Otsu method [116], and Gabor filter. After the objects are detected, we can find the time intervals corresponding to the objects and determine that those time intervals are anomalous time intervals. For example, Pukkawanna et al. [117] uses Otsu's method [116], which is an image segmentation technique, in order to detect abnormal regions in an S-transform's image output.

Snapshot Tracking

Figure 5.1 indicates that STAD focuses on detecting anomalies in the time-frequency snapshot obtained at a certain time interval without considering changes between snapshots. In the future, we may apply an image similarity measuring technique (e.g., histogram-based [118] and generic Fourier descriptor-based [119] image retrieval techniques) to roughly track changes among snapshots (S-transform's image outputs). If the current snapshot looks similar to the recent snapshot, we can assume that there are no new anomalies happening in the current time interval. Therefore, we may not need to investigate the current time interval's snapshot in order to find anomalies. This tracking process may help to reduce the computational cost of STAD and enables STAD to detect anomalies faster.

5.2.2 Rényi Divergence-based Anomaly Detection Method

Detecting Minor Anomalies

Our proposed Rényi divergence-based anomaly detection method finds anomalous flows belonging to *topN* suspicious classes (the algorithm is described in Algorithm 2), if the Rényi divergence-based anomaly score of the current time interval is larger than the current threshold. In the current version, a suspicious class is a class that has a very high dissimilarity between the flows in that class in the current time interval and the flows in that class in the previous time interval. Therefore, the current method finds only significant anomalous flows, while minor anomalous flows are ignored. To solve this problem, the method can be extended as follows. Instead of monitoring the port pair distribution of all flow types, the method separately monitors the port pair distributions of server-to-server flows, client-to-server flows, and peer-to-peer flows. In other words, the method separately monitors three types of distribution. In case the majority of traffic is web traffic, the method may separately monitor and observe the distribution of those web flows. The method then separately detects suspicious times and identifies anomalies for each flow type. Consequently, minor anomalous flows are more likely to be seen. This may enable the method to detect bulk anomalies.

Bin Width Optimization and Multi-scale Distribution Analysis

The statistical feature of traffic that is observed is the distribution of port pairs of flows. To calculate the port pair distribution of flows, we bin flows into 3,034 bins (classes) based on their source and destination ports as shown in Table 4.2. We then calculate the probability of the each flow classes. Our current Rényi divergence-based method set the bin width and the number of bins based on our knowledge and empirical experiments. Technically, too many bins or too few bins result in information loss. More specifically, too many bins (too small bin width) lead to little bias, but too much variability. On the other hand, too few bins (too big bin width) result in too much bias. This is a classic trade-off problem.

In the future, we can optimize the bin width and the number of bins based on traffic data in the network that we will employ the method to. For example, we can apply Freedman-Diaconis rule [120] to find the optimal bin width that

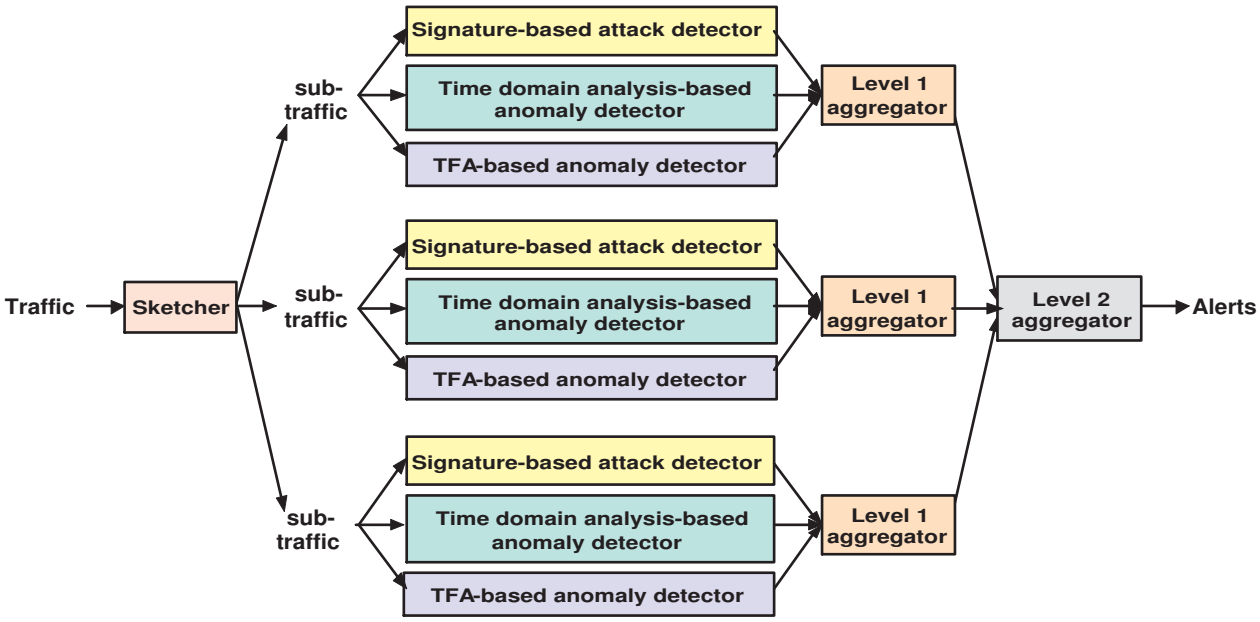

Figure 5.2: Ideal anomaly detection system

balances the bias of the distribution with its variance. The last suggestion for future work is that we may perform a multi-scale distribution analysis in order to increase the method's detection performance in terms of accuracy rate. More specifically, the port pair distribution is calculated based on several different scales (different number of bins and bin widths). The number of bins and the bin width can be determined by using different histogram bin width selection techniques such as Mean Integrated Squared Error (MISE)-based method [121] and decision theoretic approach [122]. Afterward, we apply step two of the method (Detecting a suspicious time step described in Section 4.5) to separately detect suspicious time intervals based on each distribution with a scale.

5.2.3 Future Anomaly Detection System

The improved STAD focuses on TFA, while the Rényi divergence-based method focuses on time domain analysis. Even though both methods are highly effective for the detection of anomalies, we suggest that the best anomaly detection system should perform both time domain analysis and TFA of traffic. Furthermore,

the system should perform signature matching to ensure that the system will not miss known attacks. Performing the signature matching can also help to detect bulk anomalies. The ideal system is illustrated in Figure 5.2. The system consists of six components, namely Sketcher, Signature-based attack detector, Time domain analysis-based anomaly detector, TFA-based anomaly detectors, Level 1 aggregator, and Level 2 aggregator. The Sketcher randomly divides traffic into several sub-traffic similar to the Sketcher mentioned in the future STAD.

Each sub-traffic is separately investigated by the Signature-based attack detector, Time domain analysis-based anomaly detector and TFA-based anomaly detector. The Signature-based attack detector works like a signature-based (misuse-based) NIDS. The Signature-based attack detector identifies attacks using its predefined attack signatures. The Time domain analysis-based anomaly detector detects and identifies abnormal traffic using a baseline (or a classifier) of an effective statistical or data mining-based method. The Time domain analysis-based anomaly detection aims at detecting significant anomalies. We do not suggest employing a supervised method because the supervised method still expects to detect anomalies that it knows. For the TFA-based anomaly detector, it converts the sub-traffic to a signal and detects anomalies present in the time-frequency domain. The TFA-based anomaly detection aims at detecting hidden anomalies that the Signature-based attack detector and Time domain analysis-based anomaly detection do not see. The Level 1 aggregator is responsible for roughly combining the detection results from the three detectors. The Level 1 aggregator also gives the score for each detected anomaly. Finally, The Level 2 aggregator determines the final results based on the scores (obtained from the Level 1 aggregator) and the frequency that each anomaly is found in the sub-traffics. For example, if an anomaly has a high score and appears in every sub-traffics, the Level 2 aggregator treats that anomaly as an anomaly. The Level 2 aggregator has high confidence in that detected anomaly. On the other hand, if an anomaly has a high score but appears only in a sub-traffic, the Level 2 aggregator has low confidence in that detected anomaly. In case the system found a new anomaly, the system should automatically and carefully add the signature corresponding to the new anomaly to the signature-based attack detector's signature database. Furthermore, the system automatically and carefully modifies the baseline or

classifier of the Time domain analysis-based anomaly detector so that the Time domain analysis-based anomaly detector can correctly detect next instances of those new anomalies.

5.2.4 Future Utilization of Proposed Anomaly Detection Methods

Apart from network traffic data, our S-transform-based and Rényi divergence-based anomaly detection methods can be applied to detect abnormalities or changes in several kinds of data, such as stock market data, human behavioral data, audio data, video data, and water level data. For example, our methods can be used to detect price changes in market stock prices. Our methods can be applied in a tsunami early warning system. Moreover, our methods can be a part of a traffic signal control system in order to help to design the traffic signals in a smart city. More specifically, the system collects video streams from traffic cameras. Our methods can be utilized to analyze those data streams and report if a car accident occurs. The system automatically changes the traffic based on the car incident report. In summary, our proposed methods can be applied to any systems involved with changes or are triggered by changes such as security systems.

To apply S-transform to detect anomalies in other kinds of data, the raw data should be in terms of a signal. For example, to detect the early state of a tsunami, the system gathers the sea level changes on an hourly basis. The gathered data is a time series or signal. Then, the system performs an S-transform analysis to detect abnormal changes in the sea level data.

To apply our Rényi divergence-based method, the distribution of a relevant feature of the raw data should be calculated. For example, we would like to know if a new topic emerges on Twitter. We can observe the distribution of words that are being tweeted on Twitter. If the Rényi divergence-based anomaly score that is computed based on the distributions of words is high, we can assume that a new topic may be frequently retweeted or discussed. Furthermore, we can assume that the posting behaviors of Twitter users change. The performance of our Rényi divergence-based method is affected by observed features. Therefore, users need

to understand the data that want to analyze very well and select relevant features that can reach the aims of their systems.

References

[1] M. Roesch. Snort-lightweight intrusion detection for networks. In *Proceedings of USENIX LISA*, 1999.

[2] D. E. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, SE-13(2), 1987.

[3] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: a survey. *ACM Computing Surveys*, 41(3), 2009.

[4] J. H. Kevin and M. W. George. Trends in denial of service attack technology. *CERT Coordination Center*, 2001.

[5] N. Weinstein. Xbox Live, PlayStation network spotty, 2014. Available at `http://www.cnet.com/news/xbox-live-playstation-network-spotty`.

[6] L. Ertöz, E. Eilertson, A. Lazarevic, P. Tan, P. Dokas, V. Kumar, and J. Srivastava. Detection and summarization of novel network attacks using data mining. Technical report, 2003.

[7] D. Goodin. New DoS attacks taking down game sites deliver crippling 100Gbps floods, 2014. Available at `http://arstechnica.com/security/2014/01/new-dos-attacks-taking-down-game-sites-deliver-crippling-100-gbps-floods`.

[8] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. Surveying port scans and their detection methodologies. *Computer Journal*, 54(10), 2011.

[9] J. Reynolds. Helminthiasis of the Internet. RFC 1135, 1989. Available at `https://tools.ietf.org/html/rfc1135`.

[10] D. Moore and C. Shannon. Code-Red: a case study on the spread and victims of an internet worm. In *Proceedings of IMW*, 2002.

[11] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash crowds and denial of service attacks: characterization and implications for CDNs and web sites. In *Proceedings of WWW*, 2002.

[12] G. Iannaccone, C. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot. Analysis of link failures in an IP backbone. In *Proceedings of IMW*, 2002.

[13] A. S. Gary and K. B. Hsiao-hua. Spectral imaging for remote sensing. *Lincoln Laboratory Journal*, 14(1), 2003.

[14] J. A. Barria and S. Thajchayapong. Detection and classification of traffic anomalies using microscopic traffic variables. *IEEE Transactions on Intelligent Transportation System*, 12(3), 2011.

[15] E. Y. Chen. Detecting TCP-based DDoS attacks by linear regression analysis. In *Proceedings of ISSPIT*, 2005.

[16] Y. Gu, A. McCallum, and D. Towsley. Detecting anomalies in network traffic using maximum entropy estimation. In *Proceedings of IMC*, 2005.

[17] M. Stoecklin. Anomaly detection by finding feature distribution outliers. In *Proceedings of CONEXT*, 2006.

[18] M. Afgani, S. Sinanovic, and H. Haas. Anomaly detection using the Kullback-Leibler divergence metric. In *Proceedings of ISABEL*, 2008.

[19] S. A. Khayam, H. Radha, and D. Loquinov. Worm detection at network endpoints using information-theoretic traffic perturbations. In *Proceedings of ICC*, 2008.

[20] K. R. Houerbi, K. Salamatian, and F Kamoun. Scan surveillance in internet networks. *Networking*, 2009.

[21] S. Hongxia. Adaptive packet context-constrained KL-divergence model for intrusion detection. *Networks*, 9(8), 2014.

[22] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *Proceedings of SIGCOMM*, 2004.

[23] A. Lakhina, M. Crovella, and C. Diot. Characterization of network-wide anomalies in traffic flows. In *Proceedings of IMC*, 2004.

[24] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. Technical report, 2005.

[25] X. Li, F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccrone, and A. Lakhina. Detection and identification of network anomalies using sketch subspaces. In *Proceedings of IMC*, 2006.

[26] Y. Kanda, R. Fontugne, K. Fukuda, and T. Sugawara. ADMIRE: anomaly detection method using entropy-based PCA with three-step sketches. *Computer Communication*, 36(5), 2013.

[27] G. Münz, S. Li, and G. Carle. Traffic anomaly detection using K-means clustering. In *Proceedings of GI/ITG-Workshop MMBnet*, 2007.

[28] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion detection with unlabeled data using clustering. In *Proceedings of CSS workshop DMSA*, 2001.

[29] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In *Proceedings of ICML*, 2000.

[30] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. In *Proceedings of IMC*, 2002.

[31] L. Li and G. Lee. DDoS attack detection and wavelets. *Telecommunication System*, 28(3,4), 2005.

[32] C. Huang, S. Thareja, and Y. Shin. Wavelet-based real time detection of network traffic anomalies. *International Journal of Network Security*, 6(3), 2008.

[33] W. Lu and A. A. Ghorbani. Network anomaly detection based on wavelet analysis. *EURASIP Journal on Advances in Signal Processing*, 2009(4), 2009.

[34] M. Salagean and I. Firoiu. Anomaly detection of network traffic based on analytical discrete wavelet transform. In *Proceedings of COMM*, 2010.

[35] A. Dainotti, A. Precape, and G. Ventre. NIS04-1:wavelet-based detection of DoS attack. In *Proceedings of GLOBECOM*, 2006.

[36] J. Gao, G. Hu, X. Yao, and R. K. C. Chang. Anomaly detection of network traffic based on wavelet packet. In *Proceedings of APCC*, 2006.

[37] R. Fontugne, T. Hirotsu, and K. Fukuda. An image processing approach to traffic anomaly detection. In *Proceedings of AINTEC*, 2008.

[38] S. Pukkawanna and K. Fukuda. Combining sketch and wavelet models for anomaly detection. In *Proceedings of ICCP*, 2010.

[39] C. Callegari and S. Giordano. On the use of sketches and wavelet analysis for network anomaly detection. In *Proceedings of IWCMC*, 2010.

[40] S. Pukkawanna, Y. Kadobayashi, and S. Yamaguchi. Network-based mimicry anomaly detection using divergence measures. In *Proceedings of ISNCC*, 2015.

[41] B. Claise, G. Sadasivan, V. Valluri, and M. Djernaes. Cisco systems NetFlow services export version 9. RFC 3954, 2004. Available at `https://tools.ietf.org/html/rfc3954`.

[42] M. Tavallaee, N. Stakhanova, and A. A. Ghorbani. Toward credible evaluation of anomaly-based intrusion-detection methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 40(5), 2010.

[43] R. Lippmann, J. Haines, D. Fried, J. Korba, and K. Das. The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 34(4):579–595, 2000.

[44] C. Callegari. Statistical approaches for network anomaly detection, 2009. A tutorial in ICIMP Conference.

[45] J. McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA off-line intrusion detection system evaluation as performed by Lincoln laboratory. *ACM Transactions on Information and System Security*, 3(4), 2000.

[46] M. V. Mahoney and P. K Chan. An analysis of the 1999 DARPA/Lincoln laboratory evaluation data for network anomaly detection. In *Proceedings of RAID*, 2003.

[47] MAWI traffic archive. Available at `http://mawi.wide.ad.jp`.

[48] K. Cho, K. Mitsuya, and A. Kato. Traffic data repository at the WIDE project. In *Proceedings of USENIX ATEC*, 2000.

[49] MAWILab. Available at `www.fukuda-lab.org/mawilab`.

[50] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda. MAWILab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In *Proceedings of CONEXT*, 2010.

[51] Abilene data. Available at `http://cs-www.bu.edu/fac/crovella/abilene-distro.tar`.

[52] CAIDA datasets. Available at `http://www.caida.org/data/overview`.

[53] The third international knowledge discovery and data mining tools competition dataset (KDD Cup 1999 data). Available at `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`.

[54] ISOT dataset. Available at `http://www.uvic.ca/engineering/ece/isot/datasets`.

[55] French Chapter of the Honeypot Project. Available at `https://www.honeynet.org/chapters/france`.

[56] LBNL enterprise trace repository. Available at `http://www.icir.org/enterprise-tracing`.

[57] ISOT dataset overview. Technical report, 2010. Available at `http://www.uvic.ca/engineering/ece/isot/assets/docs/isot-datase.pdf`.

[58] J. D. Brutlab. Aberrant behavior detection in time series for network monitoring. In *Proceedings of USINEX LISA*, 2000.

[59] M.-S. Kim, H.-J. Kang, S.-C. Hung, S.-H. Chung, and J. W. Hong. A flow-based method for abnormal network traffic detection. In *Proceedings of NOMS*, 2004.

[60] A. Soule, K. Salamatian, and N Taft. Combining filtering and statistical methods for anomaly detection. In *Proceedings of IMC*, 2005.

[61] A. Wagner and B. Plattner. Entropy based worm and anomaly detection in fast IP networks. In *Proceedings of WETICE*, 2005.

[62] G. Nychis, V. Sekar, D. G. Andersen, H. Kim, and H. ZhangS. An empirical evaluation of entropy-based traffic anomaly detection. In *Proceedings of IMC*, 2008.

[63] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3), 1948.

[64] M. Thorup and Y. Zhang. Tabulation based 4-universal hashing with applications to second moment estimation. In *Proceedings of SODA*, 2004.

[65] H. Ringberg, A. Soule, J. Rexford, and C. Diot. Sensitivity of PCA for traffic anomaly detection. In *Proceedings of SIGMETRICS*, 2007.

[66] N. E. Moussaid, A. Toumanari, and M. Elazhari. Intrusion detection based on clustering algorithm. *International Journal of Computer Science, Engineering and Applications*, 2(3), 1956.

[67] A. P. Muniyandia, R. Rajeswarib, and R. Rajaramc. Network anomaly detection by cascading K-means clustering and C4.5 decision tree algorithm. In *Proceedings of ICCTSD*, volume 30, 2011.

[68] M. Solanki and V. Dhamdhere. Intrusion Detection System by using K-means clustering, C4.5, FNN, SVM classifier. *International Journal of Emerging Trends & Technology in Computer Science*, 3(6), 2014.

[69] S. K. Chaturvedi, V. Richariya, and N Tiwari. Anomaly detection in network using data mining techniques. *International Journal of Emerging Technology and Advanced Engineering*, 2(5), 2012.

[70] S. A. Mulay, P. R. Devale, and G. V. Garje. Intrusion detection system using support vector machine and decision tree. *International Journal of Computer Applications*, 3(3), 2010.

[71] L. Portnoy. Intrusion detection with unlabeled data using clustering. Master's thesis, Columbia University, United States, 2000.

[72] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of BSMSP*, 1967.

[73] A. Magnaghi, T. Hamada, and T. Katsuyama. A wavelet-based framework for proactive detection of network misconfigurations. In *Proceedings of SIGCOMM*, 2004.

[74] G. Carl, R. R. Brooks, and S. Rai. Wavelet based denial-of-service detection. *Computers & Security*, 25(8), 2006.

[75] W. Lu. *An unsupervised anomaly detection framework for multiple-connection based network intrusions*. PhD thesis, Department of Electrical and Computer Engineering, University of Victoria, Canada, 2005.

[76] D. Gabor. Theory of communication. *Journal of the Institution of Electrical Engineers*, 93(26), 1946.

[77] A. Haar. Zur theorie der orthogonalen funktionen systeme. *Mathematische Annalen*, 69, 1910.

[78] C.-M. Cheng, H. T. Kung, and K.-S. Tan. Use of spectral analysis in defense against DoS attacks. In *Proceedings of GLOBECOM*, 2002.

[79] V. A. Siris and F. Papagalou. Application of anomaly detection algorithms for detecting SYN Flooding attacks. In *Proceedings of GLOBECOM*, 2004.

[80] UCLA computer science department packet traces. Available at `http://lever.cs.ucla.edu/ddos/traces`.

[81] E. Bacry. LastWave. Available at `http://www.cmap.polytechnique.fr/~bacry/LastWave/index.html`.

[82] R. Baraniuk, P. Flandrin, A. J. M. E. Jensen, and O. Michel. Measuring time frequency information content using the Rényi entropies. *IEEE Transactions on Information Theory*, 47(4), 2001.

[83] Fred-eZoneWiFi ISP. Available at `http://www.fred-ezone.ca`.

[84] H. Wang, D. Zhang, and K. G. Shin. Detecting SYN flooding attacks. In *Proceedings of INFOCOM*, 2002.

[85] C. Callegari, S. Giordano, M. Pagano, and T. Pepe. Combining sketches and wavelet analysis for multi time-scale network anomaly detection. *Computers and Security Journal*, 30(8), 2011.

[86] General purpose hash function algorithms. Available at `http://www.partow.net/programming/hashfunctions`.

[87] R. G. Stockwell, L. Mansinha, and R. P. Lowe. Localization of the complex spectrum: the S transform. *IEEE Transactions on Signal Processing*, 44(4):998–1001, 1996.

[88] P. K. Dash, B. K. Panigrahi, and G. Panda. Power quality analysis using S-transform. *IEEE Transactions On Power Delivery*, 18(2), 2003.

[89] S. Mishra, C. N. Bhende, and B. K Panigrahi. Detection and classification of power quality disturbances using S-transform and probabilistic neural network. *IEEE Transactions on Power Delivery*, 23(1), 2008.

[90] K. R. Krishnanand and P. K. Dash. A new real-time fast discrete s-transform for cross-differential protection of shunt-compensated power systems. *IEEE Transactions on Power Delivery*, 28(1), 2013.

[91] G. Livanos, N. Ranganatha, and J. Jiang. Heart sound analysis using the S transform. In *Proceedings of CINC*, 2000.

[92] Tcpreplay. Available at `http://tcpreplay.synfin.net/trac/`.

[93] daemon9, route, and infinity. Project Neptune. *Phrack Magazine*, 7(48), 1996. Available at `http://phrack.org/issues/48/13.html`.

[94] portscan.c. Available at `www.hoobie.net/security/exploits/hacking/portscan.c`.

[95] Nmap. Available at `http://insecure.org/nmap/`.

[96] Jping. Available at `http://www.tenebril.com/src/info.php`.

[97] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: multilevel traffic classification in the dark. In *Proceedings of SIGCOMM*, 2005.

[98] S. Pukkawanna, P. Pongpaibool, and V. Visoottiviseth. LD^2: a system for lightweight detection of denial-of-service attacks. In *Proceedings of MILCOM*, 2008.

[99] P. Borgnat, G. Dewaele, K. Fukuda, P. Abry, and K. Cho. Seven years and one day: sketching the evolution of internet traffic. In *Proceedings of INFOCOM*, pages 711–719, 2009.

[100] Iperf. Available at `https://iperf.fr`.

[101] tcpdump. Available at `http://www.tcpdump.org`.

[102] C. Callegari, L. Gazzarrini, S. Giordano, M Pagano, and T. Pepe. When randomness improves the anomaly detection performance. In *Proceedings of ISABEL*, 2010.

[103] C. Callegari, A. Casella, S. Giordano, M. Pagano, and T. Pepe. Sketch-based multidimensional IDS: a new approach for network anomaly detection. In *Proceedings of CNS*, 2013.

[104] R. Schweller, A. Gupta, E. Parsons, and Y. Chen. Reversible sketches for efficient and accurate change detection over network data streams. In *Proceedings of IMC*, 2004.

[105] A. Rényi. On measures of entropy and information. In *Proceedings of BSMSP*, 1961.

[106] J. Tang, Y. Cheng, and C. Zhou. Sketch-based SIP flooding detection using Hellinger distance. In *Proceedings of GLOBECOM*, 2009.

[107] J. Tajer, Ali. Makke, O. Salem, and A. Mehaoua. A comparison between divergence measures for network anomaly detection. In *Proceedings of CNSM*, 2007.

[108] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1), 1951.

[109] Rand()-generate pseudo-random bytes. Available at `https://www.openssl.org/docs/apps/rand.html`.

[110] T. van Erven and P. Harremoës. Rényi divergence and Kullback-Leibler divergence. *IEEE Transactions on Information Theory*, 6(1), 2007.

[111] J. Mazel, R. Fontugne, and K. Fukuda. A taxonomy of anomalies in backbone network traffic. In *Proceedings of IWCMC*, 2014.

[112] Song L, M. Yamada, N. Collier, and M. Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43, 2013.

[113] Bayesian approach to change points detection in time series. *International Journal of Imaging Systems and Technology*, 16(5), 2006.

[114] R. P. Adams and D. J. C. MacKay. Bayesian online changepoint detection. Technical report, United Kingdom, 2007.

[115] G. Karkavitsas and M. Rangoussi. Object localization in medical images using genetic algorithms. *International Journal of Medical, Health, Biomedical and Pharmaceutical Engineering*, 1(2), 2007.

[116] N. Otsu. A threshold selection method from gray level histograms. *IEEE Transaction on Systems, Man and Cybernetics*, 9, 1979.

[117] S. Pukkawanna, H. Hazeyama, Y. Kadobayashi, and S. Yamaguchi. Investing the utility of S-transform for detecting denial-of-service and probe attacks. In *Proceedings of ICOIN*, 2014.

[118] G. Pass and R. Zabith. Histogram refinement for content-based image retrieval. In *Proceedings of WACV*, 1996.

[119] D. Zhang and G. Lu. Shape-based image retrieval using generic Fourier descriptor. *Signal Processing: Image Communication*, 17(10), 2002.

[120] D. Freedman and P. Diaconis. On the histogram as a density estimator: L2 theory. *Probability Theory and Related Fields*, 57(4), 1981.

[121] H. Shimazaki and S. Shinomoto. A method for selecting the bin size of a time histogram. *Neural Computation*, 19(6), 2007.

[122] K. He and G. Meeden. Selecting the number of bins in a histogram: A decision theoretic approach. *Journal of Statistical Planning and Inference*, 61(1), 1997.

Publication List

Peer-reviewed Journals

1. Sirikarn Pukkawanna, Hiroaki Hazeyama, Youki Kadobayashi, and Suguru Yamaguchi. Detecting Anomalies in Massive Traffic Stream based on S-transform Analysis of Summarized Traffic Entropies. *IEICE Transactions on Information and Systems*, vol. E98-D, no.3, pp. 588-595, March 2015. (related to Section 3.5 in Chapter 3)

2. Sirikarn Pukkawanna, Youki Kadobayashi, and Suguru Yamaguchi. A New Statistical Feature and Tunable Method for Unsupervised Traffic Anomaly Detection. *IEICE Transactions on Information and Systems* (under review). (related to Chapter 4)

Peer-reviewed International Conference Papers Related to the Dissertation

1. Sirikarn Pukkawanna, Hiroaki Hazeyama, Youki Kadobayashi, and Suguru Yamaguchi. Building Better Unsupervised Anomaly Detector with S-Transform. In *Proceedings of the 7th International Conference on Network and System Security (NSS2013)*, *Lecture Notes in Computer Science*, vol. 7873, pp. 582-589, Madrid, Spain, June 2013. (related to Sections 3.3 and 3.4 in Chapter 3)

2. Sirikarn Pukkawanna, Hiroaki Hazeyama, Youki Kadobayashi, and Suguru Yamaguchi. Investing the Utility of S-Transform for Detecting Denial-of-Service and Probe Attacks. In *Proceedings of the 28th International Confer-*

ence on Information Networking (ICOIN2014), Phuket, Thailand, February 2014. (related to Section 3.2.2 in Chapter 3)

3. Sirikarn Pukkawanna, Hiroaki Hazeyama, Youki Kadobayashi, and Suguru Yamaguchi. Detecting Anomalies in Massive Traffic with Sketches. In *Proceedings of the 9th International Conference on Future Internet Technologies (CFI2014)*, Tokyo, Japan, June 2014. (related to Section 3.5 in Chapter 3)

4. Sirikarn Pukkawanna, Youki Kadobayashi, and Suguru Yamaguchi. Network-based Mimicry Anomaly Detection using Divergence Measures. In *Proceedings of the 2015 International Symposium on Networks, Computers and Communications (ISNCC2015)*, Sousse, Tunisia, May 2015. (related to Section 4.4.1 of Chapter 4)

Peer-reviewed International Conference Papers Not Related to the Dissertation

1. Yonchanok Khaokaew and Sirikarn Pukkawanna. Time series Anomaly Detection using Recessive Subsequence. In *Proceedings of the International Conference on Information Networking (ICOIN2012)*, Bali, Indonesia, February 2012.

2. Sophon Mongkolluksamee, Chavee Issariyapat, Panita Pongpaibool, Koolachat Meesublak, Nontaluck Nulong, and Sirikarn Pukkawanna. A Management System for Software Package Distribution. In *Proceedings of Portland International Center for Management of Engineering and Technology (PICMET2012)*, Vancouver, Canada, July 2012.

3. Sirikarn Pukkawanna, Youki Kadobayashi, Gregory Blanc, Joaquin Garcia-Alfaro, and Herve Debar. Classification of SSL Servers based on their SSL Handshake for Automated Security Assessment. In *Proceedings of the 3rd International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS2014)*, Wroclaw, Poland, September 2014.