UNIVERSITÀ DI PISA
Scuola di Dottorato in Ingegneria "Leonardo da Vinci"

Corso di Dottorato di Ricerca in
Ingegneria dell'Informazione
SSD ING-INF/03

Tesi di Dottorato di Ricerca

Advanced Techniques for Detecting Anomalies in Backbone Networks

Autore:
Teresa Pepe _____

Relatori:
Prof. Stefano Giordano _____

Prof. Michele Pagano _____

Prof. Franco Russo _____

Anno 2012

Copyright © Teresa Pepe, 2012
All rights reserved

Manuscript received February 28, 2012

Accepted March 28, 2012

Sommario

Con il rapido sviluppo e la crescente complessità delle reti di computer, i meccanismi tradizionali di network security non riescono a fornire soluzioni dinamiche e integrate adatte a garantire la completa sicurezza di un sistema. In questo contesto, l'uso di sistemi per la rilevazione delle intrusioni (Intrusion Detection System - IDS) è diventato un elemento chiave nell'ambito della sicurezza delle reti.

In questo lavoro di tesi affrontiamo tale problematica, proponendo soluzioni innovative per l'intrusion detection, basate sull'uso di tecniche statistiche (Wavelet Aanalysis, Principal Component Analysis, etc.) la cui applicazione per la rilevazione delle anomalie nel traffico di rete, risulta del tutto originale.

L'analisi dei risultati presentata, in questo lavoro di tesi, evidenzia l'efficacia dei metodi proposti.

Abstract

With the rapid development and the increasing complexity of computer and communication systems and networks, traditional security technologies and measures can not meet the demand for integrated and dynamic security solutions. In this scenario, the use of Intrusion Detection Systems has emerged as a key element in network security.

In this thesis we address the problem considering some novel statistical techniques (e.g., Wavelet Analysis, Principal Component Analysis, etc.) for detecting anomalies in network traffic.

The performance analysis, presented in this work, shows the effectiveness of the proposed methods.

Contents

List of Figures

List of Tables

List of Acronyms

CUSUM	Cumulative Sum
CWT	Continuous Wavelet Transform
DDoS	Distributed Denial of Service
DoS	Denial of Service
DWT	Discrete Wavelet Transform
EWMA	Exponential Weighted Moving Average
GF	Galois Field
HC	Heavy Change
HH	Heavy Hitter
HIDS	Host-based IDS
IDES	Intrusion Detection Expert System
IDS	Intrusion Detection System
IP	Internet Protocol
ITU-T	International Telecommunication Union Telecommunication standardization sector
JSD	Jensen-Shannon Divergence
KL	Kullback-Leibler divergenge
MNP-CUSUM	Multi-Chart Non-Parametric CUSUM

NSHW	Non Seasonal Holt-Winters
NIDS	Network-based IDS
OS	Operating System
OSI	Open Systems Interoperability
PC	Principal Component
PCA	Principal Component Analysis
RFC	Request for Comment

Introduction

With the rapid development and the increasing complexity of computer and communication systems and networks, traditional security technologies and measures can not meet the demand for integrated and dynamic security solutions. Moreover, along with the proliferation of new services, the threats from spammers, attackers and criminal enterprises also grow.

Recent advances in encryption, public key exchange, digital signature, and the development of related standards have set a foundation for network security. However, security on a network goes beyond these issues. Indeed it must include security of computer systems and networks, at all levels, top to bottom.

Since it seems impossible to guarantee complete protection to a system by means of prevention mechanisms (e.g., authentication techniques and data encryption), the use of an Intrusion Detection System (IDS) is of primary importance to reveal intrusions in a network or in a system.

State of the art in the field of intrusion detection is mostly represented by misuse based IDSs that are designed to detect known attacks by utilizing the signatures of those attacks. Considering that most attacks are realized with known tools, available on the Internet, a signature based IDS could seem a good solution. Nevertheless such

systems require frequent rule-base updates and signature updates, and are not capable of detecting unknown attacks.

In contrast, anomaly detection systems, a subset of IDSs, model the normal system/network behavior, which enables them to be extremely effective in finding and foiling both known as well as unknown or zero day attacks. This is the main reason why our work focuses on the development of an anomaly based IDS.

In particular we propose several novel statistical methods to be used for the description of the "normal" behavior of the network traffic. In more detail we explore the use of Wavelet Analysis, Principal Component Analysis, as well as techniques for Change-Point Detection and Heavy Hitter Detection.

All the proposed Anomaly based IDSs work on the top of a probabilistic structure, namely the sketch, that allows a random aggregation of the data, so as to obtain a more scalable system.

The remainder of this work is organized as follows: Chapter 1 is devoted to the description of the fundamentals of network security, mainly focusing on the description of the statistical Intrusion Detection Systems.

Chapter 2 presents some theoretical background on the sketch data model, used in this work as a common substrate for all the proposed systems.

The subsequent four chapters, namely Chapter 3, 4, 5, and 6, are dedicated to the detailed discussion of the implemented methods. In more detail Chapter 3, at first provides a quick overview of the theoretical background about Principal Component Analysis and then discusses the system architecture and the experimental results. Anal-

ogously Chapter 4 is devoted to the presentation of the architecture of the system based on the Wavelet Analysis, as well as of the achieved performance. Chapter 5 provides an overview of the Change Detection algorithms used by the implemented system (CUSUM, Heavy Hitter and Heavy Change detection algorithms) and then details the architecture and the performance of the system. Then Chapter 6 discusses both the architecture and the performance of a system based on a combined use of Wavelet Analysis and CUSUM.

Finally, Chapter 7 concludes the work with some final remarks.

Chapter 1

Intrusion Detection System

1.1 Network security

In the field of networking the area of network security [1] [2] usually refers to a complex process, which involves mechanisms and services necessary to guarantee the security of the information in a distributed context (e.g., the Internet).

The main objective of this process is to define a security policy that the network administrator can adopt to prevent and monitor unauthorized access, misuse, modification, or denial of the computer network and network-accessible resources.

To perform such operations it is necessary to keep in mind three important concepts:

- *Absolute security can not be guaranteed*: any system can be compromised, at least, by means of a brute-force attack [1]

[1] A brute force attack is a method of defeating a security scheme by trying a large number of possibilities; for example, exhaustively working through all possible

- *Security asymptotically improves*: securing a system is not a cost-less operation. Thus, it is necessary to attentively perform a risk evaluation phase
- *Security and easiness are opposite concepts*: adding security mechanisms to a system implies adding complexity

When talking about network security, there mainly exist two standard documents: RFC 2828 titled "Internet Security Glossary" [3] and the recommendation X.800 by ITU-T [4]. In the following sections we describe the architecture presented in X.800, only referring to [3] in some special cases.

1.1.1 The OSI security architecture

ITU-T recommendation X.800 defines a systematic approach to the security problems. For our purposes, the OSI security architecture provides a useful, if abstract, overview of many of the security concepts. The OSI security architecture focuses on security attacks, mechanisms, and services. These can be defined briefly as follows:

- **Security attack**: any action that compromises the security of information owned by an organization
- **Security mechanism**: a process (or a device incorporating such a process) that is designed to detect, prevent, or recover from a security attack
- **Security service**: a processing or communication service that enhances the security of the data processing systems and the

keys in order to decrypt a message

information transfers of an organization. The services are intended to counter security attacks, and they make use of one or more security mechanisms to provide the service

In the following we present, in more detail, each of these concepts.

Security services

X.800 defines a security service as a service provided by a protocol layer of communicating open systems, which ensures adequate security of the systems or of data transfers. Perhaps a clearer definition is found in [3], which provides the following definition: "a processing or communication service that is provided by a system to give a specific kind of protection to system resources; security services implement security policies and are implemented by security mechanisms".

X.800 divides these services into five categories and fourteen specific services:

- **Authentication**: the authentication service is concerned with assuring that a communication is authentic. In the case of a single message, such as a warning or alarm signal, the function of the authentication service is to assure the recipient that the message is from the source that it claims to be from. In the case of an ongoing interaction, such as the connection of a terminal to a host, two aspects are involved. First, at the time of connection initiation, the service assures that the two entities are authentic, that is, that each is the entity that it claims to be. Second, the service must assure that the connection is not interfered with in such a way that a third party can masquerade as one of the two

legitimate parties for the purposes of unauthorized transmission or reception. Two specific authentication services are defined in X.800:

  - Peer entity authentication: used in association with a logical connection to provide confidence in the identity of the entities connected
  - Data-origin authentication: in a connectionless transfer, provides assurance that the source of received data is as claimed

- **Access control**: in the context of network security, access control is the ability to limit and control the access to host systems and applications via communications links. To achieve this, each entity trying to gain access must first be identified, or authenticated, so that access rights can be tailored to the individual
- **Data confidentiality**: confidentiality is the protection of transmitted data from passive attacks. With respect to the content of a data transmission, several levels of protection can be identified:
  - Connection confidentiality: the protection of all user data on a connection
  - Connectionless confidentiality: the protection of all user data in a single data block
  - Selective-field confidentiality: the confidentiality of selected fields within the user data on a connection or in a single data block

  - Traffic-flow confidentiality: the protection of the information that might be derived from observation of traffic flows

- **Data integrity**: as with confidentiality, integrity can apply to a stream of messages, a single message, or selected fields within a message. Again, the most useful and straightforward approach is total stream protection. As in the previous case, several types of data integrity are defined in X.800:

  - Connection integrity with recovery: provides for the integrity of all user data on a connection and detects any modification, insertion, deletion, or replay of any data within an entire data sequence, with recovery attempted
  - Connection integrity without recovery: as above, but provides only detection without recovery
  - Selective-field connection integrity: provides for the integrity of selected fields within the user data of a data block and takes the form of determination of whether the selected fields have been modified, inserted, deleted, or replayed
  - Connectionless integrity: provides for the integrity of a single connectionless data block and may take the form of detection of data modification. Additionally, a limited form of replay detection may be provided
  - Selective-field connectionless integrity: provides for the integrity of selected fields within a single connectionless data block; takes the form of determination of whether the selected fields have been modified

- **Nonrepudiation**: nonrepudiation prevents either sender or receiver from denying a transmitted message
  - Nonrepudiation, origin: proves that the message was sent by the specified party
  - Nonrepudiation, destination : proves that the message was received by the specified party
- **Availability**: availability is the property of a system or a system resource being accessible and usable upon demand by an authorized system entity, according to performance specifications for the system.

  It is worth noticing that X.800 treats availability as a property to be associated with various security services. However, it makes sense to call out specifically an availability service. An availability service is one that protects a system to ensure its availability. This service addresses the security concerns raised by denial-of-service attacks. It depends on proper management and control of system resources and thus depends on access control service and other security services.

Security mechanisms

In the following we describe the security mechanisms defined in X.800. As can be seen the mechanisms are divided into those that are implemented in a specific protocol layer and those that are not specific to any particular protocol layer or security service.

- **Specific security mechanisms**: may be incorporated into the

appropriate protocol layer in order to provide some of the OSI security services

  - Encipherment: the use of mathematical algorithms to transform data into a form that is not readily intelligible. The transformation and subsequent recovery of the data depend on an algorithm and zero or more encryption keys
  - Digital signature: data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery
  - Access control: a variety of mechanisms that enforce access rights to resources
  - Data integrity: a variety of mechanisms used to assure the integrity of a data unit or stream of data units
  - Authentication exchange: a mechanism intended to ensure the identity of an entity by means of information exchange
  - Traffic padding: the insertion of bits into gaps in a data stream to frustrate traffic analysis attempts
  - Routing control: enables selection of particular physically secure routes for certain data and allows routing changes, especially when a breach of security is suspected
  - Notarization: the use of a trusted third party to assure certain properties of a data exchange

- **Pervasive security mechanisms**: mechanisms that are not specific to any particular OSI security service or protocol layer

- Trusted functionality: that which is perceived to be correct with respect to some criteria (e.g., as established by a security policy)
- Security label: the marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource
- Event detection: detection of security-relevant events
- Security audit trail: data collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities
- Security recovery: deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions

Security Attacks

In the literature, the terms threat and attack are commonly used to mean more or less the same thing. But, RFC 2828, defines this two concept in a different way:

- **Threat**: a potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm. That is, a threat is a possible danger that might exploit a vulnerability
- **Attack**: an assault on system security that derives from an intelligent threat; that is, an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system

Regarding the attacks, X.800 defines several specific attacks, divided into passive and active:

- **Active attacks**: active attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into four categories:
  - Masquerade: one entity pretends to be a different entity
  - Replay: involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect
  - Modification of messages: simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect
  - Denial of Service (DoS): prevents or inhibits the normal use or management of communications facilities
- **Passive attacks**: passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted. Passive attacks are very difficult to detect because they do not involve any alteration of the data. Typically, the message traffic is sent and received in an apparently normal fashion and neither the sender nor receiver is aware that a third party has read the messages or observed the traffic pattern.Two types of passive attacks are:
  - Release of message content: we would like to prevent an opponent from learning the contents of a data transmission

– Traffic Analysis: traffic analysis, is a subtler type of attack. Suppose that we had a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message. The common technique for masking contents is encryption. If we had encryption protection in place, an opponent might still be able to observe the pattern of these messages. The opponent could determine the location and identity of communicating hosts and could observe the frequency and length of messages being exchanged. This information might be useful in guessing the nature of the communication that was taking place

1.2 Intrusion Detection Systems

Since, inevitably, the best intrusion prevention system will fail, we need a mechanism which should act when an intrusion occurs. For this reason the IDS has been the focus of much research in recent years.

Definition 1 *An IDS is a software or hardware tool aimed at detecting unauthorized access to a computer system or a network.*

In other word, an intrusion detection is the act of detecting actions that attempt to compromise the confidentiality, integrity or availability of a system/network.

The attention IDSs have had is motivated by several considerations, including the following:

- If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised. Even if the detection is not sufficiently timely to preempt the intruder, the sooner the intrusion is detected, the less the amount of damage and the more quickly recovery can be achieved
- An effective IDS can serve as a deterrent, so acting to prevent intrusions
- Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility

Intrusion detection is based on the assumption that the behavior of intruder significantly differs from that of a legitimate user.

The concept of IDS was first introduced by Anderson in the early 80s [5]. The idea was to perform a post-processing of the audit data produced by a machine, so as to reveal if any intrusion had been carried out. The main flaw of this kind of system is that the detection was performed off-line.

For this reason, 1987 is usually considered as birth date of the IDSs, when Denning in [6] introduced her Intrusion Detection Expert System (IDES).

The main characteristic of IDES is to be independent of any particular system, application environment, system vulnerability, or type of intrusion. IDES is, in fact, a framework for a general-purpose IDS. Moreover, one of the most relevant feature of IDES is that it introduces the idea of performing the detection of intrusions by means of

a statistical analysis of the system input data. Thus, roughly speaking, the working of the system is based on the characterization of the behavior of a *subject*, with respect to a given *object*, by means of a *statistical profile*. An intrusion is revealed if such a profile is not respected by the input data.

In more detail, the model has six main components:

- **Subjects**: subjects are the initiators of actions in the target system. A subject is typically a terminal user, but might also be a process acting on behalf of users or groups of users, or might be the system itself. Subjects can be grouped into classes by type (groups may overlap)

- **Objects**: objects are the receptors of actions and typically include such entities as files, programs, messages, records, terminals, printers, and user- or program-created structures. When subjects can be recipients of actions (e.g., electronic mail), then those subjects are also considered to be objects in the model. Objects can be grouped into classes by type

- **Audit records**: generated by the target system in response to actions performed or attempted by subjects on objects-user login, command execution, file access, and so on

- **Profiles**: structures that characterize the behavior of a given subject (or set of subjects) with respect to a given object (or set thereof), thereby serving as a signature or description of normal activity for its respective subject and object. Observed behavior is characterized in terms of a statistical metric and

model. A metric is a random variable x representing a quantitative measure accumulated over a period. The period may be a fixed interval of time, or the time between two audit-related events. Observations x_i of x obtained from the audit records are used together with a statistical model to determine whether a new observation is abnormal. The statistical model makes no assumptions about the underlying distribution of x; all knowledge about x is obtained from observations. In more detail the metric can be:

- Event Counter: x is the number of audit records satisfying some property occurring during a period of time
- Interval Timer: x is the length of time between two related events
- Resource Measure: x is the quantity of resources consumed by some action during a period as specified in the Resource-Usage field of the audit records

Instead, the statistical model can be:

- Operational model: this model is based on the operational assumption that abnormality can be decided by comparing a new observation of x against fixed limits. Although the previous sample points for x are not used, presumably the limits are determined from prior observations of the same type of variable
- Mean and standard deviation model: this model is based on the assumption that all we know about $x_1, x_2, \ldots, x_n$,

are mean and standard deviation. A new observation x_{n+1} is defined to be abnormal if it falls outside a confidence interval

- Multivariate model: this model is similar to the mean and standard deviation model except that it is based on correlations among two or more metrics
- Markov process model: this model, which applies only to event counters, regards each distinct type of event as a state variable, and uses a state transition matrix to characterize the transition frequencies between states. A new observation is defined to be abnormal if its probability as determined by the previous state and the transition matrix is too low
- Time series model: this model, which uses an interval timer together with an event counter or resource measure, takes into account the order and interarrival times of the observations $x_1, x_2, \ldots, x_n$, as well as their values. A new observation is abnormal if its probability of occurring at that time is too low

- **Anomaly records**: generated when abnormal behavior is detected

- **Activity rules**: actions taken when some condition is satisfied, which update profiles, detect abnormal behavior, relate anomalies to suspected intrusions, and produce reports

1.2.1 IDS taxonomy

IDSs are usually classified on the basis of several aspects, in the following we describe the most important categories of these systems [7] [8].

Host-based IDS vs. Network-based IDS

A first distinction is usually made between host-based IDS (HIDS) and network-based IDS (NIDS). The main difference between these two categories is given by the input data of the system: a HIDS mainly processes the operating system's logs, while a NIDS processes the network traffic.

As a consequence of that, the first class of IDSs reveals those attacks, towards a single host, that leave some traces in the host's audit files. It should be clear that the most relevant limitation of this approach is given by the fact that these systems strongly depend on the OS. This fact usually takes to a low level of interoperability among different systems.

On the contrary, the NIDSs, processing low level data, do not depend on the hosts architecture. Moreover, these system are able to detect attacks that do not affect the system log files and can protect an entire LAN rather than a single host.

Stateless IDS vs. Stateful IDS

The second distinction we present, between stateful and stateless IDSs, is based on the approach, used to process the input data. A stateless system processes each event of the input data independently of the previous and the following events. On the contrary a stateful

IDS considers each event of the input data, as part of a stream of events. Thus the IDS decisions do not only depend on the observed event, but also on the position of the event in the stream.

It is worth noticing that the stateful technique represents a much more effective approach, since most intrusions are based, not on a single act, but on a sequence of operations, which has to be considered in the whole.

Misuse-based IDS vs. Anomaly-based IDS

A last distinction, probably the most important, is done on the basis of the detection technique. In this case we can distinguish between misuse-based IDS (also called signature-based IDS) and anomaly based IDS.

These two categories are based on a completely different approach to the intrusion detection problem. Indeed a misuse-based IDS reveals the intrusions, looking for patterns of action that are known to be related to an intrusion. These systems have a database, where the signatures of all the know attacks are stored. Thus, for each observed event, they run a pattern matching algorithm[2], to check if any of those signatures is present in the input data. To be noted that the use of a pattern matching algorithm usually implies an heavy computational effort, which often makes these systems too slow for on-line detection.

On the opposite, an anomaly-based IDS is based on the knowledge of a model, representing the normal behavior of the controlled system,

[2]A pattern matching algorithm, sometimes called string searching algorithm, is an algorithm that looks for a place where one or several patterns (also called strings) are found within a larger string or text

and an intrusion is considered as a *significant* deviation from that model. Such a model is usually learned by the IDS, during a training phase, performed on attack free input data. It is worth noticing that this kind of systems can also detect never seen before intrusions, while a misuse-based IDS, not having the signature in the database, can not detect any new attack. This ability is countervailed by a greater number of false alarms, which usually characterizes the anomaly-based systems.

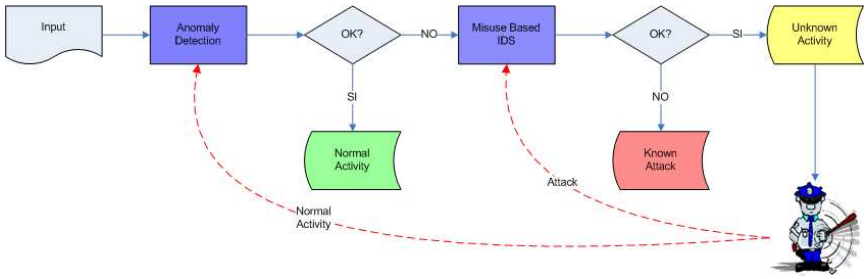


Figure 1.1: *IDS - hybrid architecture*

To conclude such an overview of the different categories, we have to highlight that to completely protect a computer network we have to simultaneously use a good combination of all the described systems. As an example, figure 1.1 shows how the anomaly-based approach and the misuse based approach can be combined to improve the overall performance. Indeed, the first check on data is performed by the anomaly-based IDS, which only forwards the suspicious data to the misuse-based system; in this way the second block only checks a small quantity of data, without excessively slowing down the processing. In the meanwhile, the misuse-based block, re-processing the data con-

sidered as anomalous by the first block, reduces the number of false alarms, which would be generated by the anomaly-based system.

1.3 Abilene/Internet2 Network

The proposed systems have been tested using a publicly available dataset, composed of traffic traces collected in the Abilene/Internet2 Network [9].

The Internet2 Network is a hybrid optical and packet network used by the U.S. research and education community. The backbone network consists of nine distinct routers distributed in nine different states in the U.S., as you can see in Figure 1.2.

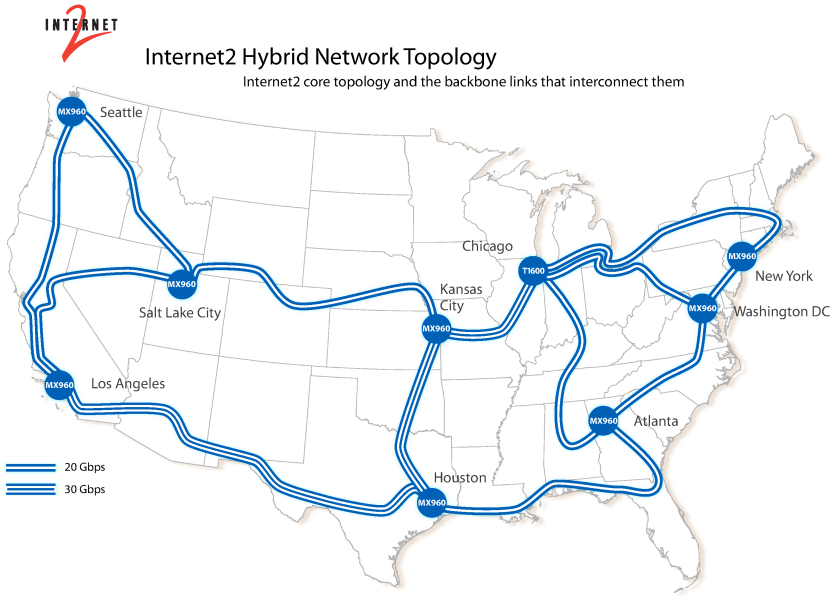


Figure 1.2: *Abilene/Internet2 Network*

The used traces consist of the traffic related to these routers, collected in one week, and organized into 2016 files, each one containing data about five minutes of traffic (Netflow data). To be noted that the last 11 bits of the IP addresses are anonymized for privacy reasons; nevertheless we have more than 220000 distinct IP addresses.

Since the data provided by the Internet2 project do not have a ground truth file, we are not capable of saying *a priori* if any anomaly is present in the data. Because of this reason we have performed a manual verification of the data (according to the method presented in [10]), analyzing the traces for which our system reveals the biggest anomalies. Moreover we have synthetically added some anomalies in the data, so as to be able to correctly interpret the offered results. In more detail, we have added anomalies that can be associated to DoS and DDoS attacks, represented by four or five distinct traffic flows, each one carrying a traffic of $5 \cdot 10^8$ bytes (154 anomalies in total), that span over a single or multiple time-bins.

Since the proposed systems work with ASCII data files, in all these systems, the input data are processed by a module called, Data Formatting. This module is responsible of reading the Netflow [11] traces and of transforming them in ASCII data files, by means of the Flow-Tools [12]. The output of this module is given by text files containing on each line an IP address and the number of bytes received by that IP in the last time-bin.

Note that from the Netflow traces we can extract several other traffic descriptors. Thus, instead of considering the number of bytes received by a given IP, the system administrator can easily choose of using another feature, if that better allows her to detect the different attacks.

Chapter 2

Sketch data model

2.1 Sketch

2.1.1 Data streaming model

In the last years, several data models have been proposed in the literature. In this section, we describe the streaming data, by using the most general model: the Turnstile Model [13].

According to this model, the input data are viewed as a stream that arrives sequentially, item by item. Let $I = \sigma_1, \sigma_2, \ldots, \sigma_n$ be the input stream.

Each item $\sigma_t = (i_t, c_t)$ consists of a *key*, $i_t \in (1, \ldots, N)$, and a *weight*, c_t. The arrival of a new data item causes the update of an underlying function $U_i[t]$ += c_t, which represents the sum of the *weights* of a given *key* until time t.

Given the underlying function $U_i[t]$ for all the keys of the stream, we can define the total sum $S(t)$, at step t, as follows:

$$S(t) = \sum_i U_i[t] \tag{2.1}$$

This model is very general and can be used in quite different scenar-

ios. As an example, in the context of network anomaly detection, the key can be defined using one or more fields of the packet header (IP addresses, L4 ports), or entities (like network prefixes or AS number) to achieve higher level of aggregation, while the underlying function can be the total number of bytes or packets in a flow.

2.1.2 Sketch

Sketches are powerful data structures that can be efficiently used for keeping an accurate estimate of the function U.

In general, sketches are a family of data structures that use the same underlying hashing scheme for summarizing data. They differ in how they update hash buckets and use hashed data to derive estimates. Among the different sketches, the one with the best time and space bounds is the so called count-min sketch [14].

In more detail, the sketch data structure is a two-dimensional $D \times W$ array $T[l][j]$, where each row l ($l = 1, \ldots, D$) is associated to a given hash function h_l. These functions give an output in the interval $(1, \ldots, W)$ and these outputs are associated to the columns of the array. As an example, the element $T[l][j]$ is associated to the output value j of the hash function l.

Let $I = \{(i_t, c_t)\}$ be an input stream observed during a given time interval. When a new item arrives, the sketch is updated as follows:

$$T[l][h_l(i_t)] \leftarrow T[l][h_l(i_t)] + c_t \tag{2.2}$$

The update procedure is realized for all the different hash functions as shown in Figure 2.1. In this way, at a given time-bin t, the bucket $T[l][h_l(i_t)]$ will contain an estimate of the quantity $U_i[t]$.

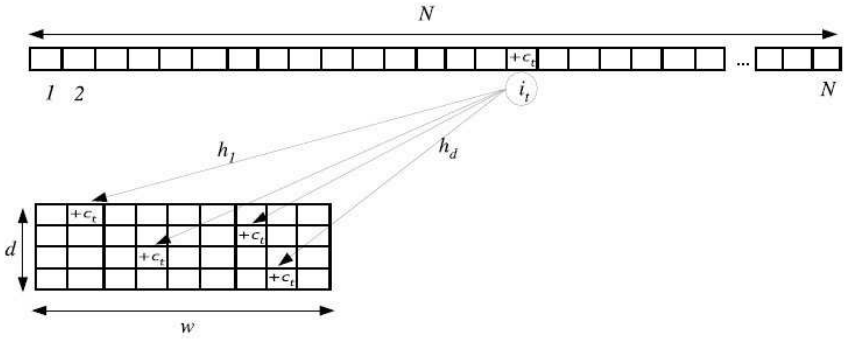


Figure 2.1: *Sketch - Update Function*

In this work, the sketches have been taken into consideration for two distinct reasons, which will be clearer in the following: on one hand they allow the storing of big quantities of data (in our case we have to store the traffic generated by more than 220000 IP addresses) with big memory savings, on the other hand they permit a random aggregation of the traffic flows. Indeed , given the use of hash functions, it is possible to have some *collisions* in the sketch table. In more detail, this last fact implies that each traffic flow will be part of several random aggregates, each of which will be analyzed to check if it presents any anomaly. This means that, in practice, any flow will be checked more than once (within different aggregates), thus, it will be easier to detect an anomaly. Indeed an anomaly could be masked in a given traffic aggregate, while being detectable in another one.

2.1.3 Reversible Sketch

Sketch data structures have a major drawback: they are not reversible. That is, a sketch cannot efficiently report the set of all keys (in our case, the flows) that correspond to a given bucket.

To overcome such a limitation, [15] proposes a novel algorithm for efficiently reversing sketches.

The basic idea is to perform an "intelligent" hash by modifying the input keys and/or the hashing functions so as to make possible to recover the keys with certain properties like big changes without sacrificing the detection accuracy.

In more detail the update procedure for the k-ary sketch is modified by introducing modular hashing and IP mangling techniques.

The *modular hashing* works partitioning the n-bit long hash key x into q words of equal length n/q, that are hashed separately using a different hash function, h_{di} $(i = (1, \ldots, q))$ (let us consider that the output of each function is m-bit long). Finally, these outputs are concatenated to form the final hash value (as depicted in Figure 2.2).

$$\delta_d(x) = h_{d1}(x)|h_{d2}(x)|\ldots|h_{dq}(x) \tag{2.3}$$

Since the final hash value consists of $q \times m$ bits, it can assume $w = 2^{q \times m}$ different values.

Note that the use of the modular hashing can cause a highly skewed distribution of the hash outputs. Consider, as an example, our case in which IP addresses are used as hash keys. In network traffic streams there are strong spatial localities in the IP addresses since many IP addresses share the same prefix. This means that the first octets

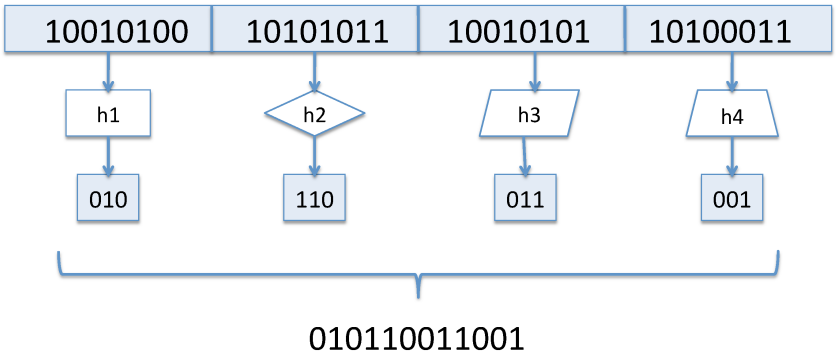


Figure 2.2: *Modular Hashing*

(equal in most addresses) will be mapped into the same hash values increasing the collision probability of such addresses.

To effectively resolve this problem, the *IP mangling* technique has to be applied before computing the hash functions. By using such a technique the system randomizes, in a reversible way, the input data so as to remove the correlation or spatial locality.

Essentially, this technique transforms the input set to a mangled set and performs all the operations on this set. The output is then transformed back to the original input keys. The function used for such a transformation is a bijective (one-to-one) function from key space $[2^n]$ to $[2^n]$.

A typical function used for this purpose is a function of the form

$$f(x) \equiv a \cdot x \mod 2^n \tag{2.4}$$

with a and 2^n relatively prime to guarantee the invertibility of the function.

The mangled key can easily be reversed by computing a^{-1} and applying the same function to the mangled key, using a^{-1} instead of a.

This function has the advantage of being extremely fast to compute but performs well only for non-adversarial key spaces in which no correlation exists among keys that have different (non empty) prefixes. Indeed, for any two keys that share the last bits, the mangled versions will also share the same last bits. Thus distinct keys that have common suffixes will be more likely to collide than keys with distinct suffixes.

However, in the particular case of IP addresses, this is not a problem. Due to the hierarchical nature of IP addresses, it is perfectly reasonable to assume that there is no correlation between the traffic of two IP addresses if they differ in their most significant bits.

However, this is not a safe assumption in general. For example, in the case of keys consisting of source and destination IP address pairs, the hierarchical assumption should not apply, and we expect to see traffic correlation among keys sharing the same destination IP but completely different IP source. And even for single IP address keys, it is plausible that an attacker could antagonistically cause a non-heavy-change IP address to be reported as a false positive by creating large traffic changes for an IP address that has a similar suffix to the target - also known as behavior aliasing.

Thus, to prevent adversarial attacks against the hashing scheme, a more sophisticated mangling function is needed.

In [15] the authors propose an attack-resilient scheme based on simple arithmetic operations on a Galois Extension Field $\mathbb{GF}(2^l)$, where

$l = \log_2 2^n$.

The function is now defined as follows:

$$f(x) \equiv a \otimes x \oplus b \tag{2.5}$$

where " $\otimes$ " is the multiplication operation defined on $\mathbb{GF}(2^l)$ and " $\oplus$ " is the bit-wise XOR operation. a and b are randomly chosen from $\{1, 2, \cdots, 2^l - 1\}$.

By precomputing a^{-1} on $\mathbb{GF}(2^l)$, we can easily reverse a mangled key y using

$$f^{-1}(y) = a^{-1} \otimes (x \oplus b). \tag{2.6}$$

The direct computation of $a \otimes x$ can be very expensive, as it would require multiplying two polynomials (of degree $l - 1$) modulo an irreducible polynomial (of degree l) on a Galois Field $\mathbb{GF}(2)$.

In practice this mangling scheme effectively resolves the highly skewed distribution caused by the modular hash functions as shown in Figure 2.3. Figure 2.3 shows the distribution of the number of keys (source IP address of each flow) per bucket for three different hashing scheme:

- modular hashing with no IP mangling
- modular hashing with the GF transformation for IP mangling
- direct hashing (a completely random hash function)

We observe that the key distribution of modular hashing with the GF transformation is essentially the same as that of direct hashing. The distribution for modular hashing without IP mangling is highly skewed.

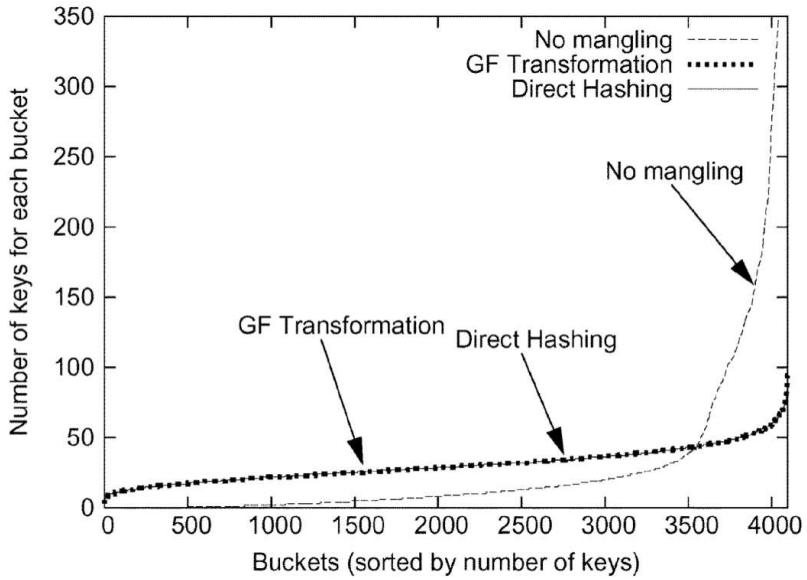


Figure 2.3: *IP Mangling - Distribution of number of keys for each bucket*

Thus IP mangling is very effective in randomizing the input keys and removing hierarchical correlations among the keys. In addition, the scheme is resilient to behavior aliasing attacks because attackers cannot create collisions in the reversible sketch buckets to make up false positive heavy changes. Any distinct pair of keys will be mapped completely randomly to two buckets for each hash table.

The other key point introduced in [15] is the algorithm for reversing the sketch, given the use of modular hashing and IP mangling.

Notation for the general algorithm

Let us introduce some notations useful for understanding the algorithm.

Let the dth row of the sketch table contain t_d heavy buckets. Let t be the value of the largest t_d. For each row of the sketch assign an arbitrary indexing of the t_d heavy buckets and let $t_{d,j}$ be the index in the row d of heavy bucket number j. Also define $\sigma_p(x)$ to be the pth word of a q word integer x. For example, if the jth heavy bucket in the row d is $t_{d,j} = 5.3.0.2$ for $q = 4$, then $\sigma_2(t_{d,j}) = 3$.

For each $d \in [D]$ and word p, denote the reverse mapping set of each modular hash function $h_{d,p}$ by the $2^m \times 2^{\frac{n}{q}-m}$ table $h^{-1}_{d,w}$ of $\frac{n}{q}$ bit words. That is, let $h^{-1}_{d,w}[j][k]$ denote the kth $2^{\frac{n}{q}}$ bit key in the reverse mapping of j for $h_{d,p}$. Further, let $h^{-1}_{d,p}[j] = \{x \in [2^{\frac{n}{q}}] | h_{d,p}(x) = j\}$.

Let $I_p = \{x | x \in \bigcup_{j=0}^{t_d-1} h^{-1}_{d,w}[\sigma_p(t_{d,j})]$ for at least $D-r$ values $d \in [D]\}$. That is, I_p is the set of all $x \in [2^{\frac{n}{q}}]$ such that x is in the reverse mapping for $h_{d,p}$ for some heavy bucket in at least $D-r$ of the D hash tables. We occasionally refer to this set as the *intersected modular potentials* for word p. For instance, in Figure 2.4, I_1 has three elements and I_2 two.

For each word we also define the mapping B_p which specifies for any $x \in I_p$ exactly which heavy buckets x occurs in for each hash table. In detail, $B_p(x) = \langle L_p[0][x], L_p[1][x], \ldots, L_p[D-1][x] \rangle$ where $L_p[i][x] = \{j \in [t] | x \in h^{-1}_{d,p}[\sigma_p(t_{i,j})]\} \bigcup *$. That is, $L_p[i][x]$ denotes the collection of indices in $[t]$ such that x is in the modular bucket potential set for the heavy bucket corresponding to the given index. The special character $*$ is included so that no intersection of sets L_p yields an empty set. For example, $B_p(129) = \langle \{1,3,8\}, \{5\}, \{2,4\}, \{9\}, \{3,2\} \rangle$

means that the reverse mapping of the 1st , 3rd , and 8th heavy bucket under $h_{0,p}$ all contain the modular key 129.

We can think of each vector $B_p(x)$ as a set of all D dimensional vectors such that the dth entry is an element of $L_p[d][x]$. For example, $B_3(23) = \langle\{1,3\},\{16\},\{*\},\{9\},\{2\}\rangle$ is indeed a set of two vectors: $\langle\{1\},\{16\},\{*\},\{9\},\{2\}\rangle$ and $\langle\{3\},\{16\},\{*\},\{9\},\{2\}$. We refer to $B_p(x)$ as the bucket index matrix for x, and a decomposed vector in a set $B_p(x)$ as a bucket index vector for x. We note that although the size of the bucket index vector set is exponential in D, the bucket index matrix representation is only polynomial in size and permits the operation of intersection to be performed in polynomial time. Such a set like $B_1(a)$ can be viewed as a node in Figure 2.4.

Define the r intersection of two such sets to be $B \bigcap^r C = \{v \in B \bigcap C | v$ has at most r of its D entries equal to $*\}$. For example, $B_p(x) \bigcap^r B_{p+1}(y)$ represents all of the different ways to choose a single heavy bucket from each of at least $D - r$ of the hash tables such that each chosen bucket contains x in its reverse mapping for the pth word and y for the $p+1$th word. For instance, in Figure 2.4, $B_1(a) \bigcap^r B_2(d) = \langle\{2\},\{1\},\{4\},\{*\},\{3\}\rangle$, which is denoted as a link in the figure. Note that there is not such a link between $B_1(a)$ and $B_2(e)$. Intuitively, the $a.d$ sequence can be part of a heavy change key because these keys share common heavy buckets for at least $D - r$ hash tables. In addition, it is clear that a key $x \in [2^n]$ is a suspect key for the sketch if and only if $\bigcap_{p=1\ldots q}^r B_p(x_p) \neq 0$.

Finally, we define the sets A_p which we compute in our algorithm to find the suspect keys. Let $A_1 = \{(\langle x1\rangle, v) | x_1 \in I_1$ and $v \in B_1(x_1)\}$. Recursively define $A_{p+1} = \{(\langle x_1, x_2, \ldots, x_{p+1}\rangle, v) | (\langle x_1, x_2, \ldots, x_w\rangle, v) \in$

A_w and $v \in B_{w+1}(x_{w+1})\}$. Take Figure 2.4 for example. Here A_4 contains $\langle a, d, f, i\rangle, \langle 2, 1, 4, *, 3\rangle$ which is the suspect key. Each element of A_p can be denoted as a path in Figure 2.4. The following lemma tells us that it is sufficient to compute A_q to solve the reverse sketch problem.

Lemma 1 *A key $x = x_1 \cdot x_2 \ldots x_q \in [2^n]$ is a suspect key if and only if $(\langle x_1, x_2, \ldots, x_q\rangle, v) \in A_q$ for some vector v.*

General algorithm for Reverse Hashing

To solve the reverse sketch problem we first compute the q sets I_p and bucket index matrices B_p. From these we iteratively create each A_p starting from some base A_c for any c where $1 \leq c \leq q$ up until we have A_q. We then output the set of heavy change keys via Lemma 1. Intuitively, we start with nodes as in Figure 2.4, I_1 is essentially A_1. The links between I_1 and I_2 give A_2, then the link pairs between $(I_1 I_2)$ and $(I_2 I_3)$ give A_3, etc.

The choice of the base case A_c affects the performance of the algorithm. The size of the set A_1 is likely to be exponentially large in D. However, with good random hashing, the size of A_p for $p \geq 2$ will be only polynomial in D, q, and t with high probability with the detailed algorithm and analysis below. Note we must choose a fairly small value c to start with because the complexity of computing the base case grows exponentially in c.

The pseudocode of the Reverse Hashing algorithm is reported in Algorithm 1. Instead, the pseudocode for the functions $MODULAR\_POTENTIALS$ and $EXTEND$ is reported in Algorithms 2 and 3, respectively.

Algorithm 1 REVERSE_HASH(r)

1: **for** $p = 1 : l$ **do**
2: $(I_p B_p) = MODULAR\_POTENTIALS(p, r)$
3: **end for**
4: $A_2 = 0$
5: **for** $x \in I_1, y \in I_2$ and corresponding $v \in B_1(x) \bigcap^r B_2(y)$ **do**
6: insert $(\langle x, y \rangle, v)$ into A_2
7: **end for**
8: **for** any given A_p **do**
9: $A_{p+1} = EXTEND(A_p, I_{p+1}, B_{p+1})$
10: **end for**
11: Output all $x_1.x_2.\cdots.x_q \in [n]$ s.t. $(\langle x_1, \ldots, x_q \rangle, v) \in A_q$ for some v

2.2 Implementation

Reversible Sketches are a common substrate for all the proposed systems, used to perform a random aggregation of the data.

In the implemented systems, the module responsible for the construction of the reversible sketch tables takes in input the data files described in Section 1.2.

As said before, each line of these files contains an IP destination address and the number of the bytes received by that IP in the last time bin (i.e. five minutes of traffic).

Each file is thus used to build a distinct sketch table. In more detail, according to the Tunstile model, presented in Section 2.1.1, the IP address IP_t is considered as the hash key i_t, while the number of bytes B_t is considered as the weight c_t.

Algorithm 2 MODULAR_POTENTIALS(p, r)

Create an $D \times 2^{\frac{n}{q}}$ table of sets L initialized to all contain the special character $*$
Create a size $[2^{\frac{n}{q}}]$ array of counters hits initialized to all zeros
for $i \in [D], j \in [t]$, and $k \in [2^{\frac{n}{q}-m}]$ **do**
    insert $h_{d,p}^{-1}[\sigma_p(t_{d,j})][k]$ into $L[d][x]$
    if $L[d][x] == 0$ **then**
        $hits[x]++$
    end if
end for
for $x \in [2^{\frac{n}{q}}]$ s.t. $hits[x] \geq D - r$ **do**
    insert x into I_p
    $B_p(x) = \langle L[0][x], L[1][x], \ldots L[D-1][x] \rangle$
end for
Output (I_p, B_p)

Algorithm 3 EXTEND(A_p, I_{p+1}, B_{p+1})

$A_{p+1} = 0$
for $y \in I_{p+1}, (\langle x_1, \ldots, x_p \rangle, v) \in A_p$ **do**
    if then$v \bigcap^r B_{p+1}(y) \neq null)$
        insert $(\langle x_1, \ldots, x_p \rangle, v \bigcap^r B_{p+1}(y))$ into A_{p+1}
    end if
end for
Output (A_{p+1})

Note that in our implementation we have used $d = 32$ distinct hash functions, which give output in the interval $[0; w-1]$, that means that

the resulting sketches will be $\in \mathbb{N}_{d\times w}$, where w can be varied.

As far as the hash functions are concerned, we have used 4-universal hashes[1] [16], obtained as:

$$h(x) = \sum_{i=0}^{3} a_i \cdot x^i \ mod\ p\ mod\ w \tag{2.7}$$

where the coefficients a_i are randomly chosen in the set $[0, p-1]$ and p is an arbitrary prime number (we have considered the Mersenne numbers).

At this point, given that we had N distinct time-bins, we have obtained N distinct sketch tables. Starting from these we consider the temporal evolution of each bucket T_{lj} of the sketch table, constructing $d \cdot w$ time series of N samples $T_{lj}[n]$ each.

The pseudocode about the sketch computation is given in Algorithm 4.

For sake of brevity we do not describe the function $IP\_MANGLING$ and $MODULAR\_HASHING$ already described in Section 2.1.3.

[1] A class of hash functions $H : (1, \ldots, N) \rightarrow (1, \ldots, w)$ is a *k-universal hash* if for any distinct $x_0, \ldots x_{k-1} \in (1, \ldots, N)$ and any possible $v_0, \ldots v_{k-1} \in (1, \ldots, w)$:

$$\Pr_{h\in H} = \{h(x_i) = v_i; \forall i \in (1, \ldots, k)\} = \tfrac{1}{w^k}$$

Algorithm 4 Building the sketch

Input: IP destination address - $ip_1.ip_2.ip_3.ip_4$
for $l = 1 : d$ **do**
    for $j = 1 : w$ **do**
        $T_n[l][j] = 0$ ▷ sketch table initialization
    end for
end for
for $n = 1 : N$ **do**
    for $t = 1 : S$ **do**
        $IP\_MANGLING(ip_t)$
        $\delta = MODULAR\_HASHING(ip_t)$
        $Store: \ ip_{1t}, \ ip_{2t}, \ ip_{3t}, \ ip_{4t}$
        for $l = 1 : d$ **do**
            $T_n[l][\delta] += B_t$
        end for
    end for
end for
Output: N distinct hash tables T_n.

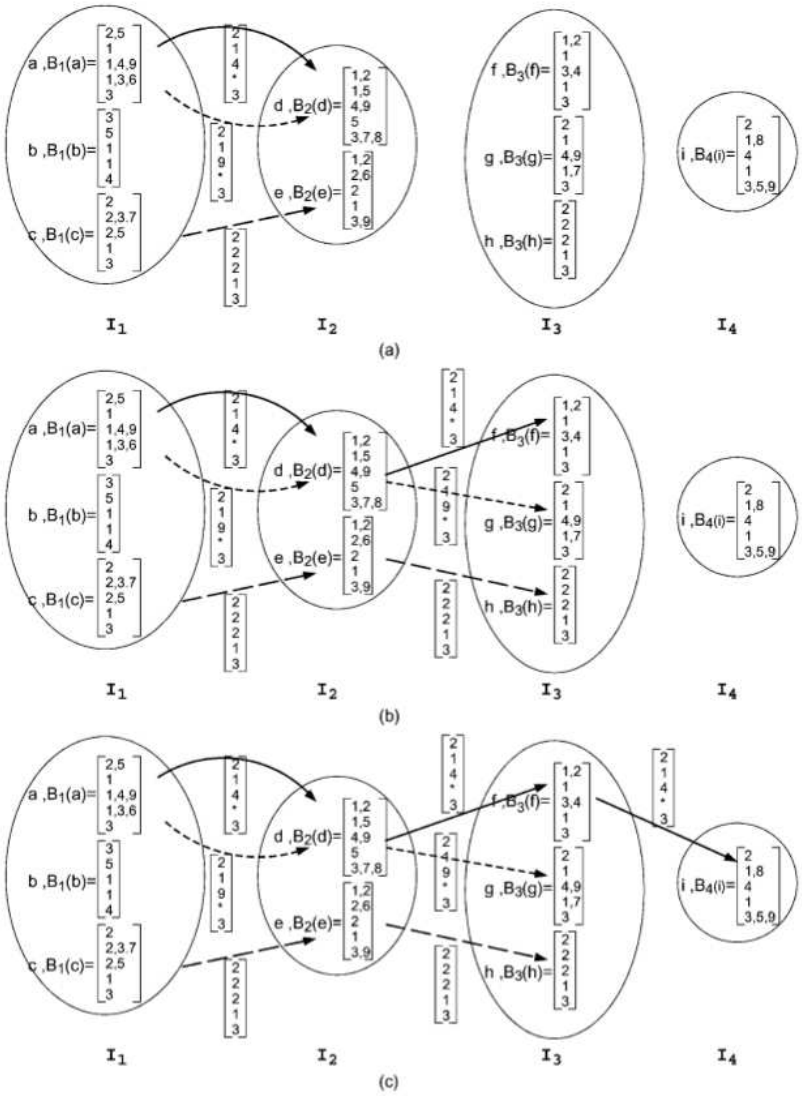


Figure 2.4: *Reverse hashing procedure (example)*

Chapter 3

Principal Component Analysis

3.1 Pricipal Component Analysis

The Principal Components Analysis (PCA) is a linear transformation that maps a coordinate space onto a new coordinate system whose axes, called Principal Components (PCs), have the property to point in the direction of maximum variance of the residual data (i.e., the difference between the original data and the data mapped onto the previous PCs).

In more detail, the first PC captures the greatest degree of data variance in a single direction, the second one captures the greatest degree of variance of data in the remaining orthogonal directions, and so on.

A simple illustration of the PCA is shown in Figure 3.1 where the PCs of a two dimensional dataset are plotted. As you can see in the figure, the number of PCs is equal to the dimensionality of the original dataset; the first PC is in the direction that maximizes the variance of the projected data (green line), the second PC, instead, is in the

orthogonal direction (blue line).

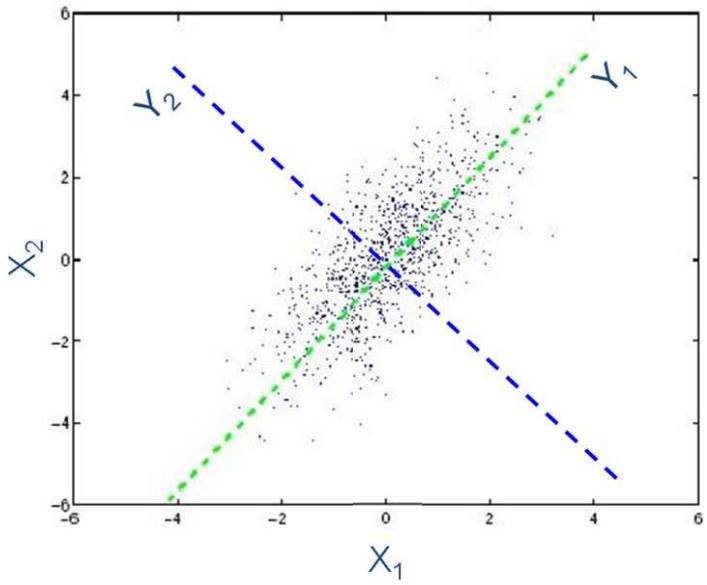


Figure 3.1: *PCs of a two-dimentional dataset*

In mathematical terms, to calculate the PCs is equivalent to compute the eigenvectors of the covariance matrix.

In more detail, given the matrix of data $B = \{B_{i,j}\}$, with $1 < i < m$ and $1 < j < t$ (e.g., a dataset of m samples captured in t time-bins), each PC, v_i, is the $i - th$ eigenvector computed from the spectral decomposition of the covariance matrix $C = BB^T$, that is:

$$BB^T v_i = \lambda_i v_i \qquad i = 1, \ldots, m \tag{3.1}$$

where λ_i is the "ordered" eigenvalue corresponding to the eigenvector v_i.

In practice, the first PC, v_1, is computed as follows:

$$v_1 = \arg\max_{\|v\|=1} \|Bv\| \tag{3.2}$$

where $\|Bv\|^2$ is proportional to the variance of the data measured along v.

Proceeding recursively, once the first $k-1$ PCs have been determined, the $k-th$ PC can be evaluated as follows:

$$v_k = \arg\max_{\|v\|=1} \left\| (B - \sum_{i=1}^{k-1} Bv_i v_i^T)v \right\| \tag{3.3}$$

where $||\cdot||$ denotes the L^2 norm.

Once the PCs have been computed, given a set of data and its associated coordinate space, we can perform a data transformation by projecting them onto the new axis.

Typically, the first PCs contribute most of the variance in the original dataset, so that we can describe them with only these PCs, neglecting the others, with minimal loss of variance.

For this reason, the PCA is also used as a linear dimension reduction technique. The main idea is to calculate the PCs and establish how many of them are sufficient to describe the original dataset. To select the PCs we can perform the scree plot method.

A scree-plot is a plot, like the one represented in Figure 3.2, of the percentage of variance captured by a given PC. Thus, studying the scree plot we can determine the optimal number of variables, in order to lower the dimensionality of a complex dataset, while maintaing the most of the variance of the original dataset.

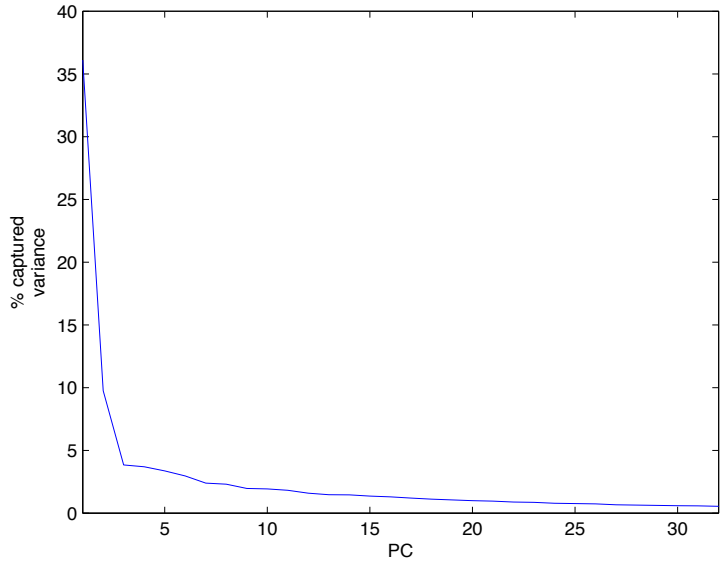


Figure 3.2: *Scree Plot*

3.2 System architecture

In this Section, we present the architecture of the system we have implemented to detect anomalies in the network traffic.

In Figure 4.3 is reported a block scheme of the proposed system.

We can distinguish four main block:

- Data Formatting
- Flow Aggregation
- Time-series Construction
- Anomaly Detector

- Identification

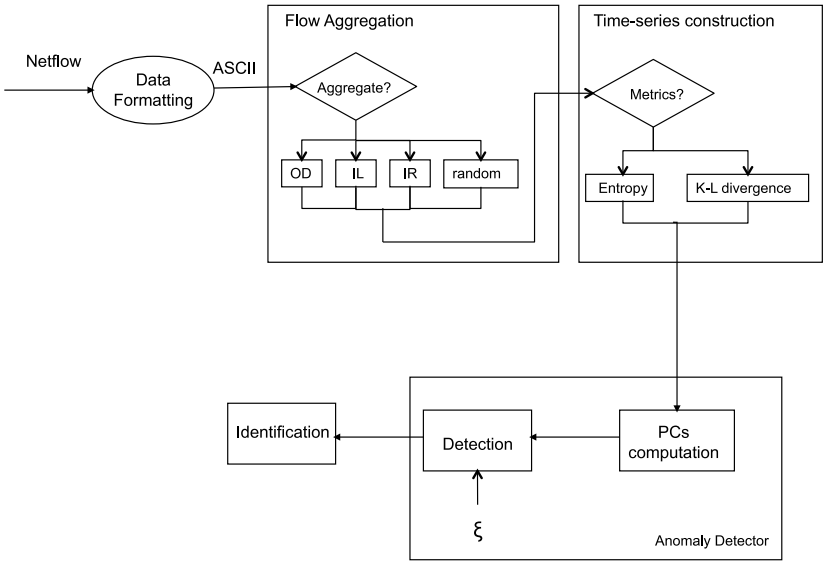


Figure 3.3: *PCA - System Architecture*

For sake of brevity we skip the detail related to the Data Formatting module since a detailed analysis of this block has been made in Section 1.2.

3.2.1 Flow Aggregation

To facilitate the detection of correlation and periodic trends in the data, we have studied different levels of aggregation. In more detail, the block called Flow Aggregation realizes four different levels of aggregation:

- Ingress Router (IR)
- Origin Destination (OD) flows
- Input Link (IL)
- Random Aggregation

Using the IR aggregation, data are organized according to which router they entered the network. The OD flow aggregates traffic based on the ingress and egress routers used by an IP flow. Instead, the IL aggregates IP flows with the same ingress router and input interface. Finally, the random aggregation is performed by means of the sketch (as described in Chapter 2).

The output of this block is given by $4T$ distinct files, each one corresponding to a specific time-bin and a specific aggregate.

3.2.2 Time-series Construction

After the data have been correctly formatted, they are passed as input to the third module, responsible for the construction of the time-series.

Typically the distribution of packet features (packet header fields) observed in flow traces reveals both the presence and the "structure" of a wide range of anomalies. Indeed, traffic anomalies induce changes in the normal distribution of the features.

Based on this observation, we have examined the distributions of traffic features as a means to detect and to classify network anomalies.

More specifically, in this work, we have taken into consideration the number of bytes sent by each IP source address.

The feature distribution has been estimated with the empirical histogram. Thus, in each time-bin for each aggregate we have evaluated the histogram as follows:

$$X^t = \{n_i^t, i = 1, \ldots, N\} \tag{3.4}$$

where n_i^t is the number of bytes transmitted by the $i-th$ IP address in the time-bin t.

Unfortunately, the histogram is an high-dimensional object quite difficult to handle with low computational resources.

For this reason we have tried to concentrate the information taken by the histogram in a single value, able to hold the most of the useful information, that, in our case, is the trend of the distribution.

Previous works [17] [18] [10] have emphasized the possibility to extract very useful information from the *entropy* of the distribution.

The entropy provides a computationally efficient metric for estimating the degree of dispersion or concentration of a distribution.

Given the empirical histogram, X^t, we can evaluate the entropy value as follows:

$$H^t = -\sum_{i=1}^{N} \frac{n_i^t}{S} \log_2 \frac{n_i^t}{S} \quad \text{with} \quad S = \sum_{i=1}^{N} n_i^t \tag{3.5}$$

Nevertheless, the entropy is only able to capture the information related to a single time-bin, while from our point of view it would be much more important to capture the the difference between packet feature distributions of two adjacent time-bins.

For this reason, in this work we have also used another metric that is the Kullback-Leibler (KL) divergence. Given two histograms X^t

(captured in time-bin t) and X^{t-1} (captured in time-bin $t-1$), the KL divergence is defined as follows:

$$D_{KL}^{t} = \sum_{i=1}^{N} n_i^{t-1} \log \frac{n_i^{t-1}}{n_i^{t}} \tag{3.6}$$

Despite of the method used, this module will output a matrix for each type of aggregation in which, for all the aggregates, the values of the metric (entropy or KL divergence) evaluated in each time-bin are reported.

This matrix has the following structure:

$$Y = \begin{pmatrix} y_{11} & \cdots & y_{1N} \\ \vdots & & \vdots \\ y_{T1} & \cdots & y_{TN} \end{pmatrix} \tag{3.7}$$

where N is the number of aggregates and T the number of time-bins.

3.2.3 Anomaly Detector

This block is the main component of the detection system. It consists of a Detection phase, during which the system detects anomalies by means of the PCA technique, and an Identification phase to identify the IPs responsible of the anomalies.

PCs computation

After the time-series have been constructed, they are passed to the module that applies the PCA. The computation of the PCs is performed using Equations (3.2) and (3.3). As described before, typically

there is a set of PCs (called *dominant* PCs) that contributes most of the variance in the original dataset.

The idea is to select the dominant PCs and describe the normal behavior only using these ones.

It is worth highlighting that the number of dominant PCs is a very important parameter, and needs to be properly tuned when using the PCA as a traffic anomaly detector.

In our approach the set of dominant PCs is selected by means of the scree-plot method. As a result, we separate the PCs into two sets, dominant and negligible PCs, that will be then used to distinguish between normal and anomalous variations in traffic.

Figure 3.2 reports a scree-plot of a particular dataset used in our study. Visually, from the graph we can observe that the first $r = 5$ PCs are able to correctly capture the majority of the variance. These PCs are the dominant PCs:

$$P = (v_1, \ldots, v_r) \tag{3.8}$$

The method is based on the assumption that these PCs are sufficient to describe the normal behavior of traffic.

Detection Phase

The detection phase is performed by separating the high-dimensional space of traffic measurements into two subspaces, which capture normal and anomalous variations, respectively.

In more detail, once the matrix P has been constructed, we can partition the space into a normal subspace ($\hat{S}$), spanned by the dominant PCs, and an anomalous subspace ($\tilde{S}$), spanned by the remaining PCs

The normal and anomalous components of data can be obtained by projecting the aggregate traffic onto these two subspaces.

Thus, the original data, in the time-bin t, Y_t are decomposed into two parts as follows:

$$Y_t = \hat{Y}_t + \tilde{Y}_t \tag{3.9}$$

where $\hat{Y}_t$ and $\tilde{Y}_t$ are the projection onto $\hat{S}$ and $\tilde{S}$, respectively, and can be evaluate as follows:

$$\hat{Y}_t = PP^T Y_t \tag{3.10}$$

$$\tilde{Y}_t = (I - PP^T)Y_t \tag{3.11}$$

To be noted that, when anomalous traffic crosses the network, a large change in the anomalous component ($\tilde{Y}_t$) occurs. Thus, an efficient method to detect traffic anomalies is to compare $\|\tilde{Y}_t\|^2$ with a given threshold (ξ).

In more detail, if $\|\tilde{Y}_t\|^2$ exceeds the given threshold ξ, the traffic is considered anomalous, and we mark the time-bin (t) as an anomalous time-bin (Figure 3.4).

If we use a random aggregation the detection phase can be improved performing a further step.

Indeed, in this case, since we have used d different hash functions h_j we have d data matrices, Y^j (one for each function). So, the previously described analysis (performed for each Y^j) returns d different responses.

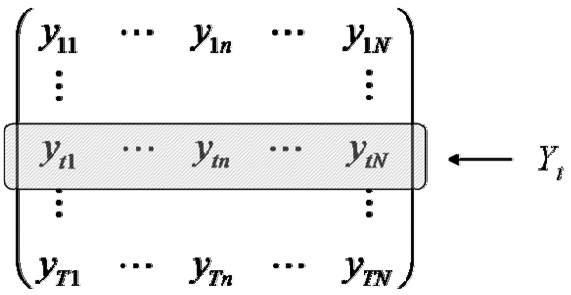


Figure 3.4: *PCA - Data matrix*

Thus, a voting analysis is performed. We evaluate the number of produced alarms and we decide if the time-bin is anomalous or not according to the following rule:

$$\begin{cases} \text{anomalous} & if \quad \text{number of alarms} > \dfrac{d}{2} - 1 \\ \text{normal} & \text{otherwise} \end{cases}$$

More details about the detection phase are given in Algorithm 5.

3.2.4 Identification

If an anomaly has been detected, the system performs a further phase called anomaly identification.

Note that the PCA works on a single time-series, so in the detection phase we are able to identify the time-bin during which traffic is anomalous. But, at this point, we do not know the specific network event that has caused the detection.

Algorithm 5 Detection

Input: Data matrices - Y^j, $j = 0, \ldots, d$
$alarm = 0$
for $j = 1 : d$ **do**
    Compute the PCs
    Select the dominant PCs : $P = (v_1, \ldots, v_r)$
    $\tilde{Y^j} = (I - PP^T)Y^j$ ▷ anomalous component of the traffic
    if $\|\tilde{Y_t}\|^2 > \xi$ **then**
        $alarm++$
    end if
end for
if $alarm > \frac{d}{2} - 1$ **then**
    the time-bin is anomalous
end if

In fact, it is worth noticing that an anomalous time-bin may contain multiple anomalous events, and that a single anomalous event can span over multiple time-bins.

In this phase we want to identify the specific flows responsible for the revealed anomaly.

Note that if the traffic aggregation is performed by means of IR, OD flow, and IL we are not able to determine the specific flow responsible for the revealed anomaly.

On the contrary, when we use random traffic aggregation, since we use the reversible sketches, we are able to correctly identify the specific flows involved in the anomalous traffic. In more detail, at first, we search the specific traffic aggregation in which the anomaly has occurred. Then reversing the sketch we can identify the specific flows that have caused the anomaly.

3.3 Experimental results

The proposed system has been tested using the dataset presented in Section 1.2.

Before detailing the performance achieved by the system in terms of number of detected anomalies and detection rate, let us analyze the sensitivity of our method to two key parameters, i.e., the dimension of the normal subspace (number of dominant PCs, r) and the sketch size w.

It is important to say that the presented performance have been obtained varying the value of the threshold ξ in a range chosen on the basis of the values of $||Y||^2$.

As previously said, for the selection of an appropriate number of PCs we can use the scree-plot method. The scree-plot reported in Figure 3.2 is related to random aggregation for a sketch size w of 64 and bin-size of 5 minutes. From the plot we can easily notice that most of the data energy is captured by the first five PCs and that after the eighth PC, the contribution of the remaining PCs is less than 4%. Thus we have decided to perform our analysis with a number of PCs $r \in [2, 7]$.

Figure 3.5 shows the detection rate (computed over the synthetically added anomalies) when varying the number of PCs (note that, given the nature of the dataset, we cannot plot a ROC curve).

It is worth noticing that a very low value of r takes to correctly detect a good number of anomalies, also raising a big number of false alarms, due to the fact that also the "normal" components are considered in the anomalous subspace. Vice-versa, considering a high value of r takes to a bad detection rate, behavior due to the fact that considering

a high number of PCs implies to insert in the normal subspace some anomalous components. Given this, it is evident that r is an important parameter and it has to be chosen so as to obtain a good trade-off between detection rate and false alarm rate.

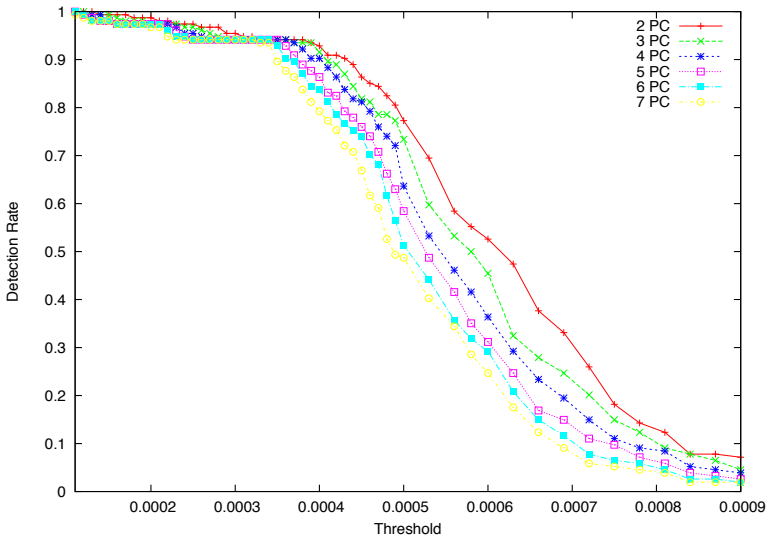


Figure 3.5: *PCA - Detection Rate vs. Number of PCs*

Concerning the sketch size, it is important to highlight that in our implementation we have used $d = 8$ distinct hash functions, which give output in the interval $[0, w - 1]$; so the resulting sketch tables will be $\in \mathbb{N}_{8\times w}$, where w is a parameter to set.

The choice of w is very important, since this parameter determines the number and the composition of the aggregates, significantly influencing the detection rate. For this reason, we have studied the

detection rate achieved by the system when varying the sketch size. Figure 3.6 shows the results of such an analysis, when r has been fixed equal to 5. From an analysis of the achieved results we have concluded that the best performance are achieved when $w = 64$. Indeed, even though the graph shows better performance for $w = 32$, in that case there are too few traffic aggregates taking to a huge number of false alarms.

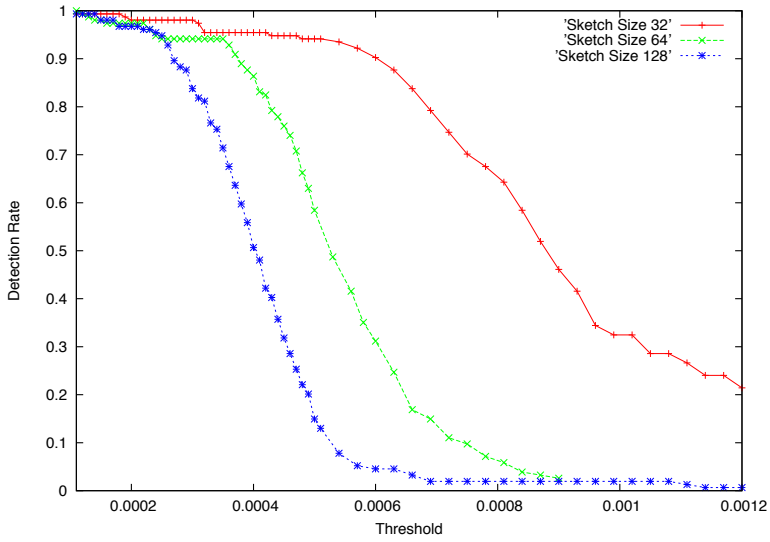


Figure 3.6: *PCA - Detection Rate vs. Sketch Size*

In the following subsections we show the performance achieved by our system, in terms of detection rate (computed over the synthetically added anomalies) and total number of detected anomalies. It is

important to highlight that the system stops analyzing a given time-bin, once an anomaly has been detected, this implies that the number of detected anomalies is equal to the number of anomalous time-bins. The presence of multiple anomalies in a time-bin will be eventually detected during the identification phase.

Such a results discussion is divided into two distinct parts:

- performance achieved with the different types of traffic aggregations
- performance achieved by using KL divergence

3.3.1 Traffic aggregations

The aim of this first performance evaluation is to establish if considering different traffic aggregation criteria can somehow modify the detection rate achieved by the system, and in particular to evaluate the effectiveness of the random aggregation provided by the use of sketch. The graphs presented in this section have been obtained with a sketch size $w = 64$ and the use of entropy.

Figures 3.7 and 3.8 respectively show the detection rate and the total number of detected anomalies when the OD aggregation is used. The different curves have been obtained varying the number of PCs. As expected the best detection rate is obtained when considering a very low number of PCs, but this implies to have a high number of false positives. By manual inspecting the dataset, it appears that the best trade-off is achieved when using 5 PCs.

From the graphs we can see that the performance do not strongly depend on the number of PCs and that the detection rate and the

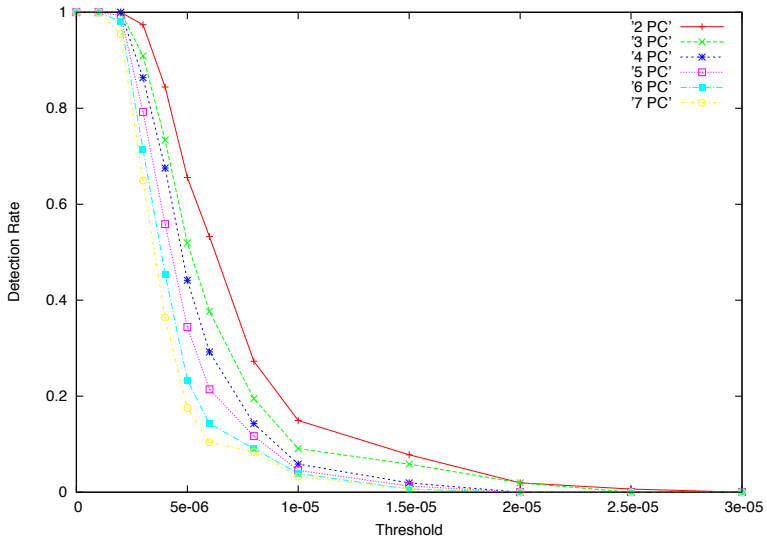


Figure 3.7: *PCA - Detection Rate for OD traffic aggregation*

total number of detected anomalies decrease following the same trend, when varying the threshold.

Regarding the IR aggregation the achieved performance are shown in Figures 3.9 and 3.10. Differently from the previous graphs we can notice that, in this case, the performance strongly depend on the number of PCs, and that we have a strong worsening when we consider more than 3 PCs.

It is also important to notice that, when using IR aggregation, the total number of detected anomalous events is much lower than in the previous case. This last consideration can take us to conclude

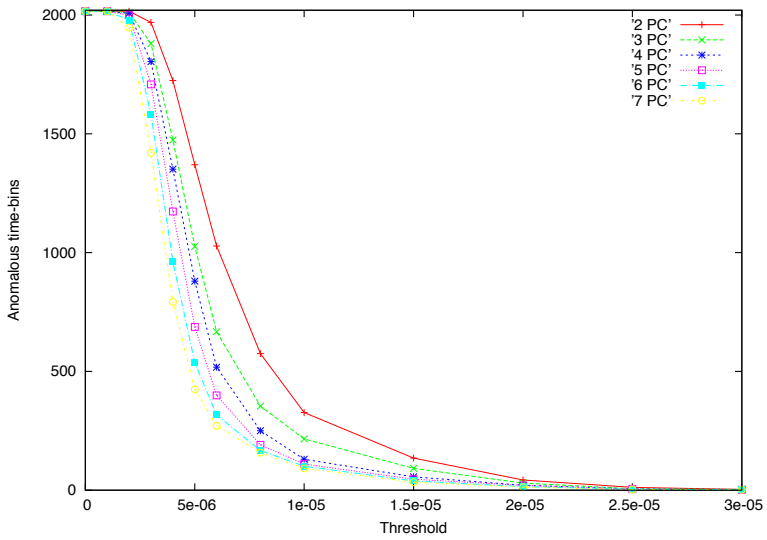


Figure 3.8: *PCA - Detected Anomalies for OD traffic aggregation*

that the IR aggregation behaves better than the OD one, but the strong dependence of this aggregation on the number of PCs makes the method hard to be tuned.

From Figures 3.11 and 3.12 we can see that the IL aggregation presents almost the same behavior of the OD aggregation. Indeed the performance do not strongly depend on the number of PCs and present the same trend when varying the threshold.

To conclude this analysis, Figures 3.13 and 3.14 present the performance achieved when using random aggregation.

As we can see this method allows us to obtain better performance

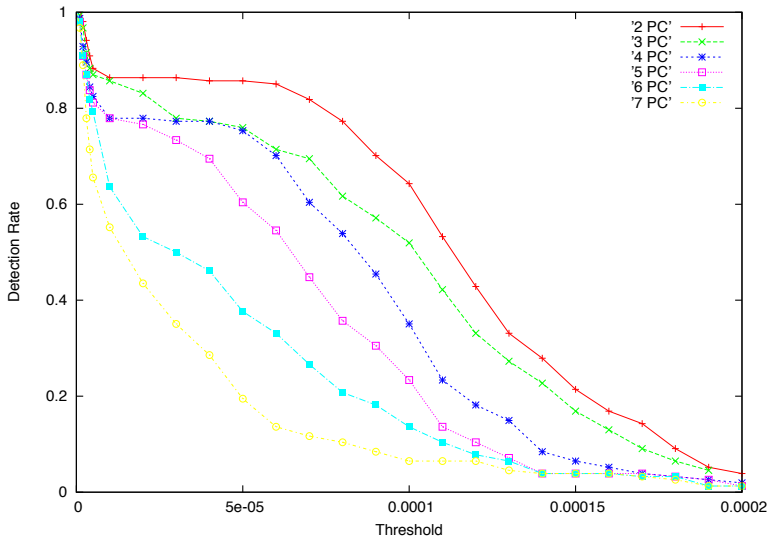


Figure 3.9: *PCA - Detection Rate for IR traffic aggregation*

in terms of detection rate: indeed the detection rate is stable over the 95%, for several values of the threshold, while the total number of detected anomalies decreases faster.

To better compare the presented results, in Table 3.1 we show the detection rate achieved by using the different aggregations, once fixed the total number of detected anomalies. It is easy to conclude that the best performance are given by the IR and random aggregations.

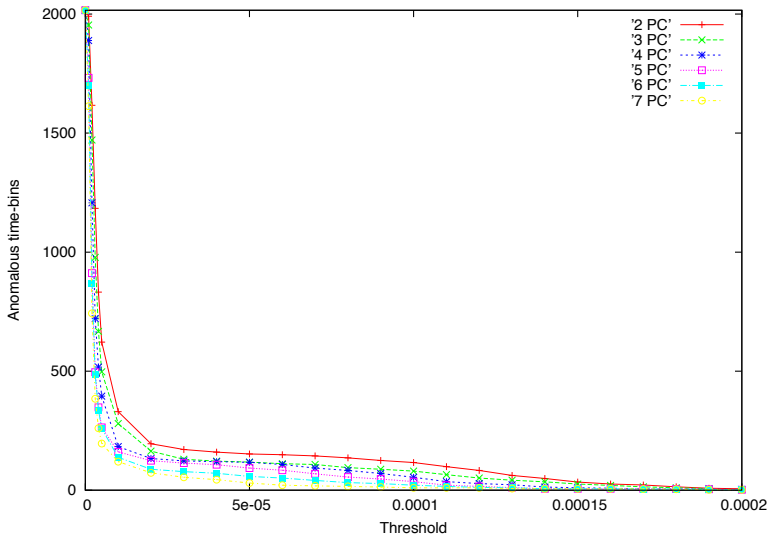


Figure 3.10: *PCA - Detected Anomalies for IR traffic aggregation*

	OD	IR	IL	Random
Detection Rate	51%	90%	43%	90%
Anomalous time-bins	910	910	910	910

Table 3.1: *PCA - Detection Rate vs. Anomalous Time-Bins*

Moreover, if we consider that in a given time-bin there can be more than an anomaly, it is clear that OD, IR, and IL aggregation do not allow us to distinguish between them, since they just reveal an anomalous time-bin, while using the sketch we can also distinguish between

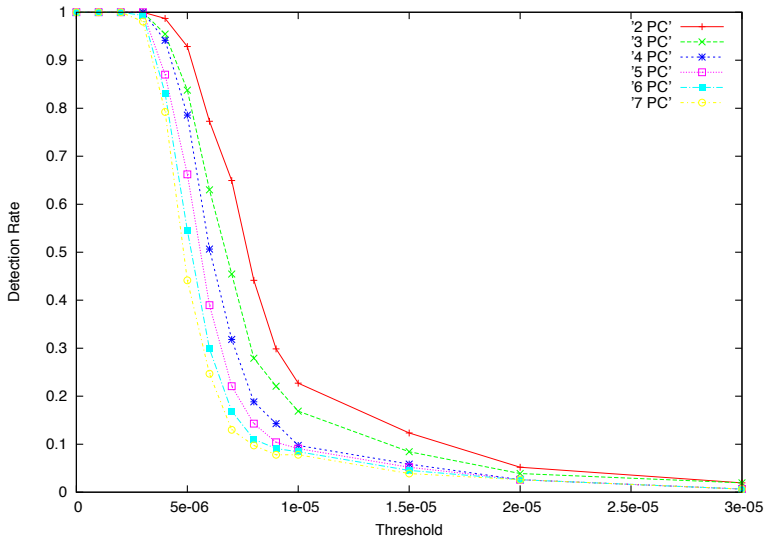


Figure 3.11: *PCA - Detection Rate for IL traffic aggregation*

contemporary anomalies, since each flow is part of several aggregates.

It is also important to highlight that we can take profit from this fact, so as to be able to identify the flows that contribute to the anomaly in a given time-bin, as previously described.

To sum up we can conclude that the best performance are obtained with the use of the random aggregation, which is also the only one that allows the identification of the anomalous flows.

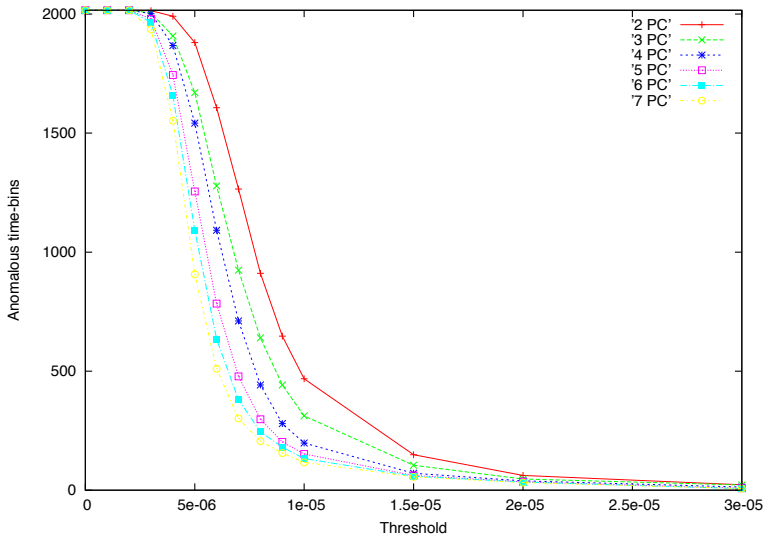


Figure 3.12: *PCA - Detected Anomalies for IL traffic aggregation*

3.3.2 KL divergence

The last session of experiments has been conducted to evaluate the effectiveness of using KL divergence instead of the entropy.

Figures 3.15 and 3.16 respectively show the detection rate and the number of detected anomalies achieved when using KL divergence. By comparing these graphs with those related to the use of entropy (Figures 3.13 and 3.14), it is easy to conclude that the performance are quite different.

Nevertheless, it is not easy to directly compare these results. To be able to perform a more significant comparison between the results

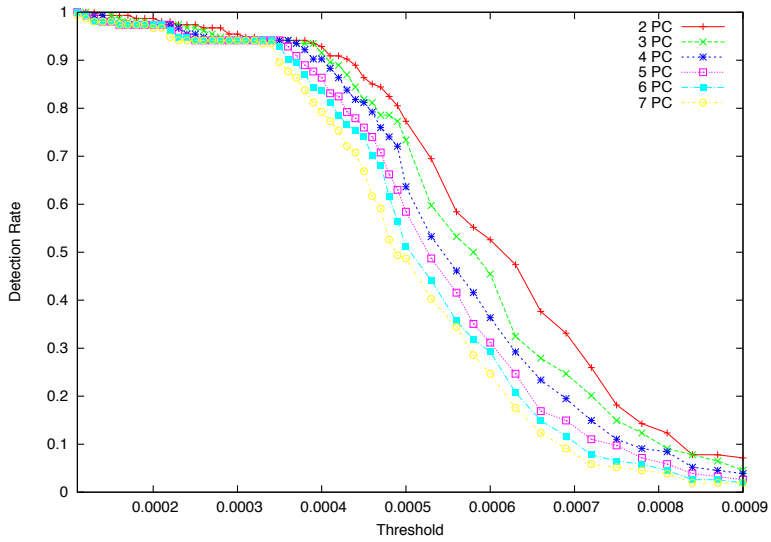


Figure 3.13: *PCA - Detection Rate for Random Aggregation*

obtained with the different approaches and to realize if, in fact, it takes to some improvements, we have evaluated some more parameters.

For this reason in Table 3.2 we present the increase in the detection rate offered by the use of KL divergence. We can notice that the combined use of entropy and KL divergence takes to great improvements in the performance. As an example, if we consider to use 5 PCs with a detection rate of 70%, we can see that the 55% of the anomalies are detected by both the systems, while the 15% detected by using the entropy is different from the 15% detected by using KL divergence, thus in this case the detection rate improves from 70% to 85%.

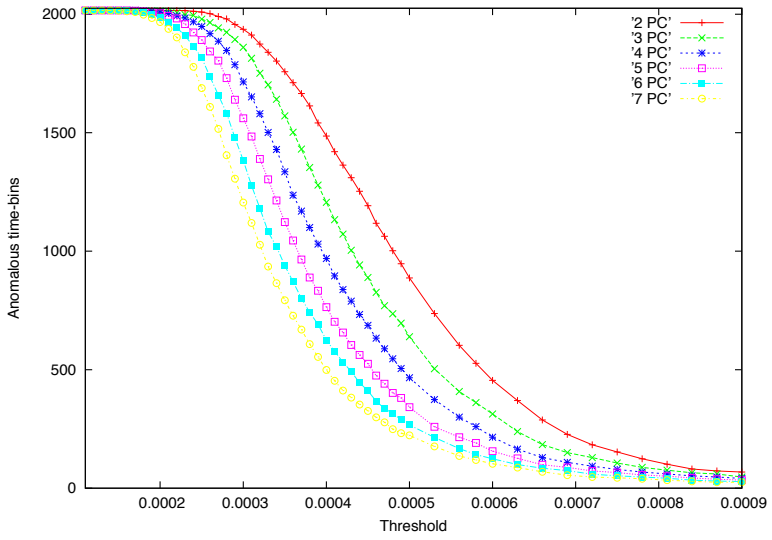


Figure 3.14: *PCA - Detected Anomalies for Random Aggregation*

PC Number	Detection Rate	Additive Detections
3	82%	10%
4	81%	9%
5	81%	10%
3	73%	15%
4	72%	16%
5	70%	15%

Table 3.2: *PCA - Kl-entropy Additive Detections*

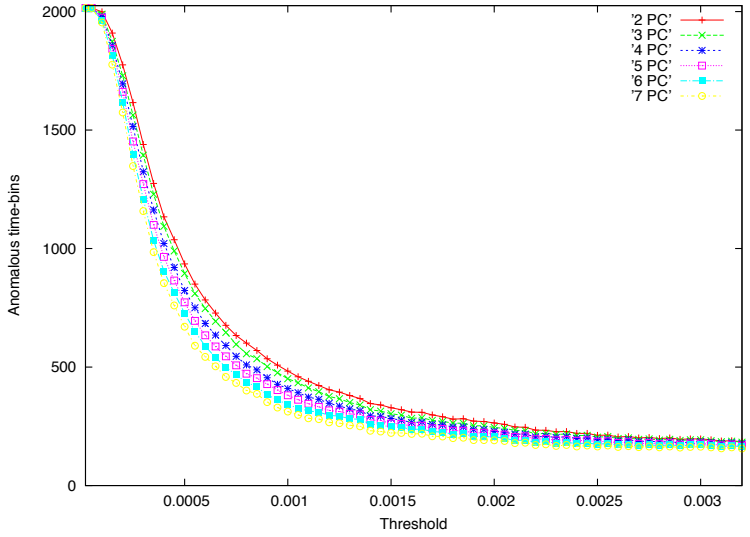


Figure 3.15: *PCA - Detected Anomalies - KL divergence*

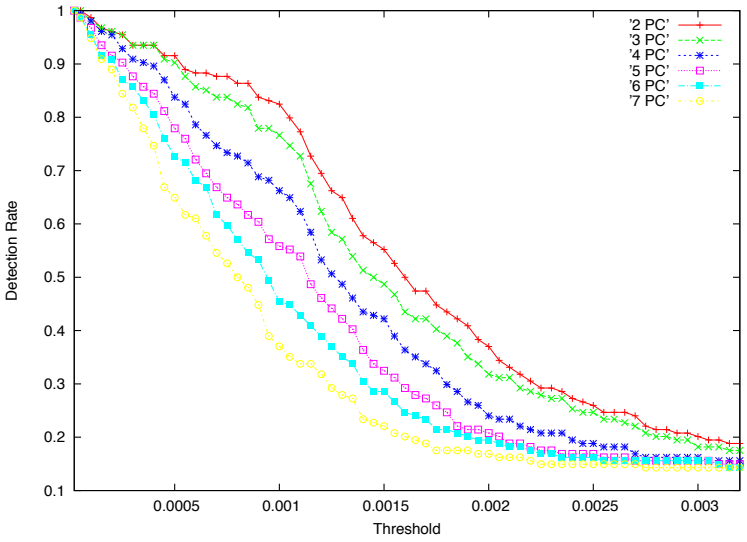


Figure 3.16: *PCA - Detection Rate - KL divergence*

Chapter 4

Wavelet Analysis

4.1 Wavelet Decomposition and Multiresolution Analysis

The Wavelet Decomposition [19] is based on the representation of any finite–energy signal $x(t) \in L^2(\mathbb{R})$ by means of its inner products with a family of functions

$$\psi_{a,b}(t) = |a|^{-\frac{1}{2}} \psi\left(\frac{t-b}{a}\right)$$

where ψ is a fixed function called *mother wavelet*.

Thus, the analysis equation for a *Continuous Wavelet Transform* (CWT) is defined as follows:

$$\mathcal{W}x(a,b) = \langle x, \psi_{a,b} \rangle = |a|^{-\frac{1}{2}} \int_{-\infty}^{\infty} x(t) \psi^* \left(\frac{t-b}{a}\right) dt$$

Starting from the CWT we can derive the *Discrete Wavelet Transform* (DWT) restricting a and b to a finite discrete set of values.

In more detail, for the discretization we can choose $a = a_0^m$, where $m \in \mathbb{Z}$ and $a_0 \neq 1$. For convenience, we can assume $a_0 > 1$. For $m = 0$, it seems natural to discretize b by taking only the integer multiples

of one fixed b_0 (for convenience $b_0 > 0$), where b_0 is appropriately chosen so that the $\psi(t - nb_0)$ cover the whole line. Varing m, the width of $a_0^{-\frac{m}{2}}\psi(a_0^{-m}t)$ is a_0^m times the width of $\psi(t)$ so that the choice $b = nb_0a_0^m$ will ensure that the discretized wavelets at level m cover the line in the same way that the $\psi(t - nb_0)$ do.

This correspond to use a set of functions $\{\psi_{m,n}(t)\}_{m,n\in\mathbb{Z}}$ defined as follows:

$$\psi_{m,n}(t) = a_0^{-\frac{m}{2}}\psi\left(\frac{t - nb_0a_0^m}{a_0^m}\right) = a_0^{-\frac{m}{2}}\psi(a^{-m}t - nb_0)$$

obtained choosing $a = a_0^m$, $b = nb_0a_0^m$, with $n, m \in \mathbb{Z}$. The values $a_0 > 1$ and $b_0 > 0$ are fixed and depend on the mather wavelet ψ.

Under quite stringent constraints on the choice of the mother wavelet, the functions $\{\psi_{m,n}(t)\}_{m,n\in\mathbb{Z}}$ may define an orthonormal dyadic wavelet basis (corresponding to $a_0 = 2$ and $b_0 = 1$). In this case, $\psi(t)$ can be represented in terms of the so–called *scaling function* $\phi(t)$

$$\psi(t) \;=\; \sqrt{2}\sum_n g_n\phi(2t - n)$$

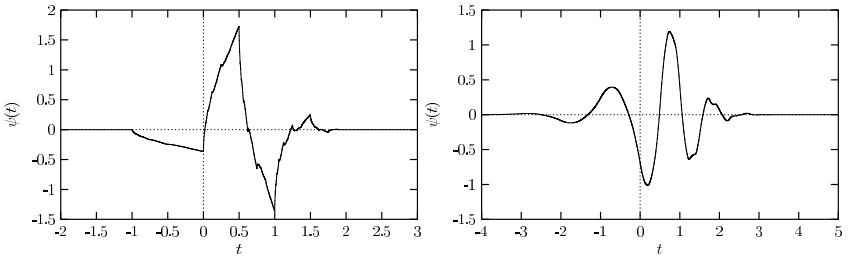
where $\phi(t)$ itself is defined by a two–scale difference equation

$$\phi(t) \;=\; \sqrt{2}\sum_n h_n\phi(2t - n)$$

with the additional constraint that $g_n = (-1)^{n-1}\,h_{-n-1}$.

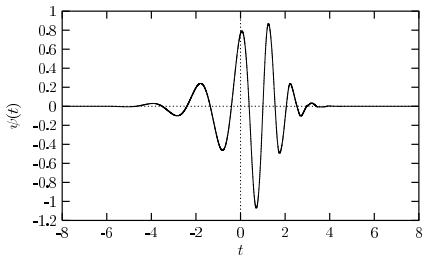
In this work, we will consider the well–known Daubechies bases family of compactly–supported mother wavelets, introduced by the Belgian mathematician Ingrid Daubechies in 1988. The number of non null coefficients h_n depends on the regularity of the mother wavelet and the number of vanishing moments: for the Daubechies basis ${}_N\psi$

of order N (with N vanishing moments) only $2N$ coefficients are non zero, so that both ${}_N\phi$ and ${}_N\psi$ has compact support of width $2N-1$. Figure 4.1 shows the Daubechies bases of order $N = 2, 5, 9$.



(a) $\psi 2$

(b) $\psi 5$



(c) $\psi 9$

Figure 4.1: *Daubechies mother wavelet*

The multiresolution analysis represents the theoretical framework for the efficient calculation of the wavelet decomposition [20]. Let $x = (x_1, x_2, \ldots)$ denote the approximation of a finite–energy signal $x(t)$ at a given resolution; the wavelet coefficients $\{x_{m,n}\}$ at lower resolutions can be obtained considering the filter bank shown in figure 4.2, where the coefficients h_n and g_n depend on the chosen mother wavelet. In particular, the outputs of the high-pass filter h_n give the

detail coefficients (at the given resolution), while the outputs of the low-pass filter give an approximation at a lower resolution, which is further decomposed in a similar way.

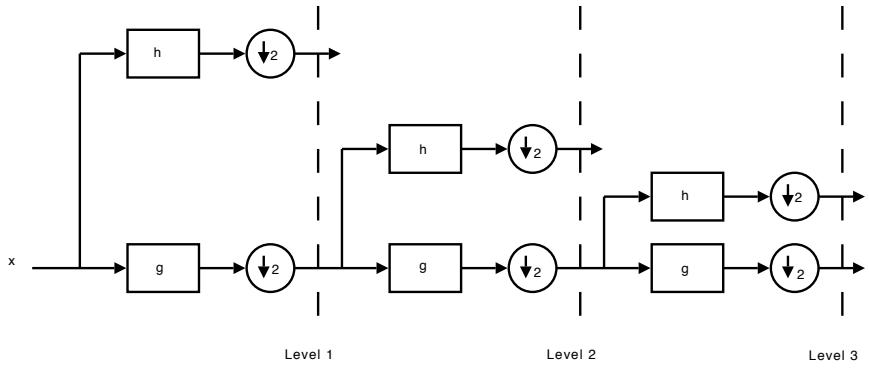


Figure 4.2: *Wavelet Decomposition*

Wavelet analysis is often used in image processing for edge detection, that is a method, which aims at identifying points at which the image brightness changes sharply or has *discontinuities*. Thus the detection of anomalies (that can be seen as discontinuities in the network traffic) seems to be a good application field for wavelet analysis.

4.2 System architecture

In this section, we present the architecture of the system we have implemented to detect anomalies in the network traffic, whose block scheme is reported in Figure 4.3. Before detailing the system, in Table 4.1 we present the notations used in the rest of the chapter.

The system is composed of four blocks:

Symbol	Meaning
ξ	threshold
A^X	detection matrix at step X
B_t	number of bytes received from ip_t, used as weight in the sketch
c_t	weigth corresponding to the key i_t
d	number of hash functions - number of rows of T_n
D_{lj}^X	Euclidean distance at step X for the time series T_{lj}
$E(D_{lj})$	average value of D_{lj}^X over all the D_{lj}^X for the element of the sketch table up to the step $X-1$
F_x	time series (traffic sent) referred to IP_x
i_t	hash key
ip_t	IP address, used as hash key in the sketch
$\{IP_x\}_{x \to X}$	set of source IP addresses that contributes to step X
k	constant factor
N	number of time-bins
p_{lj}^X	transformed coefficients at step X for the time series T_{lj}
P_{lj}	reference coefficients for the time series T_{lj}
s	sliding window length
S	number of samples of a given input file
t	input samples to wavelet transform
T_n	sketch table for time-bin n
T_{lj}	time series built from the elements (l, j) of the sketch tables
v	sliding window overlap factor
w	possible distinct outputs of the hash function - number of columns of T_n
X	detection step

Table 4.1: *Wavelet - Notations*

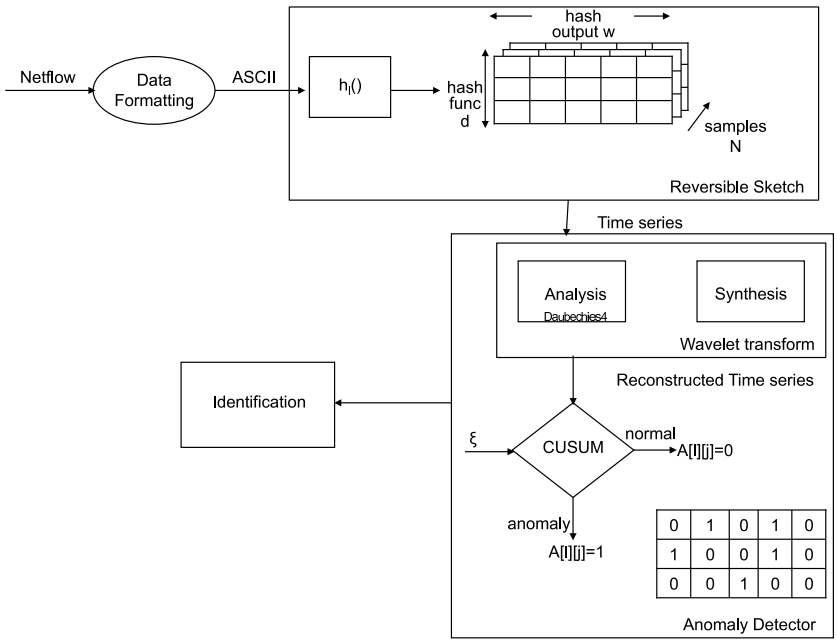


Figure 4.3: *Wavelet - System architecture*

- Data Formatting
- Sketch module
- Anomaly Detector
- Identification

In the following we have reported a detailed description of each block skipping the details related to the Data Formatting module and the Sketch module since an exhaustive analysis of these blocks has been made in Section 1.2 and 2.2 respectively.

However, it is important to highlight that as output of this two blocks we have N (number of distinct time bin) distinct hash tables. Starting from these, we consider the temporal evolution of each element T_{lj} of the sketch table, constructing $d \cdot w$ time series of N samples $T_{lj}[n]$.

4.2.1 Anomaly Detector

Wavelet coefficients computation

In this block the wavelet transform is applied to each time series. To perform such an operation the system makes use of a sliding window (see Figure 4.4) of dimension $s = 16$ samples (with an overlap factor v equal to 8 samples), and computes the transformed coefficients p_{lj}^X of each block of s samples. The first time we compute such coefficients (for the first s samples), we just store them as *reference coefficients* P_{lj}, while for all the other samples of the time series, we use such coefficients to compute the Euclidean distance D_{lj}^X between them and the *reference coefficients*.

If such a distance is lower than a given threshold ξ (see next subsection to understand how this threshold is computed) we update the *reference coefficients*, setting them to the newly computed transformed coefficients, otherwise, if $D_{lj}^X > \xi$, the system has revealed an anomaly and the *reference coefficients* are not updated.

The output of such a phase is a binary table, where the generic element (l, j) is equal to one if there is an anomaly (i.e., $D_{lj}^X > \xi$), is equal to zero otherwise.

More details about this and the subsequent phase are given in Algorithm 6.

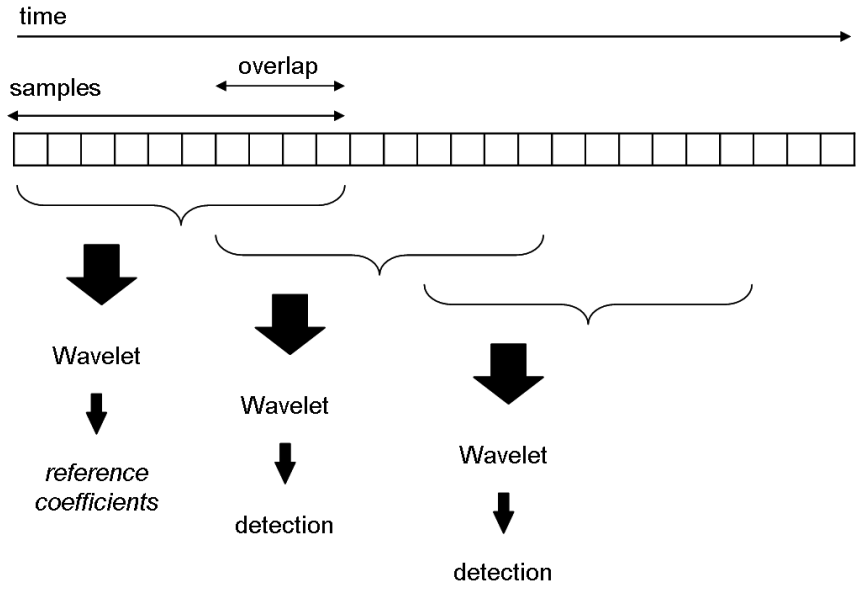


Figure 4.4: *Wavelet Analysis*

Threshold Computation

As described in the previous subsection, to evaluate the temporal evolution of the wavelet coefficients p_{lj}^X of the given time series $T_{lj}[n]$, we compute the Euclidean distance between the transformed coefficients of subsequent blocks of samples. Such a distance is then compared to a threshold ξ.

To avoid the use of a statically designed threshold, we have developed a method that allows ξ to adapt to the temporal evolution of the values of the computed distances. In more detail, ξ at a given step is set to the average value of the distances computed over all the sketch table up to the previous step plus the standard deviation multiplied by a

constant value k, that is $\xi = E(D_{lj}) + k \cdot \sigma$.

Note that, since ξ is initialized equal to zero, when the system starts working, it has a transitory phase during which it will not be able to detect anomalies.

It is worth noticing that the system, differently from many other anomaly detection algorithms, does not need a training phase during which it has to be fed with *anomaly-free traffic* (see experimental results section for more details). Indeed it just presents a transitory phase, necessary to compute the first *reference coefficients* as well as the parameters needed to compute the threshold.

4.2.2 Identification

If an anomaly has been detected during the detection phase, the system performs a new phase called anomaly identification. Note that this phase is necessary since we are not able to determine, up to this point, which flows are responsible for the revealed anomaly; indeed we can only determine in which traffic aggregation (randomly determined by the use of the sketch) the anomaly has occurred.

Given the use of the reversible sketch, we can identify the specific flows that have caused the anomaly, "simply" reversing the sketch.

4.3 Experimental results

Also in this case the proposed system has been tested using the dataset presented in Section 1.2.

To demonstrate the effectiveness of combining sketch and wavelet analysis, the results obtained by our system have been compared with

Algorithm 6 Detection

$\xi = 0$
$X = 0$
for $l = 1 : d$ **do**
  for $j = 1 : w$ **do**
    for $a = 1 : (s - v) : (N - s + 1)$ **do** ▷ sliding window: a from 1 to $(N - s + 1)$ with increment $(s - v)$
      $b = a + s - 1$
      for $r = a : b$ **do**
        $t[r] = T_{lj}[r]$ ▷ samples for the wavelet transform
      end for
      compute the transformed coefficients p_{lj}^X
      if $X == 0$ **then**
        $P_{lj} = p_{lj}^X$
      else
        compute the Euclidean distance D_{ij}^X between P_{lj} and p_{lj}^X
        $\xi = E(D_{lj}) + k\sigma(D_{lj})$
        if $D_{lj}^X \leq \xi$ **then**
          $P_{lj} = p_{lj}^X$
        else
          $A^X[l][j] = 1$ ▷ Matrix containing 1 if there is an anomaly, 0 otherwise
        end if
      end if
      $X+ = 1$
    end for
  end for
end for

those achieved by using a "classical" heavy change method built on the top of sketches, based on the evaluation of the traffic received by each aggregate in a given time-bin, with respect to the previous time-bin [21]. Note that, we have not compared our system to one simply based on the wavelet analysis of each traffic flow, because this kind of approach would be computationally unfeasible in a real scenario.

Before detailing the performance achieved by the system in terms of number of detected anomalies and detection rate, let us analyze some graphs showing the distances computed in the detection phase, comparing them with the threshold computed as described in the previous section.

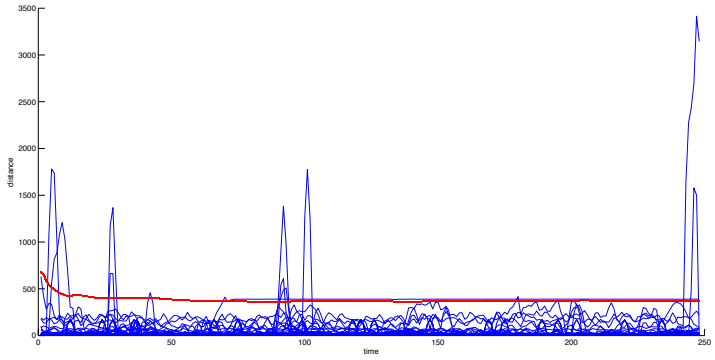


Figure 4.5: *Wavelet - Distance - Original Traces*

The Figures 4.5, 4.6 show the distances computed for the first row of the sketch table, related to the traffic of a single router (the same considerations still hold for all the other rows and routers). These distance were computed over the original traces and over the traces where we have added the anomalies. The red line in the graphs rep-

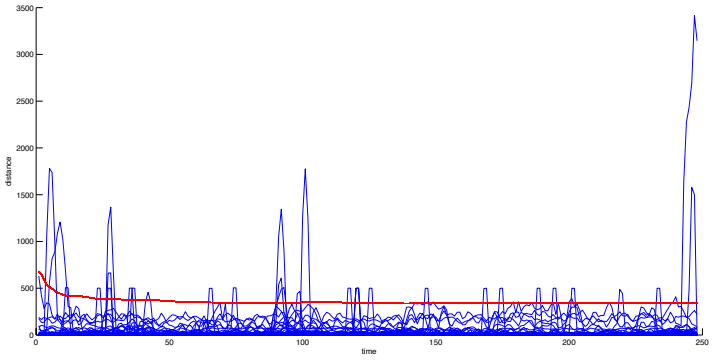


Figure 4.6: *Wavelet - Distance - Traces with artificial anomalies*

resents the threshold evolution.

In Figure 4.5 where we show the distances computed over the original traces, the distances present some peaks that, after a manual analysis of the traces, have resulted to be connected to network attacks.

Figure 4.6 shows the same distances, after that the synthetic anomalies were added. It is easy to notice that the inserted anomalies are represented by peaks in the graph that can be easily detected by our system.

The graphs in Figures 4.7-4.9 show the performance of the system, in terms of number of detected anomalies and detection rate, and compare them to the performance of a "classical" heavy change-based method.

In Figure 4.7 we plot the number of detected anomalies, both considering the original traces and those obtained after having inserted the synthetic anomalies, as a function of the threshold (i.e., of the constant factor k).

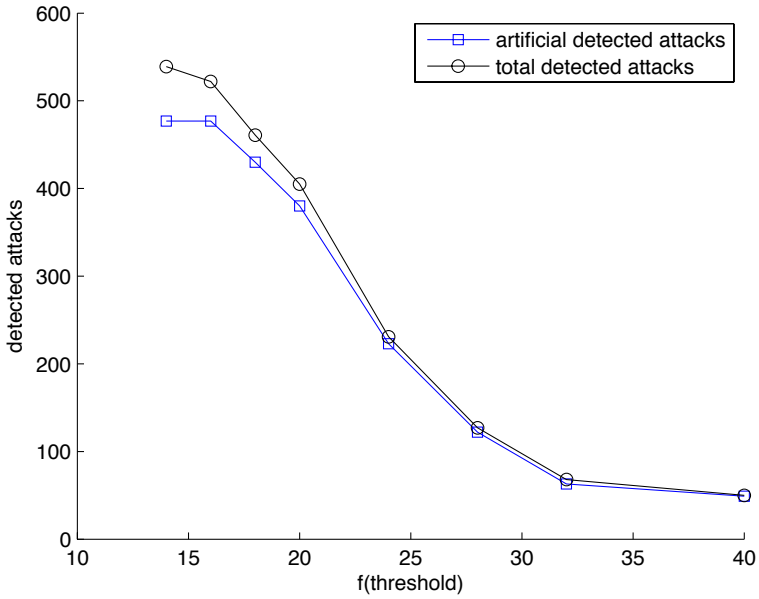


Figure 4.7: *Wavelet - Number of detected attacks*

It is easy to see that the system is able to correctly detect all the inserted anomalies (477 distinct flows), as also testified by the detection rate plotted in Figure 4.9. It is worth noticing that, when detecting all the synthetic anomalies, our system also detects some more anomalies (about 70 anomalous flows), which after a manual inspection of the traces have mainly resulted to be due to network anomalies.

Note that our approach decreases the number of false alarms with respect to the heavy change detection method, as demonstrated by Figure 4.8. Indeed, from this figure, where we report the detections for the heavy change method, we can see that the number of detected

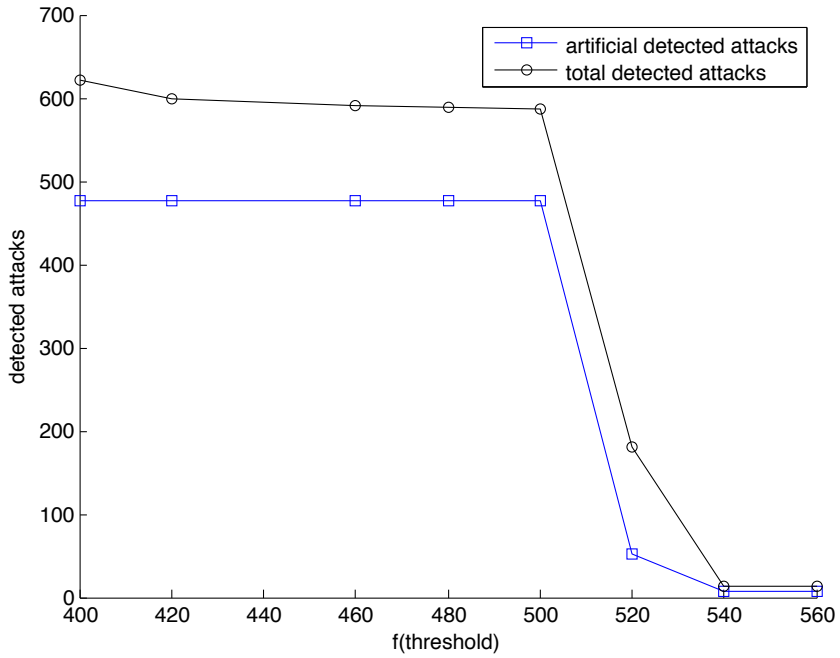


Figure 4.8: *Heavy Change Detection: number of detected attacks*

flows, not corresponding to synthetic anomalies, increases to about 120 and, in this case, the inspection of the traces has revealed that many of these flows correspond, in fact, to "normal" traffic. Moreover, we can also note that with this "classical" method the number of artificial anomalies detected by the system suddenly decreases, making hard to correctly choose the "working point" of the system.

It is also important to notice that the system has resulted to be robust to the presence of anomalies in the very initial time-bins, corresponding to a transitory phase, during which the system is not yet able to correctly compute the threshold as well as the *reference coef-*

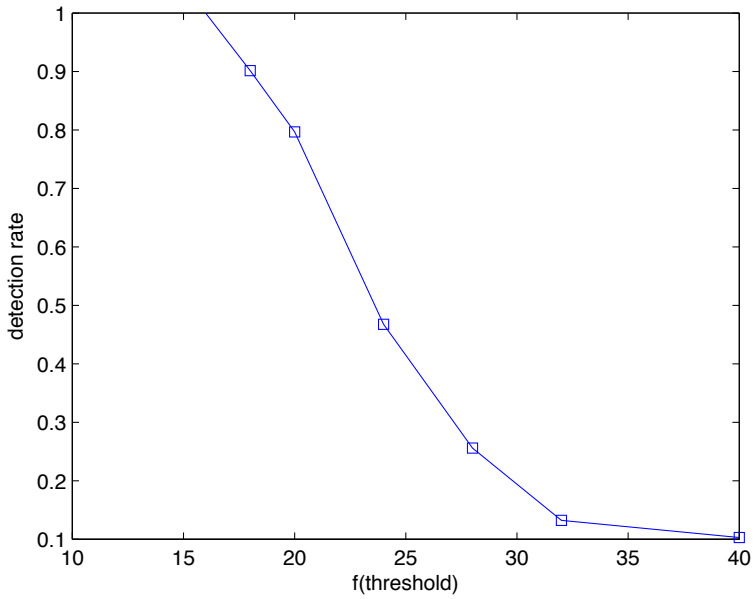


Figure 4.9: *Wavelet - Detection rate*

ficients used to detect anomalies.

Chapter 5

Change Detection

5.1 Change-Point Detection

Change-point detection is the problem of discovering time points at which properties of time series change. The main idea is that the structure of an information system can be described by a stochastic model, and that a failure leads to an abrupt change of the structure. This change occurs at an unknown point in time. The change-point detection allows us to detect such an event.

The problem can be formalized as follows. Let $X_n, n \geq 1$, be a sequence of observations that are being chosen for monitoring. The observed random variables $X_1, X_2, \ldots$ have a joint probability density function (pdf) $p_0(X_1, \ldots, X_n)$ until a change occurs at an unknown time t_a. After the change occurs, the observations have another distribution $p_1(X_1, \ldots, X_n)$. In other words, it is assumed that $X_1, X_2, \ldots$ have the conditional pdf $p_0(X_n|X_1, \ldots, X_{n-1})$ for $n < t_a$ and the conditional pdf $p_1(X_n|X_1, \ldots, X_{n-1})$ for $n \geq t_a$, where p_0 and p_1 are pre-change and post-change pdfs, respectively. Therefore, if the change occurs at time $t_a = k$, then the conditional density of the kth observation changes from $p_0(X_k|X_1, \ldots, X_{k-1})$ to $p_1(X_k|X_1, \ldots, X_{k-1})$. A

sequential change-point detection procedure is identified with a stopping time τ for an observed sequence $X_{nn\geq 1}$ i.e., the time of alarm τ, at which it is declared that a change has occurred, is a random variable depending on the observations.

The problem of discovering time points at which properties of time series change is attracting a lot of attention in the network anomaly detection field.

The idea of the approach is based on the observation that an attack leads to relatively abrupt changes in statistical models of traffic compared to the "normal mode". This changes occur at unknown points in time and should be detected "as soon as possible". Therefore, the problem of detecting an attack can be formulated and solved as a change-point detection problem.

5.1.1 Cumulative Sum control chart(CUSUM)

The CUSUM is a sequential analysis technique, typically used for monitoring change detection.

Let us suppose to have a time series, given by the samples x_n from a process, the goal of the algorithm is to detect with the smallest possible delay a change in the distribution of the data. The assumption of the method is that the distribution before and after the change ($f_{\theta_1}(x)$ and $f_{\theta_2}(x)$) are known. As its name implies, CUSUM involves the calculation of a cumulative sum, as follows:

$$\begin{aligned} S_0 &= x_0 \\ S_{n+1} &= S_n + log\big(\tfrac{f_{\theta_2}(x)}{f_{\theta_1}(x)}\big) \end{aligned} \tag{5.1}$$

The rationale behind the CUSUM algorithm is that, before the

change the quantity $log\left(\frac{f_{\theta_2}(x)}{f_{\theta_1}(x)}\right)$ is negative, whereas after the change it is positive: as a consequence, the test statistics S_n decreases before the change, and it increases linearly with a positive slope after the change, until it reaches the threshold ξ when the alarm is raised. Figure 5.1 shows an intuitive derivation of the method.

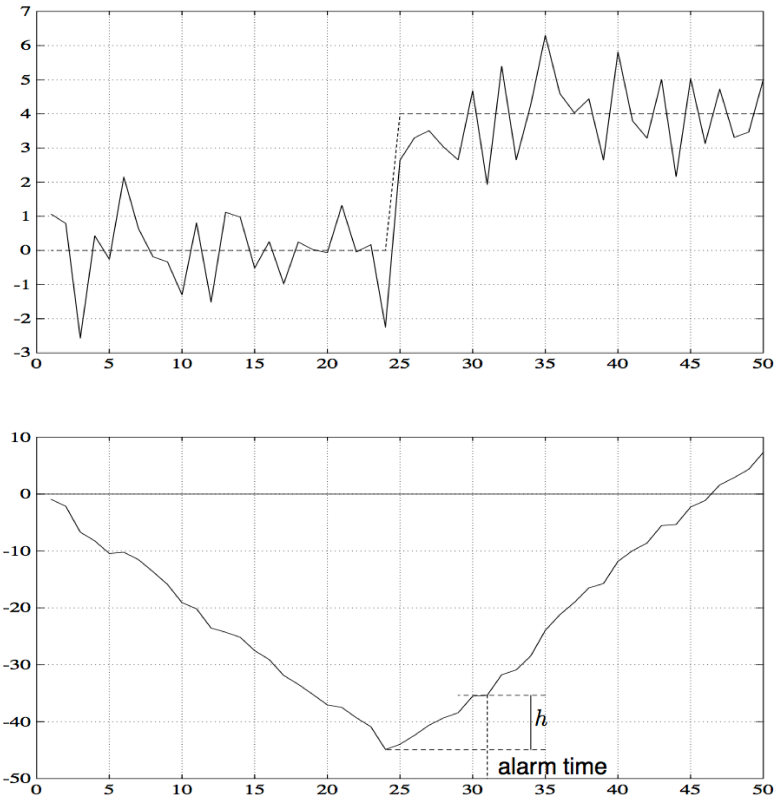


Figure 5.1: *Intuitive derivation of the CUSUM: time series (upper graph) and CUSUM statistics (lower graph)*

Note that the assumption about the knowledge of the two distributions $f_{\theta_1}(x)$ and $f_{\theta_2}(x)$, implies that CUSUM is only able to decide between two simple hypotheses. But, in case of network anomalies we cannot suppose that the distribution after the change is known (usually neither the distribution before the change is known). This implies the need of using the non parametric version of the algorithm [22], which leads to a different definition of the cumulative sum S_n. In more detail in this work we have used the multi-chart non parametric CUSUM (MNP-CUSUM), in which the quantity S_n is defined as:

$$\begin{aligned} S_0 &= x_0 \\ S_{n+1} &= (S_n + x_n - (\mu_n + c \cdot \sigma_n))^+ \end{aligned} \tag{5.2}$$

where μ_n and σ_n are the mean value and the standard deviation until step n, while c is a tunable parameter of the algorithm.

5.2 Streaming Change Detection

Using Streaming technique, the anomaly detection problem can be formulated as a Heavy Hitter (HH) detection problem or a Heavy Change (HC) detection problem. In the HH detection problem, the goal is to identify the set of flows that represent a significantly large portion of the ongoing traffic or the capacity of the link. In the HC detection problem, the goal is to detect the set of flows that have drastic change from one time period to another.

5.2.1 Heavy Hitter Detection

A HH, in a dataset, is an element whose relative frequency exceeds a specified threshold.

In more detail, given an input stream $I = \{(i_t, c_t)\}$ with the associated total sum S, a HH is a key, whose associated underlaying function $U[i]$ is not smaller than a specified portion of the expected size of the whole dataset. The problem can be formalized as follows. Given a threshold ξ $(0 < \xi < 1)$ the set of HHs is defined as:

$$HH = \{i \mid U[i] > \xi \cdot S\} \tag{5.3}$$

In the context of network anomaly detection, a HH is an entity which accounts for at least a specified portion of the total activity measured in terms of number of packets, bytes, connections, etc. A HH could correspond to an individual flow or connection. It could also be an aggregation of multiple flows/connections that share some common property, but which themselves may not be HH.

Given this, we define the HH detection problem as the problem of finding all the HHs, and their associated values, in a data stream. As an example, let us consider that the destination IP address is the *key*, and the byte count the *weight*; then in this case the corresponding HH detection problem is to find all the destination IP addresses that account for at least a portion ξ of the total traffic.

5.2.2 Heavy Change Detection

In the contest of anomaly detection, the goal of HC detection is to efficiently identify the set of flows that have drastic changefrom one

time period to another with small memory requirements and limited state information.

Modelling the data stream using the Turnstile model the problem can be formalized as follows:

A HC is a key i, whose associated underlaying function $U[i]$, evaluated in a given time-bin, significantly differs in size if compared to the same function evaluated in the previous time-bins.

For sake of simplicity, let us suppose that we want to detect the HC related to two adjacent time-bins. In this case, a key is a HC if the difference between the values evaluated in the two time-bins exceeds a given threshold (ψ). The problem can be formalized as follows. Let $U^1[i]$ and $U^2[i]$ be the values associated to the key i, evaluated in the time-bin 1 and 2 respectively, and let D_i be the relative difference, defined as $D_i = |U^1[i] - U^2[i]|$. Then the set of HCs is defined as follows:

$$HC = \{i \mid D_i > \psi\} \tag{5.4}$$

As an example, in the context of network anomaly detection, the goal of HC detection can be to identify the flows that have significant changes in traffic volume from one time period to another.

5.3 System architecture

In this section we detail the architecture of the proposed systems.

We have implemented three different systems based on the three different Change Detection approaches described in the previous sections.

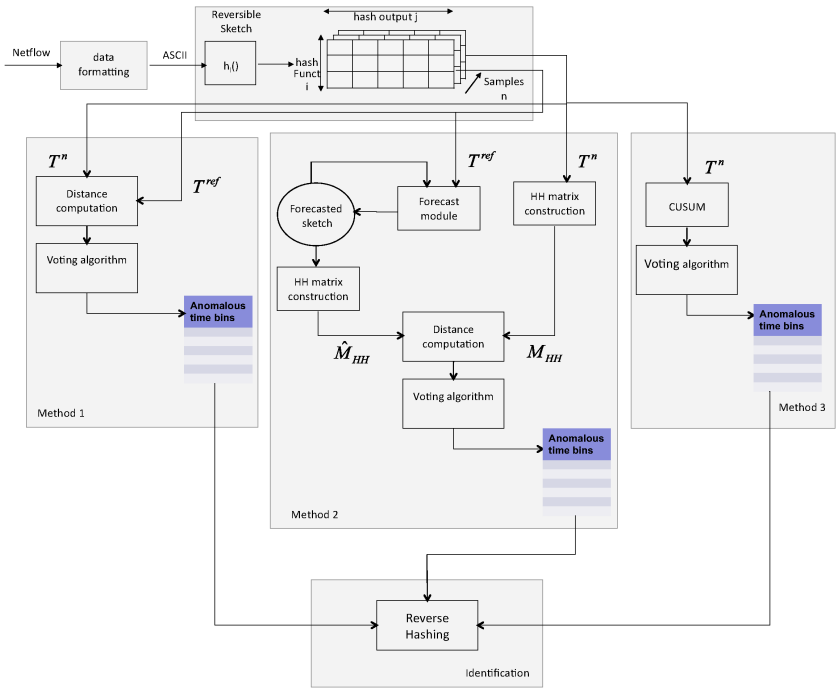


Figure 5.2: *Change Detection - System Architecture*

Figure 5.2 shows a block scheme of the systems.
We can distinguish four main blocks:

- Data Formatting
- Sketch module
- Anomaly Detector
- Identification

As you can see in the figure the three systems share the first two blocks (already described in Section 1.2) and the Identification module, while they differ for the Anomaly Detector module.

The HC-based method is called Method 1 in the figure, while Method 2 and Method 3 respectively refers to a HH change detection approach and a CUSUM based method.

The following subsections describe the Anomaly Detector module of the proposed systems.

5.3.1 Anomaly Detector

Method 1

Method 1 is a simple HC-based method that works comparing the data related to two adjiacent time-bins, that is two consecutive sketch tables.

In practice, the system computes the euclidean distance d_{ij} between each element $T[i][j]$ of the current sketch table and the corresponding element $T^{ref}[i][j]$, where $T^{ref}[i][j]$ is the element $T[i][j]$ corresponding to the last non anomalous time-bin.

In more detail, assuming that the system is processing the time-bin n, then each element of the sketch table T^n is compared with the corresponding element of the "reference" sketch table T^{ref}, where T^{ref} is equal to the "last occurred" non anomalous sketch table, i.e., $T^{ref} = T^{n-r}$ for some $(r = 1, 2, \ldots, n)$. Note that this algorithm has been introduced to avoid the "masking effect" that can be caused by anomalies that span over multiple time-bins.

If the computed distances exceed a given threshold ($d_{ij}^n > \xi$) in at least H distinct rows of the sketch table (where H is a tunable parameter), the system considers the current time-bin as anomalous and then performs the anomaly identification for revealing the responsible IP flows.

The pseudo-code related to the detection algorithm, run at each time-bin, is reported in Algorithm 7.

Method 2

This method, is much more complex than the first one but it still results to be (as demonstrated in the experimental section) suitable for on-line detection of anomalous flows. Basically, it tracks the variations in the HH distribution of the network traffic.

In more detail, as it can be seen from Figure 5.2, the sketch T is given in input to two distinct modules, namely a forecast module and a HH matrix construction module.

The forecast module takes in input its own output at the previous step and the "reference" sketch table T^{ref}, that is -as in the previous case- the "last occurred" non anomalous sketch table, and uses these two elements for forecasting the value of the next sketch table. Note

Algorithm 7 Detection: Method 1

for $i = 1 : D$ **do**
  $c_i = 0$
end for
$C = 0$
for $i = 1 : D$ **do**
  for $j = 1 : W$ **do**
    $d^n_{i,j} = |T^n[i][j] - T^{ref}[i][j]|$
    if $d^n_{i,j} > \xi$ **then**
      $c_i = 1$
    end if
  end for
  $C+ = c_i$
end for
if $C > H$ **then** ▷ H tunable parameter
  time-bin n is anomalous
else
  $T^{ref} = T^n$
end if
Output: anomalous time-bins

that the use of T^{ref} has been introduced, as in Method 1, to avoid the "masking effect" due to "long" anomalies.

The prediction phase is performed by using an Exponentially Weighted Moving Average (EWMA) forecast algorithm, described by the following equation:

$$\widehat{T^n} = \alpha T^{n-1} + (1-\alpha)\widehat{T^{n-1}}$$

or the Non-Seasonal Holt-Winters (NSHW) algorithm,

$$\widehat{T}^n = \widehat{T}_s^n + \widehat{T}_t^n$$

with

$$\widehat{T}_s^n = \alpha T^{ref} + (1-\alpha)\widehat{T}^{n-1}$$

$$\widehat{T}_t^n = \beta(\widehat{T}_s^n - \widehat{T}_s^{n-1}) + (1-\beta)\widehat{T}_t^{n-1}$$

where α and $\beta \in [0,1]$ are tunable parameters.

Algorithms 8 and 9 report the pseudo code for the forecasting phase at each time-bin using EWMA and NSHW, respectively.

Algorithm 8 Building the forecasted sketch (EWMA)

```
for i = 1 : D do
    for j = 1 : w do
        if n == 2 then
            T̂^n[i][j] = T^1[i][j]
        end if
        if n > 2 then
            T̂^n[i][j] = αT^ref[i][j] + (1 − α)T̂^{n−1}[i][j]
        end if
    end for
end for
Output: forecasted sketch
```

Algorithm 9 Building the forecasted sketch (NSHW)

for $i = 1 : D$ **do**
  for $j = 1 : w$ **do**
    if $n == 2$ **then**
      $\widehat{T_s}^n[i][j] = T^1[i][j]$
      $\widehat{T_t}^n[i][j] = T^2[i][j] - T^1[i][j]$
    end if
    if $n > 2$ **then**
      $\widehat{T_s}^n[i][j] = \alpha T^{ref}[i][j] + (\alpha - 1)\widehat{T}^{n-1}[i][j]$
      $\widehat{T_t}^n[i][j] = \beta(T_s^n[i][j] - T_s^{n-1}[i][j]) + (1 - \beta)\widehat{T_t}^{n-1}[i][j]$
    end if
    $\widehat{T}^n[i][j] = T_s^n[i][j] + T_t^n[i][j]$
  end for
end for
Output: forecasted sketch

Given this step, the system has two distinct values for the sketch at time-bin n, the real value T^n and the predicted value $\widehat{T^n}$. Both these tables are fed to a module, responsible for computing an *empirical* distribution of the HHs.

This "distribution" is computed by evaluating the HHs present in the traffic, that is the traffic aggregates (namely the sketch buckets) that exceed a given threshold, given by a percentage of the total traffic in the time-bin, S^n. The related buckets are then updated by inserting the quantity of traffic for which that aggregate exceed the threshold, while all the other buckets are set to one byte (this last point is mainly done for computational reasons). Finally each row of the matrix is normalized so as that its elements sum to one. Algorithm 10 illustrates the procedure for computing this matrix in a given time-bin.

This matrix is named M_{HH}^n if computed starting from T^n and $\widehat{M_{HH}^n}$

Algorithm 10 Building the HH matrix

for $i = 1 : D$ **do**
  for $j = 1 : w$ **do**
    $M^n_{HH}[i][j] = 1$ ▷ HH matrix initialization
  end for
end for
for $i = 1 : D$ **do**
  $NF = 0$
  for $j = 1 : w$ **do**
    if $T[i][j] - \xi S^n > 0$ **then**
      $M^n_{HH}[i][j] = T^n[i][j] - \xi S^n$ ▷ S^n total traffic in the time-bin
    end if
    $NF+ = M^n_{HH}[i][j]$
  end for
  for $j = 1 : w$ **do**
    $M^n_{HH}[i][j] = M^n_{HH}[i][j]/NF$ ▷ matrix normalization
  end for
end for
Output: matrix of the distribution of the HH

if calculated starting from $\widehat{T^n}$.

Given these two matrices, the system compares the actual HH *distribution* in M_{HH} with the forecasted one in $\widehat{M}_{HH}$. To perform such a task the system computes a distance between each line of the two matrices. In this case, the system allows us to choose between the Kullback-Leibler (KL) divergence, $KL(M^n_{HH}[i][\cdot], \widehat{M}^n_{HH}[i][\cdot])$ and the Jensen-Shannon divergence (JSD), $JSD(M^n_{HH}[i][\cdot], \widehat{M}^n_{HH}[i][\cdot])$. Note that JSD has been introduced in this system to overcome the potential limitations given by the asymmetric nature of KL. Indeed, the JSD between two generic vectors P and Q is defined as:

$$JSD(P,Q) = \frac{1}{2} \cdot KL(P,M) + \frac{1}{2} \cdot KL(M,Q) \tag{5.5}$$

where KL is defined as

$$KL(P,Q) = \sum_i p_i \cdot log(p_i/q_i) \tag{5.6}$$

and M is the "average" of P and Q, that is $m_i = (p_i + q_i)/2$.

To decide if the considered time-bin is anomalous, we have implemented a voting algorithm, that is if the computed distance exceeds a given threshold ψ for more than H rows of the matrix, the system reveals an anomalous time-bin and the anomaly is thus identified.

The whole detection phase, for a given time-bin, is described by the pseudo code in Algorithm 11.

Method 3

This third method is based on the use of the MNP-CUSUM algorithm, described in Section 5.1.1.

Algorithm 11 Detection: Method 2

$C = 0$
for $i = 1 : D$ **do**
  $d_i^n = Dist(M_{HH}^n[i][\cdot], \widehat{M}_{HH}^n[i][\cdot] = 1)$
  ▷ $Dist = JSD$ or $Dist = KL$
  if $d_i^n > \psi$ **then**
    $C++$
  end if
end for
if $C > H$ **then** ▷ H tunable parameter
  time-bin n is anomalous
else
  $T^{ref} = T^n$
end if
Output: anomalous time-bins

As it can be seen from Figure 5.2, this module takes in input the sketch T. It is worth noticing that in this method the single buckets of the sketch are used independently to construct $d \cdot w$ time series on which the CUSUM algorithm is applied. Hence, the value contained in each bucket of T is used to update the CUSUM statistics, related to the time series associated to that bucket.

About the algorithm parameters, the quantity μ and σ have been estimated by using the EWMA algorithm, while the value of the parameter c has been set equal to 0.5 (note that the algorithm has experimentally shown to be robust to the choice of this parameter).

An anomaly, in a given time series, is thus detected at a given time-bin, if the test statistics starts increasing with a positive slope and exceeds the threshold ξ (in the experimental tests the threshold ξ has been set by means of Monte Carlo Simulation).

The output of this phase is a binary matrix $A[d][w]$, for each time-bin, that contains a "1" if the time series corresponding to a given bucket is considered anomalous at that time-bin, "0" otherwise.

Note that, given the nature of the sketches, each traffic flow is part of several random aggregates (namely D aggregates), corresponding to the D different hash functions.

Due to this fact, a voting algorithm is applied to the matrix A. The algorithm simply verifies if at least H rows of A contain at least a bucket set to "1" (H is a tunable parameter).

5.3.2 Identification

In all the three cases, if the voting system outputs the presence of an anomaly in a given time-bin, the system applies the reversible sketch algorithm (described in Chapter 2) to the sketch given by the value of all the time series in that time-bin for identifying the IP flows responsible for the anomalies.

5.4 Experimental results

The proposed systems have been tested using the dataset described in Section 1.2.

Tables 5.1 to 5.12 report the results achieved by the three systems. Since, given the nature of the dataset, we cannot plot a ROC curve, in these tables we report the total number of detected anomalies and the number of synthetic anomalies detected by the systems. Note that the tables have been obtained varying the values of the threshold. The real values of such a threshold are not reported since are not significant

Threshold	Total Anomalies	Synthetic Anomalies
th_a	1969	154
th_b	1920	48
th_c	1381	28
th_d	1269	23

Table 5.1: *Change Detection - Method 1*

in themselves, just consider that the first values correspond to the highest threshold value for which the three systems detect all the 154 synthetic anomalies.

Hence, in general, the system is always able to obtain a 100% detection rate (revealing all the 154 synthetic anomalies), but the performance can be strongly different depending on the total number of detected anomalies that has a direct impact on the number of false alarms.

To start with, let us analyze the performance offered by the "classical" HC-based system, reported in Table 5.1. In this case, we can easily see that for detecting all the synthetic anomalies, we have to accept a total number of detection equal to 1969, which is not acceptable. Moreover the number of detected synthetic anomalies suddenly decreases when increasing the threshold, while the number of total detected anomalies remains quite stable, making very hard the application of the system in the "real world".

Concerning Method 2, a first set of experimental tests has been realized to evaluate the impact of the sketch dimension (namely the impact of the parameter w) on the system performance. The ob-

Threshold	Total Anomalies	Synthetic Anomalies
ψ_1	1018	154
ψ_2	602	152
ψ_3	254	145
ψ_4	192	142
ψ_5	177	135
ψ_6	157	127
ψ_7	137	109
ψ_8	115	90
ψ_9	91	70
ψ_{10}	65	46

Table 5.2: *Change Detection - Method 2 ($w = 256$, EWMA $\alpha = 0.2$, JSD)*

Threshold	Total Anomalies	Synthetic Anomalies
ψ_1	310	154
ψ_2	256	152
ψ_3	199	148
ψ_4	179	144
ψ_5	172	142
ψ_6	167	137
ψ_7	163	133
ψ_8	151	123
ψ_9	135	111
ψ_{10}	115	94

Table 5.3: *Change Detection - Method 2 ($w = 512$, EWMA $\alpha = 0.2$, JSD)*

Threshold	Total Anomalies	Synthetic Anomalies
ψ_1	210	154
ψ_2	191	151
ψ_3	175	145
ψ_4	164	136
ψ_5	144	123
ψ_6	125	105
ψ_7	80	61
ψ_8	45	26
ψ_9	34	16
ψ_{10}	24	6

Table 5.4: *Change Detection - Method 2 ($w = 1024$, EWMA $\alpha = 0.2$, JSD)*

tained results are presented in Tables 5.2 - 5.4 and correspond to three distinct values of w, namely $w = 256$, $w = 512$, and $w = 1024$. Regarding the other parameters, these tests have been realized using the EWMA forecasting algorithm (with $\alpha = 0.2$) and the JSD divergence.

Note that varying the value of the parameter w, not only does it have an impact on the memory and computational resource consumption (that would take to choose the lowest possible value for w), but it also determines the dimension of the traffic aggregates. Indeed, sketches are implicitly used for randomly aggregating the traffic flows. Thus a low value of w means having few big aggregates and vice-versa a big value of w means having many small aggregates. It is quite obvious that this can influence the possibility of detecting anomalous flows in the aggregates.

From Table 5.2 we can easily see that using a sketch of dimension $w = 256$ takes to unacceptable performance. Indeed, in this case, for having a 100% detection rate, we have 1018 total detection of anomalous time-bins (over a total of 2016) that correspond to a high number of false alarms.

Things get definite better when considering $w = 512$ (see Table 5.3) and $w = 1024$ (see Table 5.4). Indeed in these cases the system is able to detect all the synthetic anomalies, with a total number of detections of 310 and 210 respectively. To really evaluate the performance of the two settings, we have performed a manual verification of the dataset, checking the additional detections of the system. After that we can conclude that, almost all the additional detections obtained with $w = 512$ (310 total detections minus the 154 synthetic anomalies) are real anomalies already present in the traces. From this analysis we can thus conclude that the best performance are obtained with $w = 512$, indeed $w = 256$ takes to a big number of false alarms, while $= 1024$ takes to some false negatives (missed detections).

Moreover, note that, in any case, event though all the additional detections obtained with $w = 512$ would not be "real" anomalies they would correspond to a maximum false alarm rate of 8.3% that could be considered as "acceptable".

We can also easily notice, by analyzing Table 5.3, that the number of detected synthetic anomalies varies quite slowly when increasing the value of the threshold, while the number of total detection decreases much faster. This fact makes easy the tuning of the system.

The second and third sets of experimental tests have been conducted to tune the parameter of the forecasting algorithms, EWMA and

Threshold	Total Anomalies	Synthetic Anomalies
ψ_1	473	154
ψ_2	333	153
ψ_3	246	150
ψ_4	208	146
ψ_5	190	144
ψ_6	172	137
ψ_7	167	134
ψ_8	150	121
ψ_9	1697	117
ψ_{10}	125	100

Table 5.5: *Change Detection - Method 2 ($w = 512$, EWMA $\alpha = 0.5$, JSD)*

Threshold	Total Anomalies	Synthetic Anomalies
ψ_1	688	154
ψ_2	344	152
ψ_3	284	149
ψ_4	225	146
ψ_5	1320	142
ψ_6	276	136
ψ_7	157	122
ψ_8	1697	119
ψ_9	1767	109
ψ_{10}	126	85

Table 5.6: *Change Detection - Method 2 ($w = 512$, EWMA $\alpha = 0.8$, JSD)*

NSHW respectively.

In more detail, Tables 5.3, 5.5, and 5.6 present the results achieved by the system using a sketch table of dimension $w = 512$, the JSD divergence, and three distinct value of the smoothing parameter of the EWMA algorithm, namely $\alpha = 0.2$, $\alpha = 0.5$, and $\alpha = 0.8$.

From the tables we can notice that when increasing the value of the smoothing parameter, we have an increase in the number of total detection. Indeed for detecting all the synthetic anomalies the system detects a total number of anomalies equal to 310 (case $\alpha = 0.2$), 473 (case $\alpha = 0.5$), or 688 (case $\alpha = 0.8$). Also in this case, a manual verification of the dataset has highlighted that most of the additional detections obtained varying α corresponds to false alarms.

Moreover, additional tests (not shown for sake of brevity) have demonstrated that varying the smoothing parameter around the value 0.2 (i.e., $\alpha \in [0.1, 0.3]$), does not take to any significant variation in the system performance.

From these considerations we can conclude that the best performance are obtained when $\alpha = 0.2$. Note that this is also supported by the literature, indeed it is known that 0.2 is in the typical range of the "optimal" smoothing parameter.

Moreover, using a low value of the smoothing parameter implies the use of a model not much responsive to the fluctuations in the data. This has a direct impact on our system performance. Indeed, by analyzing Tables 5.5 and 5.6 we can notice that the system present a "strange" behavior. Indeed the total number of detections is not always decreasing, when increasing the value of the threshold. This is due to the presence of "noisy samples" in the data and it is hence

Threshold	Total Anomalies	Synthetic Anomalies
ψ_1	402	154
ψ_2	299	152
ψ_3	237	146
ψ_4	206	144
ψ_5	187	139
ψ_6	166	133
ψ_7	159	128
ψ_8	137	111
ψ_9	122	98
ψ_{10}	107	87

Table 5.7: *Change Detection - Method 2 ($w = 512$, NSHW $\alpha = 0.2$ $\beta = 0.2$, JSD)*

mitigated when the parameter α tends to zero.

Analogously to what done for tuning the smoothing parameter of the EWMA algorithm, Tables 5.7 - 5.9 present an analysis of the system performance obtained varying the value of the parameter β of the NSHW algorithm (for sake of brevity we do not show the results corresponding to different values of α, since they are similar to those obtained for EWMA). In more detail the presented results have been obtained by using a sketch dimension $w = 512$, the JSD divergence, $\alpha = 0.2$, and three distinct values of β, namely $\beta = 0.2$ (Table 5.7), $\beta = 0.5$ (Table 5.8), and $\beta = 0.8$ (Table 5.9).

For these tables, the considerations done for the previous set of tests are still valid and take us to conclude that the best value for the β parameter is $\beta = 0.2$.

Threshold	Total Anomalies	Synthetic Anomalies
ψ_1	531	154
ψ_2	372	153
ψ_3	279	150
ψ_4	237	144
ψ_5	196	139
ψ_6	185	133
ψ_7	165	129
ψ_8	154	118
ψ_9	1695	113
ψ_{10}	122	94

Table 5.8: *Change Detection - Method 2 ($w = 512$, NSHW $\alpha = 0.2$ $\beta = 0.5$, JSD)*

Threshold	Total Anomalies	Synthetic Anomalies
ψ_1	660	154
ψ_2	436	152
ψ_3	322	150
ψ_4	252	145
ψ_5	206	140
ψ_6	186	137
ψ_7	185	132
ψ_8	158	121
ψ_9	1696	115
ψ_{10}	146	99

Table 5.9: *Change Detection - Method 2 ($w = 512$, NSHW $\alpha = 0.2$ $\beta = 0.8$, JSD)*

Threshold	Total Anomalies	Synthetic Anomalies
ψ_1	968	154
ψ_2	538	153
ψ_3	329	150
ψ_4	276	147
ψ_5	1326	146
ψ_6	412	136
ψ_7	1319	127
ψ_8	1768	122
ψ_9	1767	111
ψ_{10}	659	89

Table 5.10: *Change Detection - Method 2 ($w = 512$, no forecasting, JSD)*

Given these results, we can make a comparison between the use of the EWMA and the NSHW algorithms, by comparing Table 5.3 and 5.7 that correspond to the best settings for the two considered cases. The inspection of the dataset takes us to conclude that the best performance are achieved when using the EWMA algorithm.

Table 5.10 present the results of the system when disabling the forecasting module ($\widehat{M}_{HH}$ is directly computed starting from T^{ref}). By comparing this table with the previous ones, we can see that disabling the forecasting module takes to worsen the performance. This result was predictable, indeed disabling the forecasting module is equivalent to use the EWMA algorithm with $\alpha = 1$, and the previous analysis had already highlighted that the best performance are achieved with low values of the smoothing parameter.

Lastly, Table 5.11 presents the performance achieved using the KL

Threshold	Total Anomalies	Synthetic Anomalies
ψ_1	221	154
ψ_2	197	152
ψ_3	182	148
ψ_4	178	144
ψ_5	174	141
ψ_6	168	137
ψ_7	161	130
ψ_8	135	104
ψ_9	115	86
ψ_{10}	90	63

Table 5.11: *Change Detection - Method 2 ($w = 512$, EWMA $\alpha = 0.2$, KL)*

divergence instead of the JSD divergence. Note that JSD divergence has been introduced to overcome the potential limitations due to the non-symmetric nature of the KL divergence, but no evidence is provided in the literature to conclude that it can offer better performance in our case. Nevertheless the results presented in Table 5.11 (and the manual inspection of the dataset), compared to those in Table 5.3, confirm our intuition.

After this analysis we can thus conclude that the presented system outperforms the "classical" HC-based methods and that the best settings are those corresponding to the results presented in Table 5.3, that is $w = 512$, EWMA algorithm with $\alpha = 0.2$, and JSD divergence.

Instead, concerning the CUSUM based method (see Table 5.12), we can conclude that the system behaves very similarly to Method 2 (with

$w = 512$ and $\alpha = 0.2$), when reaching a detection rate of 100%. Indeed when detecting all the 154 synthetic anomalies, the system only detects 135 more anomalies, most of them corresponding to "real" anomalies already present in the traces (as for Method 2). But its performance are much less stable than those of Method 2, when raising the threshold. Indeed, when detecting a total number of about 180 anomalies we can see that Method 3 only detects 35 synthetic anomalies, while our proposed method (Method 2) still detects 144 synthetic anomalies. This greater stability of our system makes easier the tuning of the system parameters and makes the method more suitable for "real world" application.

Finally, to evaluate the computational complexity in time and memory space of Method 2 (we have not performed the same kind of test for Method 1 and Method 3 because they are computationally much easier), we have used a general purpose PC, equipped with an Intel Core 2 Duo processor at 3GHz and 2GB of RAM. The experimental results have shown that the system is able to process a whole week of traffic from the Abilene/Internet2 network in about 531s, with a maximum memory consumption of 0.9% (about 1.8 MB). In more detail the system is able to analyze a single time-bin of 5 minutes of traffic related to a single router in about 29ms, demonstrating to be suitable for the on-line detection of anomalies in backbone networks.

Threshold	Total Anomalies	Synthetic Anomalies
th_A	289	154
th_B	287	153
th_B	178	35
th_B	160	14
th_C	154	12
th_D	134	10
th_E	125	8
th_F	107	7
th_G	94	6
th_H	88	5
th_I	73	4
th_L	47	3

Table 5.12: *Change Detection - Method 3*

Chapter 6

Wave-CUSUM - combining Wavelet and CUSUM

6.1 Wavelet Pre-Filtering

The direct application of the methods described in the previous chapters is sometimes difficult because of the time-varying nature of the traffic (e.g., daily and weekly trends) that makes somehow hard to distinguish a network anomaly from a "normal" variation of the distribution of the traffic.

To solve such an issue, we propose to combine a "classical" CUSUM based approach together with the wavelet analysis. In more detail the latter is used to filter out the seasonal trends in the network traffic before applying the "real" anomaly detection algorithm, based on the CUSUM method.

The main idea is used the Wavelet Decomposition for organizing the data onto different *components*, that is a hierarchy of component

"signals".

The lower *component* contains very sparse filtered information that can be thought of as sophisticated aggregations of the original data. We refer to that part of the representation as the low-frequency representation. In contrast, the very high strata in the hierarchy capture fine-grained details of the data, such as spontaneous variations. These are referred to as the high-frequency strata.

In more detail the method works as follows.

At first, we extract from the original signal the aforementioned hierarchy of derived signals. This is done as an iterative process.

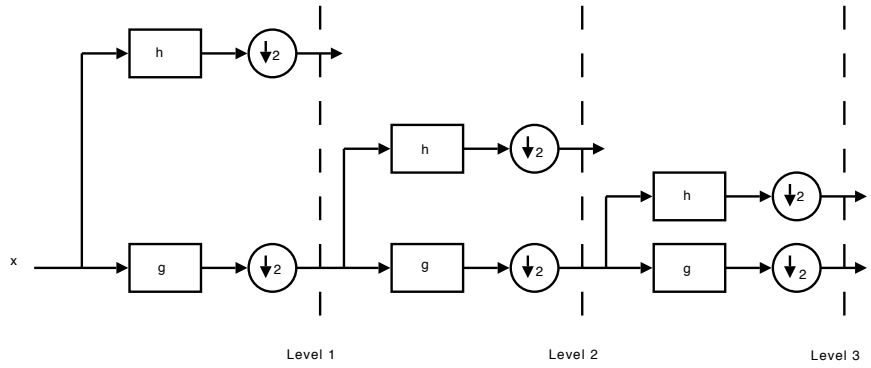


Figure 6.1: *Wavelet Decomposition*

The input for each iteration is a signal x (see Figure 6.1) of length N. The output is a collection of two or more derived signals, each of which is of length $N/2$. Each output signal is obtained by convolving x with an specially designed filter F and then decimating every other coefficient of that convolution product. We denote by $F(x)$ the output signal so obtained.

One of the special filters, denoted in the figure as g, has a smoothing/averaging effect, and its corresponding output $g(x)$ is the low-frequency output. Instead, the output of the other filter $h(x)$ should capture only the fine-grained details, i.e. the high-frequency content of the signal x.

The iterations proceed with the further decomposition of $g(x)$, creating the (shorter) signals $g^2(x), h_1 g(x)$. Continuing in this manner, we obtain a family of output signals of the form $h_i g^{j-1}(x)$. The index j counts the number of low-pass filtering iterations applied to obtain the output signal: the larger the value of j, the lower the derived signal is in our hierarchy. Indeed, we refer to $h_i g^{j-1}(x)$ as belonging to the jth frequency level, and consider a higher value of j to corresponding to a lower frequency. If our original signal x consists of measurements taken at five minute intervals, then the derived signal $h_i g^{j-1}(x)$ consists of data values that are $2^j \times 5$ minutes apart one from the other. Thus, as j grows, the corresponding output signal becomes shorter and records a smoother part of the signal. The values of the derived signals $h_i g^{j-1}(x)$ are known as the *wavelet coefficients*.

Instead, if we perform the synthesis iterations at each step the input signals for the iteration are $g^j(x), h_i g^{j-1}(x)$, and the output is the signal $g^{j-1}(x)$. This is exactly the inverse of the jth iteration of the analysis algorithm. By employing that step sufficiently many times, one recaptures the original signal.

Instead, if we would like to ignore some information we can alter some of the values of some of the derived signals of the decomposition step and then applying reconstruction. The general idea is to suppress all the values that carry information that we would like to ignore. For

example, if we wish to view the fine-grained spontaneous changes in the data only, we will apply a threshold to the entries in all the low-frequency levels, i.e. replace them by zeros.

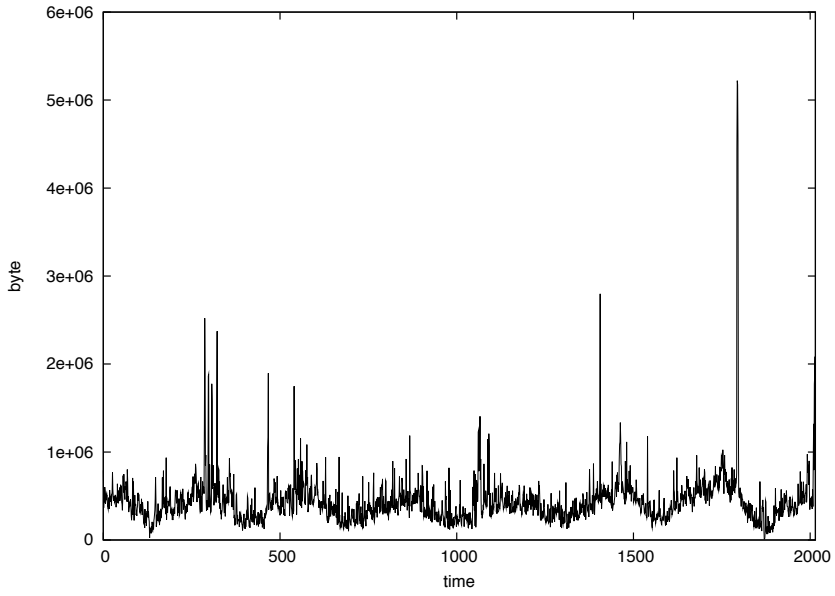


Figure 6.2: *Original Signal (traffic aggregate over one week)*

Thus, using the wavelet analysis we are able to extract three different output signals:

- Low-frequency component (Figure 6.3)
- Mid-frequency component (Figure 6.4)
- High-frequency component (Figure 6.5)

The Low-frequency component (L-component) is obtained by synthesizing all the low-frequency wavelet coefficients from levels 9 and

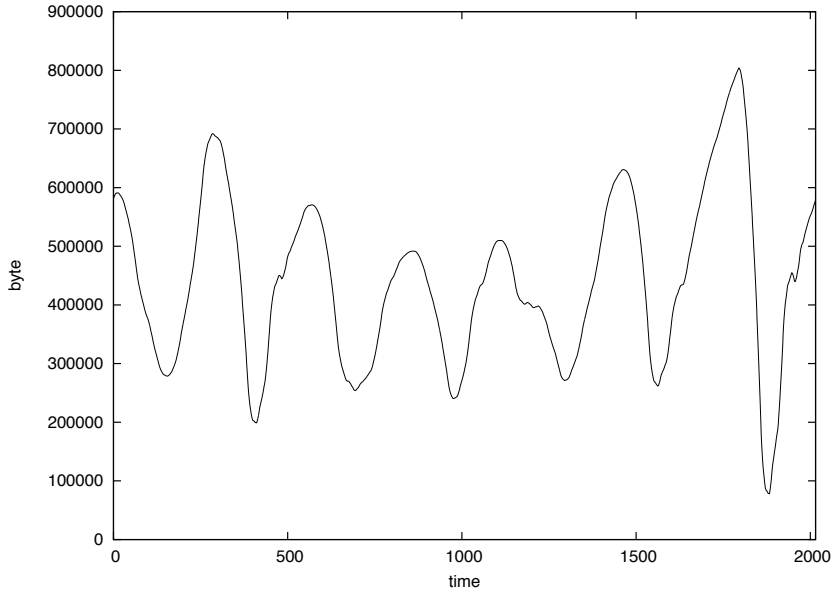


Figure 6.3: *Reconstructed Signal - Low-frequency Component*

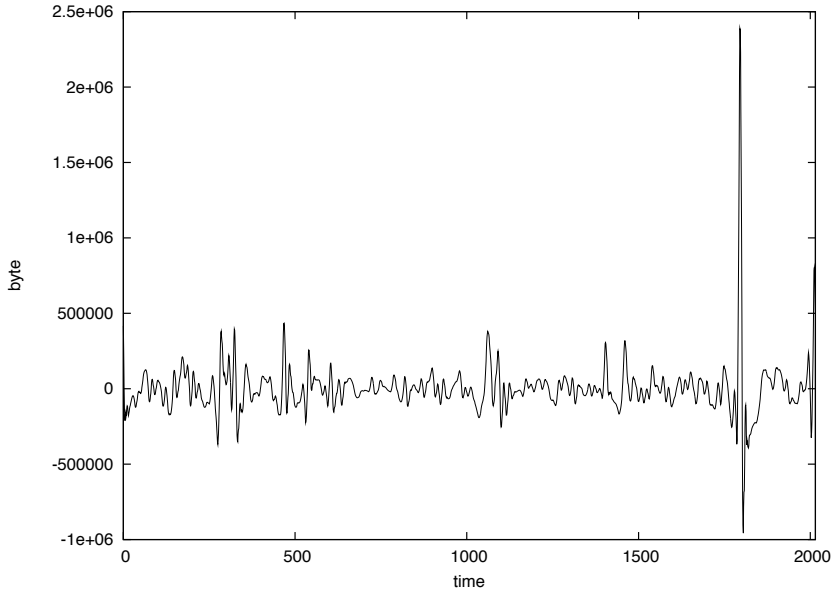


Figure 6.4: *Reconstructed Signal - Mid-frequency Component*

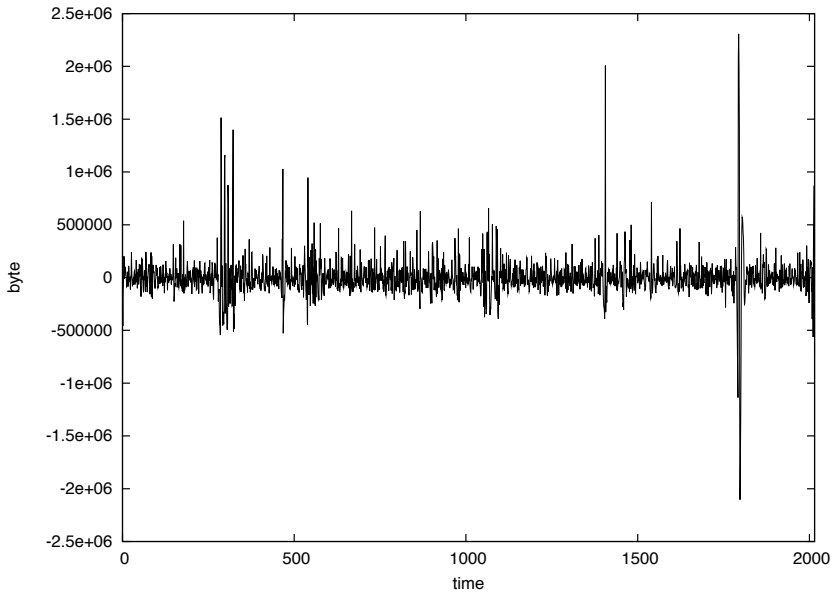


Figure 6.5: *Reconstructed Signal - High-frequency Component*

up. It is able to capture patterns and anomalies of very long duration (several days and up). The signal here is very sparse, and captures weekly patterns in the data quite well. For many different types of Internet data, the L-component of the signal reveals a very high degree of regularity and consistency in the traffic, hence can reliably capture anomalies of long duration.

The Mid-frequency component (M-component), instead, is obtained by synthesizing the wavelets coefficients from frequency levels 6, 7, 8. The signal here has zero-mean, and is supposed to mainly capture the daily variations in the data.

Finally, the High-frequency component (H-component) is obtained

by thresholding the wavelet coefficients in the first 5 frequency levels, i.e. setting to zero all coefficients whose absolute value falls below a chosen threshold (and setting to zero all the coefficients in level 6 and up). The need for thresholding stems from the fact that most of the data in the high-frequency component consists of small short-term variations, variations that we think of as "noise" and do not aid us in our anomaly detection objective.

6.2 System Architecture

In this section we detail the architecture of the proposed Intrusion Detection System, (Figure 6.6 depicts the proposed architecture).

The system is composed of four blocks:

- Data Formatting
- Sketch Module
- Anomaly Detector
- Identification

In the following we describe the Anomaly Detector and the Identification module. Instead, for an exhaustive analysis of the first two blocks refer to Section 1.2.

Note that at the end of this first two phases, given that we had N distinct time-bins, we have obtained N distinct sketch tables $T^n_{D\times w}$, where $n \in (1, 2, \ldots, N)$ is the time-bin.

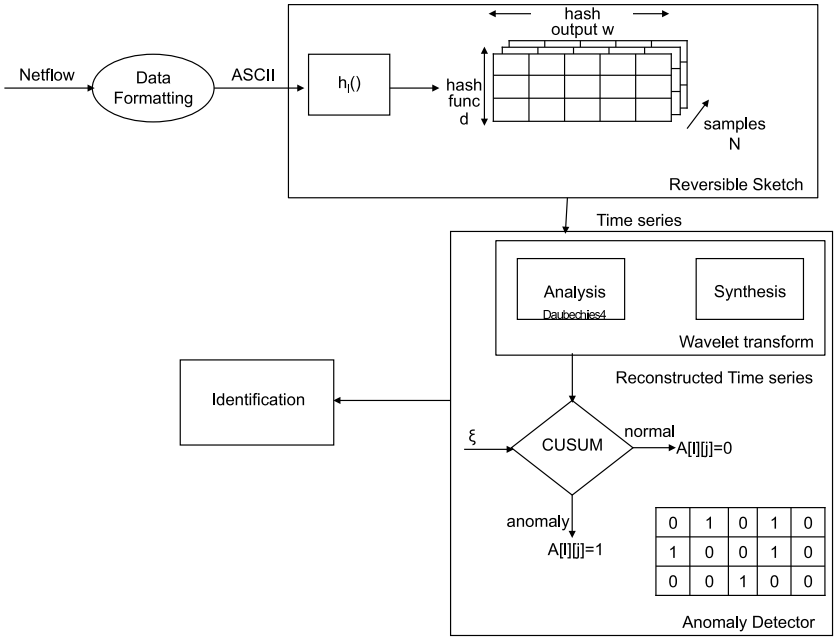


Figure 6.6: *Wave-CUSUM - System Architecture*

6.2.1 Wavelet Module

At this point, starting from the N distinct sketch tables we consider the temporal evolution of each bucket T_{lj} of the sketch table, constructing $d \cdot w$ time series of N samples $T_{lj}[n]$.

These time series are given as input to the wavelet module, which computes the coefficients of the transformed signal. The wavelet decomposition is performed on six steps (namely we obtain six levels of transformed coefficients) by using Daubechies-4 as mother wavelet. To filter out the daily and weekly trends the signal is then reconstructed by using the coefficients from levels one to five, and inserting null coefficients in the sixth level. The result of this synthesis operation (as shown in Section 6.3) is the original signal without the seasonal behavior.

Note that, taking the value of the different time series at each timebin, we are able to reconstruct the corresponding sketch table (this will be used in the following for the detection and identification phases).

6.2.2 Detection Module

The reconstructed time series obtained after the wavelet analysis and synthesis, are given in input to the detection module, where the MNP-CUSUM algorithm is performed (analogously to what already described in Section 5.3.1).

About the MNP-CUSUM algorithm it is worth noticing that the quantity μ and σ have been estimated by using the EWMA algorithm, while the value of the parameter c has been set equal to 0.5 (also note that the algorithm has experimentally shown to be robust to the choice of this parameter).

An anomaly, in a given time series, is thus detected at a given time-bin, if the test statistics starts increasing with a positive slope and exceeds the threshold ξ. Note that, in the experimental tests the threshold ξ has been set by means of Monte Carlo Simulation.

The output of this phase is a binary matrix ($A[d][w]$), for each time-bin, that contains a "1" if the time series corresponding to a given bucket is considered anomalous at that time-bin , "0" otherwise.

Note that, given the nature of the sketches, each traffic flow is part of several random aggregates (namely D aggregates), corresponding to the D different hash functions. This means that, in practice, any flow will be checked D times to verify if it presents any anomaly (this is done because an anomalous flow could be masked in a given traffic aggregate, while being detectable in another one).

Due to this fact, a voting algorithm is applied to the matrix A. The algorithm simply verifies if at least H rows of A contain at least a bucket set to "1" (H is a tunable parameter). If so, the mediator reveals an anomaly, otherwise the matrix A is discarded.

6.2.3 Identification phase

In case the voting system outputs the presence of an anomaly in a given time-bin, the system applies the reversible sketch algorithm to the sketch table given by the value of all the time series in that time-bin for identifying IP flows responsible for the anomalies (see Chapter 2 for the algorithm details).

6.3 Experimental results

Before analyzing the performance of the system in terms of detected anomalies, in the first three figures we empirically show the reason for which the method is able to improve the "classical" CUSUM methods.

Figure 6.2 shows the temporal evolution of one of the buckets of the sketch over one week, before performing the wavelet analysis. In more detail, it represents the number of bytes received by an IP aggregate and, as can be clearly observed, presents the "typical" seasonality of the network traffic, with a daily and a weekly trend. It is easy to understand that the seasonality in the data can make difficult the detection of some anomalies, which can be masked by such a trend.

Figure 6.7 shows the same traffic aggregate, after the wavelet analysis. As described in the previous section the signal has been reconstructed after that the wavelet coefficients corresponding to the low frequencies (level six) have been put to zero. The result is that the daily as well as the weekly trends of the signal have "disappeared", making intuitively easier the detection of the anomalies.

This intuition is confirmed by the plot of the CUSUM statistics in Figure 6.8, where we can easily see that the statistics related to the original signal (dotted line) is much more "rugged" than the one related to the reconstructed signal (solid line). Hence, considering that (as stated in the theoretical section on CUSUM) an anomaly is revealed if the CUSUM statistics starts increasing and exceeds a given threshold, our method seems to be more robust to signal noise that the "classical" method; the following tables show that this intuition is confirmed by the experimental results.

Since, given the nature of the dataset, we cannot plot a ROC curve,

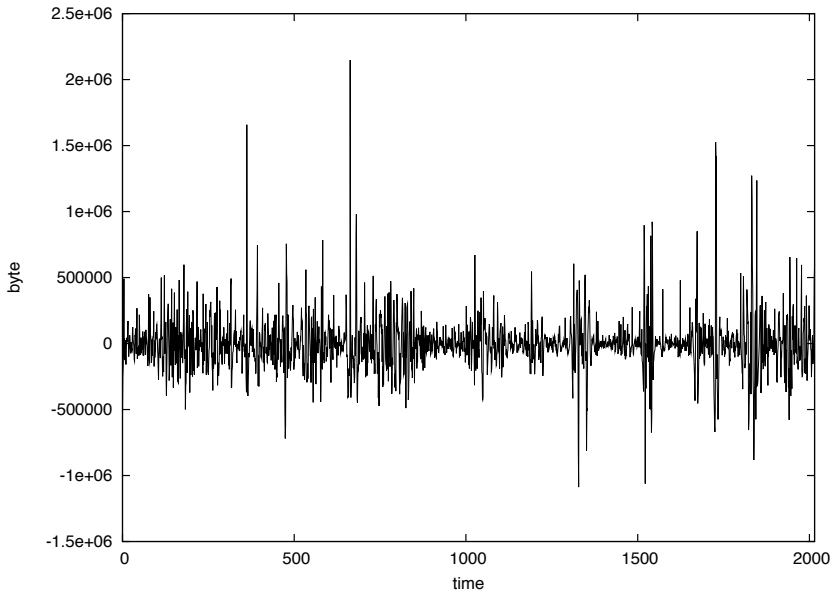


Figure 6.7: *Wave-CUSUM - Reconstructed Signal*

in the following tables we report the total number of detected anomalies and the number of synthetic anomalies detected by the system. Note that the tables have been obtained varying the values of the threshold ξ. The real values of such thresholds (set by Monte-Carlo simulation) are not reported since are not significant in themselves, just consider that the first values (namely ξ_1) always corresponds to the highest threshold value for which the system is able to detect all the 154 synthetic anomalies.

Hence, in general, the system is always able to obtain a 100% detection rate (revealing all the 154 synthetic anomalies), but the performance can be strongly different depending on the total number of

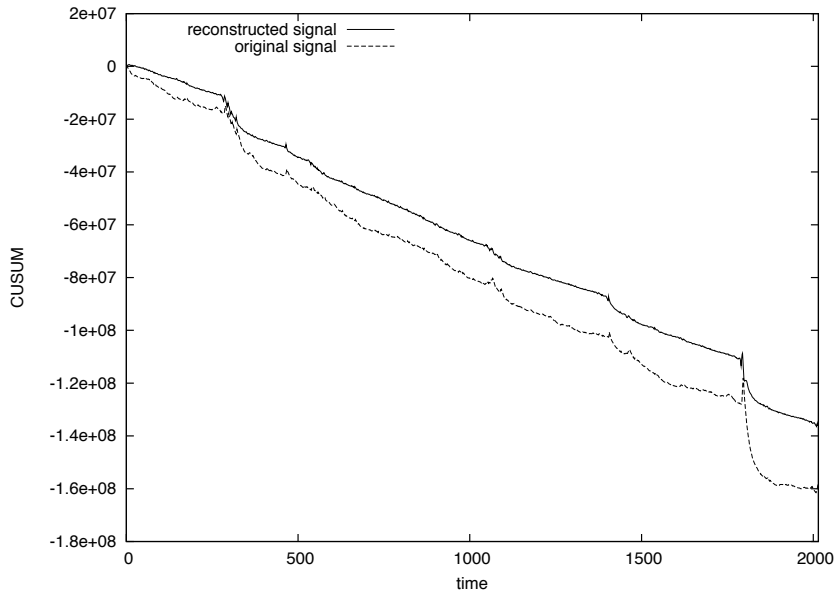


Figure 6.8: *Wave-CUSUM - CUSUM statistics*

detected anomalies that has a direct impact on the number of false alarms.

Slightly differently to what done when testing the systems described in the previous chapters, in this case to assess the validity of the proposed system we have carried out three distinct sets of simulations, in which we have injected, in the original traces, anomalies of low, medium and high intensity/volume. It is worth noticing that, in all the three cases, these anomalies result to be much "smaller" than those used for the previous systems. This is justified by the fact that using the same traces used for the other methods, the behavior of the CUSUM method already takes to very good performance and it is thus

Threshold	Total Anomalies	Synthetic Anomalies
ξ_1	703	154
ξ_2	412	137
ξ_3	348	94
ξ_4	297	67
ξ_5	226	9
ξ_6	189	7
ξ_7	102	6

Table 6.1: *Wave-CUSUM - low volume anomalies*

difficult to demonstrate that the wavelet pre-filtering phase effectively takes to some improvements. But, this becomes much more evident when using anomalies of smaller entity as the one used in this section.

Tables 6.1 and 6.2 respectively present the performance of our system (named Wave-CUSUM) and of the classical CUSUM system, when facing the detection of low volume anomalies. We can easily notice that the performance are strongly different; indeed our system, when detecting all the 154 synthetic, also detects 549 additional anomalies (703 total anomalies minus the synthetic ones), while the "classical" system detects 809 additional anomalies. In this case, to really evaluate the performance of the system, we have performed a manual verification of the dataset, checking the additional detections of the system. From that, we can conclude that, about 150 anomalies of the 549 detected by our system are real anomalies and 150 are suspicious activities. That means that our system has a false positive rate between the 12% and 20%, when the detection rate (over the synthetic

Threshold	Total Anomalies	Synthetic Anomalies
ξ_1	963	154
ξ_2	806	19
ξ_3	778	17
ξ_4	740	16
ξ_5	710	14
ξ_6	623	13
ξ_7	595	12

Table 6.2: *CUSUM - low volume anomalies*

anomalies) is 100%, while for the classical system the false positive rate raises to about the 25%-33%. Note that the "classical" system does not detect all the anomalies already present in the data, while detecting some more "false" anomalies. Moreover we can notice from Table 6.1 that we can obtain a negligible false alarm rate, when the detection rate is about the 89%, while it is not possible to lower the false alarm rate without significantly worsen the detection rate for the classical system.

Tables 6.3 and 6.4 represent the same results seen in the previous tables, when considering medium volume anomalies. Also in this case, we can easily see that the two systems present strongly different performance. Indeed, in this case our system is able to achieve an almost ideal behavior obtaining the 100% of correct detection in correspondence of a negligible false alarm rate, while the "classical" system, when achieving the 100% of detection rate also has a false alarm rate between 22% and 30%. Also we can notice has our system results to

Threshold	Total Anomalies	Synthetic Anomalies
ξ_1	308	154
ξ_2	256	117
ξ_3	205	80
ξ_4	182	75
ξ_5	121	16
ξ_6	80	7
ξ_7	60	6

Table 6.3: *Wave-CUSUM - medium volume anomalies*

Threshold	Total Anomalies	Synthetic Anomalies
ξ_1	751	154
ξ_2	612	23
ξ_3	598	17
ξ_4	571	16
ξ_5	537	15
ξ_6	514	13
ξ_7	491	11

Table 6.4: *CUSUM - medium volume anomalies*

Threshold	Total Anomalies	Synthetic Anomalies
ξ_1	303	154
ξ_1	262	154
ξ_2	127	19
ξ_3	124	17
ξ_4	85	16
ξ_5	39	14
ξ_6	33	13

Table 6.5: *Wave-CUSUM - high volume anomalies*

be robust to small variations in the values of the threshold, while the "classical" method performance abruptly change when varying the threshold.

Finally, Tables 6.5 and 6.6 show the performance of the two systems in the detection of high volume anomalies. Note that in this case, in our system, the value of ξ_1 is not the highest value for which the system detects all the synthetic anomalies, but the highest value for which the system is able to also detect the about 150 anomalies already present in the original data. For these results, the considerations done in the previous two cases are still valid. Indeed our system is able to detect all the anomalies with a negligible false alarm rate, while with the "classical" system we must accept a high false alarm rate, also when not detecting all the anomalies.

Threshold	Total Anomalies	Synthetic Anomalies
ξ_1	554	154
ξ_2	424	27
ξ_3	418	14
ξ_4	382	12
ξ_5	365	10
ξ_6	340	9
ξ_7	317	8

Table 6.6: *CUSUM - high volume anomalies*

Chapter 7

Conclusions

In the last years, the increasing number and variety of security attacks in IP-based network infrastructures have caused growing problems for network operators and users, leading to the need of developing new security architectures.

In this scenario the use of Intrusion Detection Systems has emerged as a key element, since it permits to tackle security threats by masquerader, misfeasor and clandestine users. In order to identify new ad hoc attacks, the development of anomaly based NIDSs assumes a primary role.

In this dissertation we have presented some novel anomaly based NIDSs, which detect anomalies by means of some novel statistical techniques.

We have discussed several statistical approaches, such as Wavelet Analysis, Principal Component Analysis, Heavy change detection, and CUSUM.

In more detail, regarding the Wavelet Analysis we have detailed a novel multi time-scale anomaly detection method, based on a combined use of sketch and wavelet analysis. The use of wavelet transform is justified, despite our approach is original in several aspects,

by the vast literature on the topic, which demonstrates the effectiveness of such analysis in detecting network anomalies. Nevertheless, the application of the wavelets to the single traffic flows results to be unscalable and thus not applicable in modern backbone networks. For this reason we have applied this transform on the top of probabilistic structures, namely the sketches, that allow us to obtain a scalable real-time system, while simultaneously improving the detection rate of classical approaches.

As far as Principal Component Analysis is concerned, we have based our approach on the main idea of using PCA to decompose the traffic variations into their normal and anomalous components, thus revealing an anomaly if the anomalous components exceed an appropriate threshold. It is important to highlight that all we have worked at different time-scales and on different levels of aggregation so as to detect anomalies that could be masked at some aggregation level. Moreover, we have applied the method both to the entropy associated to some given traffic descriptors and to the Kullback-Leibler divergence computed over the histogram of such descriptors.

Regarding the change detection techniques, we have explored both the use of "classical" methods and novel methods. In more detail, we have presented a novel method, based on the idea of discovering heavy changes in the distribution of the heavy hitters in the network traffic. To this aim we have explored the use of several forecasting algorithms for predicting the near-future distribution of the heavy hitters and we have then applied heavy change based methods to the predicted values.

Finally we have discussed the combined use of the wavelet analysis

and the CUSUM algorithm. The CUSUM (CUmulative SUM) algorithm is one of the most promising techniques to detect significant changes in the descriptors of the network traffic. Thus the main idea is to use the variation of the distribution of these descriptors to detect a network anomalies. Nevertheless the time-varying nature of the traffic (e.g., daily and weekly trends) makes somehow difficult to distinguish a network anomaly from a normal variation of the distribution of the traffic. To solve such an issue, in this thesis, we have proposed to combine a classical CUSUM based approach together with the wavelet analysis. In more detail the latter is used to filter out the seasonality from the traffic aggregates so as to improve the performance of the CUSUM based anomaly detection techniques.

For all of these techniques we have discussed the system architecture. Moreover we have shown the performance achieved by the different systems over the Abilene/Internet2 dataset. The performance analysis has highlighted that all the implemented systems obtain, for a proper choice of their parameters, very good performance.

Bibliography

[1] W. Stallings, *Cryptography and Network Security*. USA: Prentice Hall, 4th ed., 2005.

[2] J. McLean, "The specification and modeling of computer security," *Computer*, vol. 23, no. 1, pp. 9–16, 1990.

[3] R. Shirey, "RFC 2828 - Internet security glossary," 2000.

[4] "X.800 security architecture for OSI," 1991.

[5] J. Anderson, "Computer security monitoring and surveillance," tech. rep., James P. Anderson Co., 1980.

[6] D. Denning, "An intrusion detection model," *IEEE Transactions Software Engineering*, vol. SE-13, no. 2, pp. 222–232, 1987.

[7] S. Axelsson, "Research in intrusion-detection systems: A survey," Tech. Rep. 98–17, Chalmers University of Technology, Sweden, 1998.

[8] S. Axelsson, "Intrusion detection systems: A survey and taxonomy," Tech. Rep. 99-15, Chalmers University of Technology, Sweden, 2000.

[9] "The Internet2 Network." http://www.internet2.edu/network/.

[10] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 217–228, 2005.

[11] B. Claise, "Cisco Systems NetFlow Services Export Version 9." RFC 3954 (Informational), Oct. 2004.

[12] "Flow-Tools Home Page." http://code.google.com/p/flow-tools/.

[13] S. Muthukrishnan, "Data streams: algorithms and applications," in *Proceedings of the annual ACM-SIAM symposium on Discrete algorithms*, (Philadelphia, PA, USA), pp. 413–413, Society for Industrial and Applied Mathematics, 2003.

[14] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58 – 75, 2005.

[15] R. Schweller, A. Gupta, E. Parsons, and Y. Chen, "Reversible sketches for efficient and accurate change detection over network data streams," in *Proceedings of the ACM SIGCOMM conference on Internet Measurement*, IMC '04, (New York, NY, USA), pp. 207–212, ACM, 2004.

[16] M. Thorup and Y. Zhang, "Tabulation based 4-universal hashing with applications to second moment estimation," in *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, (Philadelphia, PA, USA), pp. 615–624, Society for Industrial and Applied Mathematics, 2004.

[17] A. Lakhina, M. Crovella, and C. Diot, "Characterization of network-wide anomalies in traffic flows," in *In ACM Internet Measurement Conference*, pp. 201–206, 2004.

[18] A. Lakhina, "Diagnosing network-wide traffic anomalies," in *In ACM SIGCOMM*, pp. 219–230, 2004.

[19] I. Daubechies, *Ten lectures on Wavelets.* no. 61 in CBMS-NSF Series in Applied Mathematics, Philadelphia: SIAM, 1992.

[20] S. Mallat, "Multifrequency channel decompositions of images and wavelet models," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, pp. 2091–2110, Dec. 1989.

[21] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: methods, evaluation, and applications," in *Internet Measurement Comference*, pp. 234–247, 2003.

[22] A. G. Tartakovsky, B. L. Rozovskii, R. B. Blaek, and H. Kim, "Detection of intrusions in information systems by sequential change-point methods," *Statistical Methodology*, vol. 3, no. 3, pp. 252 – 293, 2006.