SAPIENZA
UNIVERSITÀ DI ROMA

SAPIENZA, UNIVERSITÁ DI ROMA

DOTTORATO DI RICERCA IN COMPUTER SCIENCE

XXV CICLO - 2012

ON THE CYBER SECURITY ISSUES OF THE INTERNET INFRASTRUCTURE

DOMENICO VITALI

SAPIENZA
UNIVERSITÀ DI ROMA

SAPIENZA, UNIVERSITÁ DI ROMA

DOTTORATO DI RICERCA IN COMPUTER SCIENCE

XXV CICLO - 2012

DOMENICO VITALI

ON THE CYBER SECURITY ISSUES OF THE INTERNET INFRASTRUCTURE

Thesis Committee

Prof. Luigi V. Mancini (Advisor)
Prof. Giancarlo Bongiovanni
Prof. Riccardo Silvestri

International Reviewers

Prof. Panos Papadimitratos
Prof. Kavé Salamatian

Author's address:
Domenico Vitali
Computer Science Department
Sapienza, Università di Roma
Via Salaria 113, 00198 Rome, Italy
e-mail: vitali@di.uniroma1.it
www: http://sites.google.com/site/domenicovitali/

Abstract

The Internet network has received huge attentions by the research community. At a first glance, the network optimization and scalability issues dominate the efforts of researchers and vendors. Many results have been obtained in the last decades: the Internet's architecture is optimized to be cheap, robust and ubiquitous. In contrast, such a network has never been perfectly secure. During all its evolution, the security threats of the Internet persist as a transversal and endless topic.

Nowadays, the Internet network hosts a multitude of mission critical activities. The electronic voting systems and financial services are carried out through it. Governmental institutions, financial and business organizations depend on the performance and the security of the Internet. This role confers to the Internet network a critical characterization. At the same time, the Internet network is a vector of malicious activities, like Denial of Service attacks; many reports of attacks can be found in both academic outcomes and daily news.

In order to mitigate this wide range of issues, many research efforts have been carried out in the past decades; unfortunately, the complex architecture and the scale of the Internet make hard the evaluation and the adoption of such proposals.

In order to improve the security of the Internet, the research community can benefit from sharing real network data. Unfortunately, privacy and security concerns inhibit the release of these data: its suffices to imagine the big amount of private information (e.g., political preferences or religious belief) it is possible to get while reading the Internet packets exchanged between users and web services.

This scenario motivates my research, and represents the context of this dissertation which contributes to the analysis of the security issues of the Internet infrastructures and describes relevant security proposals.

In particular, the main outcomes described in this dissertation are:

- the definition of a secure routing protocol for the Internet network able to provide cryptographic guarantees against *false route announcement* and *invalid path* attack;
- the definition of a new obfuscation technique that allow the research community to publicly release their *real* network flows with formal guarantees of security and privacy;
- the evidence of a new kind of leakage of sensitive informations obtained hacking the models used by sundry Machine Learning Algorithms.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Internet as critical infrastructure

Over the last decades, the Internet has grown to become a pivot of our society. Thanks to the huge number of "connected" users, Internet is a useful and easy vector able to offer services globally. Internet represents a large media where communications and information pass through and reach the entire world. With the large spread of the mobile devices (e.g., smarthphone, tablets and others), Internet assumes a key role in the communication pattern; the ability of stay connected every-where every-when cannot be satisfied without the technological support of such a network.

This aspect gives Internet a primary role in our community. Nowadays, Internet is the mean for providing a multitude of critical services: example of this is the banking services where users perform economical transactions just clicking on a web page, hedge fund systems or the electronic voting systems which enable the users to express their political preferences. At the same time, Governmental Institutions, financial and business organizations strictly depend on the performance and security of Internet. This aspect in conjunction with the economical and social relevance of the above mentioned services, confers to the Internet a critical characterization.

Internet can be seen as a graph where the nodes are the *Autonomous*

Systems (in this thesis referred as AS) and an edge represents the connection between two AS nodes. The Autonomous Systems are uniquely identified by a the *AS number* and are organized in a hierarchical (multi-tier) fashion. Each AS node manages a collection of connected Internet Protocol (IP) *network prefixes*. The network prefixes as well as the AS numbers are allocated by a Regional Internet Registry (RIR). Globally, Internet numbers are managed by the Internet Assigned Numbers Authority (IANA), an organization operated by the Internet Corporation for Assigned Names and Numbers (ICANN).

Today, Internet is composed by $\sim$400.000 IP prefixes managed by about 40.000 AS nodes. Each AS exchanges raw data (traffic packets) and routing information with other AS nodes. The routing information are all the data needed by the routers to make the right decisions on how to forward the packets. The connection between Autonomous Systems is named *Peering Agreement*: a voluntary interconnection among AS nodes for the purpose of exchanging traffic between the customers of their network prefixes. This interconnections are usually driven by political and economic agreements. A Peering agreement may occur inside an Internet eXchange Point (IXP), namely the physical infrastructure through which Autonomous Systems exchange Internet traffic between their networks. In the Internet architecture, an IXP introduces two benefits: i) it reduces the traffic of a network which must be delivered via upstream providers decreasing the cost of their service, ii) improves routing efficiency and fault-tolerance of the Internet.

The security of the Internet infrastructure is constantly threatened by cyber attacks. In fact, in the last years, new viruses and cyber weapons have been introduced to violate the national infrastructure. *Stuxnet*, [148], and *Flame*, [78], are examples of such cyber attacks.

One of the well known cyber attacks to the Internet infrastructure is the Youtube hijack, reported by *Ripe NCC* in [127], [105] and by the *BBC* in [6]. On Sunday, 24 February 2008, the Pakistan Telecom (uniquely identified by the AS number AS17557) started the announcement of a subprefix of the Youtube's networks. Youtube's upstream provider by just

forwarded the announcement without verifying its validity, which caused a global-scale hijacking of traffic related to YouTube into the Pakistan Telecom autonomous system. Although this event happened in a short time interval, the YouTube's networks returned to be available after about two hours. In the same fashion, in 2010, one of the Data Centers operated by China Telecom announced about 37,000 unique network prefixes as reported in [46]: for a period of 18 minutes the China Telecom Hijacked 15% of the world's Web traffic to servers in China. Further details can be found in [8] and [5]. The report in [46] claims that the incident affected the traffic to and from U.S. government and military web domains (.gov and .mil), including those of the US Senate, the US Army, the US navy, the US Marine Corps, the US Air Force.

Many other examples of cyber attacks can be found in the history of the Internet. In front of this kind of events the research community is discussing the feasibility of making Internet more secure.

These examples show that the security of the services reachable through the Internet is based on the security of the Internet infrastructures.

Most of the security issues of the Internet infrastructure are related to the Internet routing protocol, namely the Border Gateway Protocol (BGP) defined in the RFC 4271, [126]. BGP is a path vector protocol that allows the Internet routers to exchange routing informations; currently, the Internet routing implements the version 4 of the protocol [125], proposed in 1995. The lack of security mechanisms exposes the BGP protocol to a wide range of threats that are constantly undermining security of the Internet.

The security of the BGP protocol has become a critical goal of the Internet management community, since every (erroneous or malicious) misconfiguration of a participating router can lead to severe attacks, e.g. the network disruptions or traffic hijacking [98, 3]. As BGP has been designed with no security features, there are many examples of how easily BGP alterations had a negative impact on the normal Internet traffic such as the China hijack described above. BGP vulnerabilities are due to the lack of a scalable means to verify the authenticity and legitimacy of BGP control traffic. The lacks of security in the BGP protocol has been highlighted

early after its widespread deployment, since no cryptographic authentication of the communications is considered. Announced routes are not verified but rather directly propagated. The lack of security mechanisms leads to forged route announcement propagation, which is probably the major weakness of the BGP protocol.

The security concerns of the Internet network are largely transposed into the research communities. In the last decade, a huge amount of papers and surveys have been written, [143, 19, 73, 154, 16, 67, 149, 147, 110, 154]. A number of cryptographic solutions to prevent BGP attacks have been proposed, [73, 147, 110], but they have not been adopted due to involved operations and considerable overheads. Many studies on the adoption of these solutions have been also provided [74, 138].

The earliest solution to BGP security (i.e., S-BGP [73]) involves digital signature schemes, where AS nodes sign their announcements that can later be verified by other peers. However, digital signatures require a Public Key Infrastructure (PKI) to issue and deliver public key certificates. S-BGP has fostered a number of research efforts to secure BGP [15, 73, 154]. Nevertheless, involved operations, complex trust relations and the difficulty to set up and manage a global PKI hinder the real-world deployment of any security means designed for BGP.

Furthermore, it seems that no secure version of BGP has efficiently addressed the problem of both *route* and *prefix* revocation. With current PKI-based solutions, once an AS has been authorized to forward an IP prefix, there is no means to securely revoke the authorization. In particular, it is possible to revoke the AS certificate so that all its announcements become invalid. However, no proposal provides fine-grained revocation of single announcements.

Observation 1 *The Internet cyber security depends on the security of the underlying routing protocol. The Border Gateway Protocol, namely the state of the art of the Internet Routing Protocol, does not provide suitable security guarantees.*

The distributed nature of the Internet makes research activities a great

challenge, at the same time, the Internet scale makes it hard to apply solutions, evaluates algorithms and proposals.

In order to evaluate the application of new Internet protocols, the research communities can get benefit by the sharing of *real* network data collected by the nodes of the network. In fact, each AS node of Internet maintains a limited and isolated point of view of the entire network. It seems that the amount of data that cross these nodes does not suffice to build a complete and coherent view of the entire Internet Infrastructure. It sounds like the old Indian story of John Godfrey Saxe titled "The blind men and an elephant", [130].

Examples of real data to be shared include network traffic, routing tables and network topology as also stated in [21]. In details, the authors of [21] provide a model for evaluating the adaptability of a secure BGP routing protocol by an ISP point of view; the model uses the exchanged traffic and the Internet topology (in terms of weighted AS-level graph) as parameter to define the *Adaptability value* of a secure BGP protocol.

Furthermore, real network data are a very valuable resource for researchers; for instance, to model the network behavior, to experiment new protocols, and to study many cyber security attacks.

Observation 2 *The sharing of real network data provides a great benefit for the study and the evaluation of new network research proposals.*

Unfortunately, getting accurate details on real AS traffic is impractical since this kind of data are usually confidential, or may contain private informations: security and privacy concerns discourage the publication of such data sets. Mainly, the concerns related with the release of real network data are two: by point of view of the data owner, shared information may reveal sensitive details exploitable by malicious users; on the other hand, there exist legal questions about the users privacy.

Consider a real data set of *network flows* of Internet traffic, namely the list of connections exchanged between two AS nodes. Based on network flows about the Web sites visited by a given individual, an adversary may infer sensitive data, such as political preferences, religious belief, health

status, and more. At the same time, network flows may reveal personal communications among specific individuals, such as the existence of email exchanges, and chat sessions among them. Moreover, network flows can be exploited by adversaries to gather useful informations in planning network attacks, for instance, for identifying possible bottlenecks in the target network in order to increase the impact of Denial of Service attacks.

As a consequence, various research efforts have been carried on to protect privacy of real network data while preserving the practical interest in using the released network flows. Early techniques were based on the substitution of the real IP addresses with pseudo-IDs, for instance in *Crypto-PAn* [52]. However, it has been shown that this technique is insufficient, since an adversary may reconstruct the real IPs based on the values of other fields of the flows [76], exploiting his knowledge of the characteristics of network hosts (*fingerprinting* attacks), or by injecting peculiar flows in the monitored network (*injection* attacks). For this reason, more sophisticated techniques have been proposed, based on the perturbation of other data in the flows, e.g., [14, 115, 134, 54]. However, it seems that the techniques proposed do not provide any formal guarantee of protections, and it has been recently shown that they are prone to different kinds of attacks [18].

In addition consider the topology of Internet, namely the physical and logical interconnection between AS nodes (usually named "peering agreements"). These agreements are usually driven by political and economic agreements, hence, they are confidentials and reserved since them can be maliciously exploited by a competitors.

Furthermore, the sharing of real network data can also bring benefits to other research fields. The *Denial of Service attack detection* is another relevant research field which can get benefit of the sharing of real network data. The Denial of Service attacks is one of the most effective malicious activities achievable in a network. In the last years, thousand of Distributed Denial of Service attacks (DDoS) have been registered against Governmental Institutions and companies [84, 124]. The insight behind these attacks is quite simple: the high number of connections, or high

rate of session-work required by the adversary, fill the resources (e.g storage, bandwidth or CPU) of the targeted appliance. Automatic tools, freely available in the Internet, help attacker in performing these malicious actions. Such easy-to-use interface makes these activities exploitable by all Internet users: political activists (aka "hacktivists"), individuals or groups of interest keep increasing the use of these tools to express their own disagreement against private companies or political initiative (i.e., websites of Public Administrations, mail servers etc.). The Cisco security annual report, [28], documents this.

Many network administration tools, such as DoS detection algorithms and Traffic Classifiers, can be improved if companies, Institutions and researchers could share the real data with a high degree of security, privacy and precision. Even if the sharing of network data can help the research communities, it's important to notice how the lack of such sharing can cause unexpected results.

In fact, many traffic classification engine, as well as DoS detection algorithms, have been proposed in the past, [99, 48, 49, 7, 106, 50, 2, 152]; each of them proposes new techniques, implements different algorithms and performs good results. Usually, the motivation for such results is twofold: i) the researcher performs algorithms evaluations using synthetic data, namely, traffic logs artificially generated or ii) the training data sets used to evaluate algorithms are not so representative. Consider the Denial of Service Detection algorithms, the literature abounds of conflicting proposals. Many papers propose the use of sundry metrics borrowed by the Information Theory field, [85, 132, 86, 87, 129], as the best solution, at the same time, other results claim that statistical approaches work better, [150, 37, 53, 112, 136]. Equally, similar considerations can be done for Intrusion Detection Systems [22, 113, 75, 103, 26, 114].

The relevance of shared real network data is also documented by the huge amount of research and paper that survey the anonimization and privacy issues, or by the number of projects which involve the use of real data. The DARPA Intrusion Detection Data Sets [1] provided by the Lincoln

[1] `http://www.ll.mit.edu/`

Laboratory, Massachusetts Institute of Technology, is a valid example of this phenomenon. The Lincoln Laboratory published dataset of real network data in which several attacks have been recorder. These data sets were largely used by the research community as support of Intrusion Detection Systems.

Observation 3 *The security and privacy concerns inhibit the release of real network data. No previous proposal of network data anonymization provides formal guarantee about security and privacy.*

Note that, the real network data can be released in a *direct* or *indirect* fashion. The privacy and security concerns are not only related to the real network data directly released. For example, real network data can be "synthesized" in mathematical models, used by the Machine Learning Algorithms. Usually, the Machine Learning (ML) systems are trained to perform a variety of complex tasks. Machine Learning systems learn how to recognize patterns (training phase), make unexpected decisions, or react to a dynamic environment (classification phase). During the learning phase, the relationships and the correlations implied in the training samples (e.g., real network data) are gathered inside the *model*. Afterwards, the model is used during the classification phase to classify and evaluate new data. These trained models may leak sensitive information as discussed in chapter 6.
The models used in Machine Learning algorithms are also subject of many privacy and security attacks. In the lasts years, two techniques have been developed to mitigate these issues: the Privacy Preserving Data Mining, [142], and the Differential Privacy, [44]. The Privacy Preserving Data mining is a prolific research area in which many techniques able to preserve the privacy of the single records are proposed. On the other hand, in 2004, Cynthia Dwork proposed Differential Privacy, namely a technique able to maximize the accuracy of queries from statistical databases while minimizing the identification of its records.

The Machine Learning algorithms are largely used for many networking operations, [48, 49, 7, 77, 106, 17], security tasks, [116, 131, 75, 104, 103,

66, 26, 114], medical activities, [146, 97, 135, 88, 63] and more [70, 140].

The outcomes of my research focus on the issues described in this scenario and contribute in several aspects of the cyber security issues of the Internet Infrastructure.

1.1.1 Research questions

There are many security aspects that require a deep analysis and more efforts by researchers and vendors. In particular, this dissertation addresses the following research questions:

Question 1 *How can the Internet community get a secure Internet Routing protocol? Can this solution be easily applicable on the Internet?*

Question 2 *How can a network administrator detect a Denial of Service attacks? Are the most used metrics really reliable?*

Question 3 *Can companies, Institutions and research communities safely share the real network data of their network while preserving the security of their network and the privacy of their users?*

Question 4 *Can companies, Institutions and research communities safely share a Machine Learning Classifier with others, without any leakage of sensitive informations? For example, can Institutions and research communities safely share a DoS detection system or a Internet Traffic Classifier learned with the real network data of their network?*

1.1.2 Contributions

The outcomes proposed in this dissertation, provide the following contributions:

Contribution 1 (Internet Routing Security [94]) Lack of security mechanisms expose the Border Gateway Protocol (BGP) to a wide range of threats that are constantly undermining security of the Internet. Chapter 3

introduces the protocol reBGP, [94], a secure BGP protocol that leverages state of the art cryptographic primitives to mitigate prefix hijacking and invalid path attacks as well as to manage revocation and reassignment of IP prefixes. Protocol reBGP uses Identity-Based Cryptography to eliminate the requirement of a Public Key Infrastructure and aggregated signatures to reduce the communication overhead.

In reBGP, a trusted authority (e.g., the ICANN) assigns an IP prefixes to an AS issuing a prefix endorsement that certifies the AS as the originator for that prefix. Similarly, the AS nodes sign their announcements within a route endorsement that authorize their peers to propagate their routes. This leads to a "chain of trust" that guarantees the validity of paths in the network.

Protocol reBGP enjoys constant time route verification, so that the overhead for the added security is minimal and balanced between the routers. Furthermore, reBGP introduces a mechanism for route endorsement revocation, allowing any AS to revoke previous endorsements, simply issuing new ones. The new endorsement becomes immediately valid and automatically revoke the old ones. The same approach is used by the ICANN to revoke prefix endorsements, in order to reassign an IP prefix to a new originator AS. Finally, protocol reBGP has been extensively simulated in order to validate its correctness and to evaluate the introduced overheads in terms of computations and communications.

Contribution 2 (Denial of Service Detection [43]) The Denial of Service Attacks is a malicious activity aimed to negate a service or a resource to a legitimate users; in literature, many results about this threat can be found, since it represents a macro-category of attacks achievable in many context (operating system, network, application based etc etc) and it does not require an high skill by the users.

Starting from the huge data set of *real* network data, this dissertation contributes on the topic with (i) a validation the theoretical research results, namely applying the most used information theory metrics; (ii) the analysis of the proposed metrics using a lightweight dump data set (i.e.,

using the network data format provided by CISCO^{TM} netflow protocol), and (iii) reporting the ability to perform (D)DoS detection by the analysis of "aggregated" network traffic data [43].

Furthermore, this thesis compares the different metrics and evaluate their effectiveness against several network activities (like (D)DoS attacks and nightly scheduled maintenance jobs), in terms of anomaly detection and robustness.

Contribution 3 (Obfuscation of sensitive data in network Flows [41]) Real world network traffic traces represent a gold mine for different research activities. Existing techniques proposed to sanitize network flows do not provide any formal guarantees and, on the other side, many anonymization procedure does not preserve statistical and semantics property of original data set, reducing data utility.

This dissertation provides a novel obfuscation technique, named (k, j)-obfuscation, [41], for network flows that provides formal guarantees under realistic assumptions about the adversary's knowledge. The (k, j)-obfuscation technique is supported by extensive experiments with a large set of real network flows collected at an important Italian Tier II Autonomous System, hosting sensitive government and corporate sites (section 2.5 provides a detailed description of the network data sources.).

Contribution 4 (Security risk on sharing ML Classifiers [56]) The sharing of data between research communities is strongly significant to achieve good results. Whether we speak about Machine Learning Classifier, many countermeasures have been proposed. In this context, the results proposed in this dissertation provide evidence that new kinds of attack exists, and that previous privacy preserving techniques do not provide any formal or practical guarantee.

A new information leakage, that it has not been considered before, is the outcomes presented in chapter 6. The contribution in this context shows that it is unsafe to release trained classifiers since valuable information about the training set can be extracted from them. Results show

a new attack strategy: a meta-classifier can be trained to extract meaningful data from targeted classifiers. Furthermore, chapter 6 describes several attacks against existing ML classifiers, in particular, it presents attack against an Internet traffic classifier implemented via Support Vector Machines (SVMs) and a speech recognition software based on Hidden Markov Models (HMMs).

During my research, in addition to [41, 43, 56, 94], the following publications have also been presented:

- D.Vitali, A. Spognardi, L.V. Mancini, *Replication schemes in Unattended Sensor Networks*, proceedings of the 4th IFIP International Conference on New Technologies, Mobility and Security (NTMS), February 2011, Paris, France (DOI 10.1109/NTMS.2011.5721047).

- D. Vitali, A. Spognardi, L. V. Mancini and A. Villani, *MhRep: Multihop Replication Scheme for Data Survival in Unattended Wireless Sensor Networks*, proceedings of the IEEE Symposium on Reliable Distributed Systems Workshops (SRDSW), October 2011, Madrid, Spain (DOI 10.1109/SRDSW.2011.15).

1.1.3 Thesis overview

Chapter 2 surveys the related works of the issues debated in this dissertation.

Chapter 3 introduces *reBGP*, a new secure version of the Border Gateway Protocol, reBGP, which provides cryptographic guarantees against *invalid path* and *false route* announcements attacks. Furthermore, reBGP, does not require the deployment of Public Key Infrastructure, hence, it can be easily set to secure the Internet network.

Chapter 4 investigates the effectiveness of sundry Denial of Services detection strategies based on several Information theory's metrics using a huge and *real* dataset of CISCO^{TM} network flows.

Chapter 5 presents a new obfuscation technique for network flows able to provide formal guarantees of work under realistic assumptions on the power of the adversary.

Chapter 6 introduces a new attack to the Machine Learning models aiming to extract meaningful information which cannot be protected by the previous privacy-aware techniques (e.g., privacy preserving data mining or differential privacy)

Chapter 7 concludes this dissertation and proposes future works.

Chapter 2

State of the Art

This chapter describes the state of the art of the topics addressed by this dissertation.

2.1 Internet Routing Security

This section introduces the state of the art of the proposal aimed to provide a secure Internet routing protocol.

Main security issues of the BGP protocol concern the *control plane*, i.e., the way BGP speakers build the routes of the Internet. Attacks to the control plane have direct implications on the *data plane*, that is how the traffic is routed [58, 149]. This section reviews prominent works related to BGP protocol and refer the reader to [19] for a comprehensive review of the weaknesses of BGP.

Nowadays, the most debated solution to secure BGP is S-BGP, proposed by Kent et al. in [73]. Their protocol provides a range of security features for BGP but suffers from high management cost. In particular, S-BGP assumes the deployment of two Public Key Infrastructures (PKI) to authenticate UPDATE messages and the Internet resources (namely, AS numbers and the association between IP prefixes and AS number).

Other PKI-based proposals are soBGP, [147], and psBGP, [110]. Protocol soBGP does not target AS-path authentication but only guarantees

the authenticity of announcements made by IP prefix owners. Protocol psBGP, [110], proposes to replace one of the PKIs required by S-BGP with a reputation scheme. In particular, psBGP uses a centralized PKI for authenticating AS numbers, and a decentralized trust model for verifying the correctness of IP prefix origination. Goldber et al., [59], provides a quantitative comparison among the above BGP security proposals (S-BGP and soBGP) and shows their weaknesses against several attacks.

SPV, [67], leverages Merkle Hash Trees to reduce the use of a central PKI and one-time signatures to protect BGP against path forgeries. SPV is a complex protocol that requires the management and the exchange of a significant amount of state informations. Security flaws of SPV have been exposed in [123]; the same work also suggests how to fix the protocol.

Zhao et al. [158] proposes to reduce the overhead introduced by S-BGP, using aggregate signatures to sign/verify AS-paths in the UPDATEs. In [158], multiple messages are combined together to obtain a Merkle Hash Tree, so to sign only the root of the tree. However, the deployment of a PKI like the one assumed in S-BGP [73] is still required.

The authors of [15] present an aggregate signature scheme that allows for "lazy verification". In this scheme, signers can add their signature to the signed-so-far message and delay its verification at a later time. However, the scheme requires a per-singer random string that makes the signature length dependent on the number of singers.

A symmetric-key based approach is proposed in [16]. The authors claim that their scheme affords UPDATE message authentication as long as at least one router on the path is honest. Message complexity is linear in the number of signer and verification cost is logarithmic in the number of singers.

The idea of using Identity-Based Cryptography to avoid PKI in secure BGP routing has been suggested by others papers (e.g., [57] and [10]) but, it seems that there is no implementation nor evaluation. Notably, the authors of [67] highlight the problem of revocation in Identity-Based settings, one of the issues that protocol reBGP addresses (Chapter 3). The only BGP proposal that leverages IB cryptography and has been evaluated, relies

on Trusted Platform Modules (TPM), and is presented in [90]. Its authors claim that AS-path authenticity can be guaranteed by checking only the announcements of neighboring routers, as long as cryptographic keys and routing policy management are enforced by a TPM running on each router. In other words, the authors assume that signing key are sealed in the TPM and routers cannot obtain a signature over a route that does not comply with the routing policy. As a result, the security of the schema heavily depends on the security of the TPM. Moreover, the authors do not discuss how TPM keys should be revoked/updated when IP prefixes are moved from one AS node to another.

2.2 Denial of Service attack detection strategies

This section overviews some results about the strategies for detection of Denial of Service cyber attacks. As shown in the following, conflicting results are the effect of the lack of the sharing of *real* network data.

Detection and mitigation of (D)DoS attacks is still an open challenge [38, 34]. A systematic analysis of DDoS attacks is presented by [101], where the authors define a complete taxonomy of attacks, proposing different criteria such as Exploited Weaknesses, Degree of Automation, Exploited Weakness to Deny Service, Source Address Validity, Possibility of Characterization, Dynamics, Persistence of Agent Set, Victim Type or Impact on the Victim. Many results, like [53] and [112], agree upon the use of Entropy and Relative Entropy (*information divergence*) as effective metrics for anomaly detection (an introduction to those metrics will follow in Section 4.2). In [53], the authors analyze several genuine network full traces, using blocks of 1000 consecutive packets to compute entropy and frequency-sorted distribution of selected packet attributes. The full network traces used include the headers and the payloads of the packets. Since the network traces are not known to contain malicious activities, the authors overlay synthetic DDoS attacks at various degree of concentrations. An attack alarm is raised if the computed entropy value overcomes a thresh-

old, pre-determined from empirical analysis.

Authors in [112], show that the use of entropy of the IP addresses distribution using fixed-dimension blocks of packets can be very time consuming even for small organizations and imposes a CPU-burning process for big bandwidth network edge. Moreover, they introduce a dynamic definition of alarm threshold in order to mitigate entropy fluctuations, based on its standard deviation. Others works, like [107] and [129], improve attack detection using other information theory metrics, like cumulative entropy. Other entropy based metrics are based on the concepts of *information divergence*, such as Rényi, [86], and Kullback-Leibler divergence, [85]. Their main advantages are the ability to improve the anomaly detection, providing at the same time earlier responses and low false positive rate [150]. The previous entropy based approaches, indeed, experience many false positives in case of fluctuations of traffic pattern. Section 4.3 of this dissertation addresses these aspects.

Common issues of related works is the consistency and the nature of used data sets. Almost all the papers refer to datasets that are historically consolidated (like the DARPA dataset) or that have been collected from restricted and unrepresentative traffic, [53, 80]. DARPA dataset has been created by the Information Systems Technology Group (IST) of MIT Lincoln Laboratory, under Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL/SNHS) sponsorship. The purpose of this dataset is to collect and distribute the first standard corpora for evaluation of computer network Intrusion Detection Systems (IDS), [38, 39]. Although its nature, MIT datasets have been used by researchers to evaluate malicious activities, like DoS or DDoS in sundry researches, [85, 112, 107, 129]. The DARPA data sets are organized based on the DDoS attacking software leading that these attacks have the simplicity of structure and type in spite of the complexity of the real data. As stated in [68], the methodology used to generate the data by MIT Lincoln Laboratory and the nature of data itself are not appropriate for simulating different, non academic, network environments. So the experimental results of many works suffer of the above limitation. In general, many pro-

posed works use synthetic traffic in combination with attack-free network traces, like [86] and [150]. Such a methodology can limit the validity of the research results and lack of generality. In fact, malicious activities artificially injected can miss relevant features that real activities exhibit: this lacking may produce unrealistic behavior and, in the worst case, unreal or unexpected results. The work in [150] is an example of such limitation: in order to enrich the simulations dataset, authors generate synthetic traffic and attack traffic using, respectively, Gaussian and Poisson distributions.

Furthermore, the use of DARPA-2000 or other similar traces impose a huge computation effort and technical limit. In fact, these dataset includes *tcpdump* logs (TCP *header* and *payload*) and so require a non-negligible effort by network equipments and large computations efforts.

An alternative to full packet traces are $CISCO^{TM}$ Netflow, [139]. In fact, $CISCO^{TM}$ Netflow provides a lightweight picture of the network since it does not include the payload informations. One recent study that uses $CISCO^{TM}$ Netflow technology to perform DDoS detection is [132], that proposes a multi-layer approach that combines several steps on sampled netflows. Again, in [132], synthetic attacks were introduced in the real traffic gathered from a tier I ISP, in order to simulate DDoS attack.

Today, huge and real data sets are needed to analyze the emergent types of coordinated and DDoS attacks, made by volunteer users, as opposite to the early crackers with bad intention. A recent analysis of high tech cyber threats to national critical infrastructures, [27], introduces the concept of "hacktivism", to emphasize the new user role. The authors report some real cases where citizens were involved to disrupt national infrastructures, "carrying out politically-motivated hacking and bringing down Government agencies' website". The recent *Operation Payback* is actually a proof of this statement. Again, in the case of historically consolidated datasets [112], it is easy to notice that they are too old to represent recent (D)DoS attack under the hacktivists or zombies.

Instead, the analysis and results presented in Section 4.3.1, are based on completely *real* network data with the *known* of *real* traffic anomalies.

2.3 Obfuscation strategies

This section provides an overview on the obfuscation strategies previously presented in literature. Furthermore, this section presents the attacks that can be performed on anonymized network data sets.

Early techniques for network flows obfuscation were based on the encryption of source and destination IP addresses. However, those techniques proved to be ineffective, since an adversary might be able to re-identify message source and destination by other values in a network flow, or in a sequence of flows (see, e.g., [14, 153, 13, 32]).

King et al. in [76] proposes an extensive taxonomy of attacks against network flow sanitization methods; techniques fall into two main categories:

- *Fingerprinting:* re-identification is performed by matching flows fields' values to the characteristics of the target environment (such as knowledge of network topology and its settings, types of OS and services of target hosts, etc). Typical re-identifying values for network flows are: Type of Service (tos), TCP Flags, timestamps, number of bytes, and number of packets per flow.

- *Injection:* the adversary injects a sequence of flows in the network to be logged, that are easily recognized due to their specific characteristics; e.g., marked with uncommon TCP flags, or following particular patterns. In order to perform this attack, the adversary must know in advance in which network the flows will be collected, and during which time period.

Additional techniques can be used to exploit the results of the above attacks to decrypt IP addresses of new network flows. In particular, if the IP address encryption is performed with the same key across the whole set of flows (as in most existing defense techniques), and the adversary discovers an IP mapping in one flow, he can decrypt the same IP address in any other flow.

2.3.1 Defenses tailored to network flows

Several efforts have been devoted to the implementation of frameworks (e.g., [134]) or configurable tools (e.g., [54]) through which the network administrator can define ad-hoc and per-field obfuscation policies.

Roughly speaking, defenses that can be found in the literature (e.g., [14, 115] among many others) take a "reactionary" approach: typically, in those works, a new kind of attack is identified, and a defense technique is proposed for that attack, which is generally based on the permutation or generalization of some fields' values. However, proposed techniques do not provide a general solution. Indeed, as theoretically proved by Brekne and Årnes in [13], and empirically shown by Burkhart et al. in [18], those techniques can be easily defeated by the injection of flows following complex patterns over sufficiently long periods of time.

As a case study, it can be considered the well-known *Crypto-PAn* [52] technique, which is currently incorporated within several network flow collector tools. Crypto-PAn is a sanitization tool for network flows that encrypts IP addresses in a prefix-preserving manner. Crypto-PAn has the following properties: *(a)* it performs a *one-to-one* mapping from original IP addresses to anonymized IP addresses, based on AES encryption; *(b)* IP address encryption is *prefix-preserving*; and *(c)* it is *consistent across the whole set of released traces*.

A malicious user, which acts inside the monitored network, can inject bogus and easily detectable flows (i.e., marked with uncommon TCP Flags that match distinguishing statistical or behavioral patterns) in order to understand how one IP address is mapped to its encrypted value inside the obfuscated flow set.

The defense technique described in section 5.1 adopts cryptographic primitives to hide real IP addresses (Pseudo Random Number Generator), and obfuscation of flow fields' values. However, differently from previous works, it provides strong confidentiality protection even when the adversary can reconstruct the mapping between an IP address and its encrypted value, possibly as a result of injection attacks.

2.3.2 Microdata anonymization techniques

Techniques proposed in the database area for microdata anonymization have the advantage of providing formal privacy guarantees, under specific assumptions. Hence, it is natural to investigate the application of these techniques to network flows. At first glance, network flow logs seem very similar in nature to any other record sets stored in a relational database (census data, medical records, etc.). However, as explained below, those techniques are unfeasible to network flows, due to the peculiar characteristics of these data.

The simplest microdata anonymity principle is k-anonymity [128], which consists in making any record indistinguishable in a group of at least k records based on *Quasi Identifier (QI)* values; i.e., values that, joined with external information, may reduce the candidate set of record respondents. Any group of records having the same values for QI attributes is called a *QI-group*. The main criticism found in the literature about the application of this principle to network flow logs (see, e.g., [31]) regards loss of information: indeed, since many fields of network flows may act as QI, data quality would be degraded to an unacceptable extent. The same argument holds for more sophisticated privacy principles that guarantee not only anonymity but also sensitive value diversity, such as l-diversity [95] and t-closeness [89].

However, it can be observed that the above mentioned principles are not even applicable to the anonymization of network flows. Indeed, if the private value of each individual does not change in released microdata (this is the case of network flow logs, if IP address encryption is consistent across the whole set of flows), works in [128, 95, 89] are effective only under the assumption that each individual is the respondent of at most one record in the released microdata. Indeed, if the adversary knows that the same individual (in this case the IP address I) is the respondent of one tuple in more than one QI-group, an adversary may be able to derive the confidential information (the encryption of I) by simply intersecting the private values of tuples in those QI-groups.

Since the same IP address typically appears in multiple network flows, an appropriate privacy principle to be considered is m-invariance [151], which has been proposed to enforce both anonymity and diversity for incremental release of microdata. This principle ensures that *(i)* all the QI-groups in which an individual's records appear have the same set of private values, and *(ii)* each QI-group does not contain records having the same private value. However, the application of m-invariance to network flow logs is unfeasible, since the cardinality of IP addresses is very large; this would result in a very coarse generalization of QI-values, and in the introduction of a large number of counterfeit flows to enforce property *(i)*.

In order to overcome the above problems, the (k, j)-obfuscation technique introduced in section 5.1, adopt a many-to-one mapping among IP addresses and encrypted values; this mapping is consistent across the whole set of network flows.

In general, the fact that two specific hosts A and B exchanged some messages may be considered confidential information. Hence, since an IP address uniquely identifies its host, the (k, j)-obfuscation technique assumes that confidential information in a network flow is the set of attributes $\{src\_addr, dst\_addr\}$. Indeed, if removing those fields from network logs, no confidentiality violation could reasonably be perpetrated. Unfortunately, removal of IP addresses from logs would completely disrupt the utility of the data. Note that, from the point of view of privacy preservation, it is not relevant distinguishing between source and destination IP, since in many cases a flow from A to B (request) implies the existence of a flow from B to A (response). When joined with external information, fields in the network flow other than IP addresses may restrict the candidate set of source and/or destination hosts. For instance, as shown in [153], based on network flow data such as packet and byte counts, it is possible to identify the Web server originating the request.

Observation 4 *A network flow is obfuscated if it cannot be associated with high confidence to its source and destination IP addresses.*

2.4 Leakage of Machine Learning models

This section describes the security and privacy concerns related to the *real* data indirectly released. In fact, as introduced in section 1.1, security and privacy concerns are not only related to raw network data, but may affects mathematical models used in Machine Learning algorithms.

In the past, many results on this topic have been published. Although the scope of the results reported in section 6 is strictly related to information leakages issues, this section briefly describes also relevant results in terms of privacy concerns, i.e., Privacy Preserving Data Mining (PPDM), [142] and Differential Privacy, [44]. It is worth describing some of these related results, even though this thesis considers a the type of leakage which has not been considered before.

As formalized by Dwork in [44], *differential privacy* deals with the general problem of privacy preserving analysis of data. More formally, *a randomized mechanism M provides $\epsilon-$differential privacy if, for a database D_1 and D_2, which differ by at most one element, and for any t:*

$$\frac{Pr[M(D_1) = t]}{Pr[M(D_2) = t]} \leq e^{\epsilon}$$

In the differential privacy model, a trusted server holds a database with sensitive information. Answers to queries are perturbed by the addition of random noise generated according to a random distribution (usually a *Laplace* distribution). Two settings are defined: *non interactive*, where the trusted server computes and publishes statistics on the original data, and *interactive*, where the server sits in the middle and directly alters the answers to user queries to guarantee specific privacy properties.

Chaudhuri et al. [24] design a privacy preserving logistic regression algorithm which works in the $\epsilon-$differential privacy model [45]. The idea is quite simple: the result of the trained classifier is perturbed with a dynamic amount of noise. This approach does not consider the security issues due to the exposure of the model generated during the learning phase of the linear regression algorithm.

Other machine learning algorithms, such as Decision Trees, Artificial Neural Networks, Clustering, have been re-engineered to provide differential privacy and several of them are defined within the SulQ framework [9].

Privacy Preserving Data Mining, [1], is a research area aimed at developing techniques that perform data mining primitives while protecting the privacy of individual data records. In [142], Verykios et al. classified PPDM techniques in five classes. Among them, the *Privacy preservation* class refers to techniques used to preserve privacy for *selective* modifications of data records. This can be achieved through heuristic values (e.g., selecting the values that minimize the utility loss of the data), cryptographic protocols (e.g., via Secure Multiparty Computation [93]), or reconstruction-based techniques (e.g., strategy aimed at reconstructing the original data distribution using randomized data).

2.5 Monitored network

The research activities I worked on were funded by the European project ExtrABIRE, [30]. The objective of the project as *to evaluate the overall resiliency of the Internet infrastructure of a Member State and, more generally, to assess the impacts of a coordinated cyber-attack on its Internet infrastructure.The final aim of the Project is to develop a national Internet contingency plan that will include the identification of processes, procedures, organizational issues and technical countermeasures that Member States and private organizations should adopt and implement to mitigate threats to their Internet connectivity*[1].

One of the partners of the ExtrABIRE project is the CASPUR consortium (Consorzio interuniversitario per le Applicazioni di Supercalcolo Per Università e Ricerca,[2]). The collaboration with the CASPUR allowed me the ability of analyze a big amount of *real* data. In fact, during these activities, an important AS node of the Internet infrastructure has been moni-

[1] With the respect of the Non-Disclosure-Agreement of the *ExTrABIRE* project, no detailed information about AS (such as AS name or number) nor ISP interconnections will be provided to preserve AS and host privacy.

[2] `http://www.caspur.it`

tored. During this research activities, a probe of network flows collected more than 2 years of Cisco^{TM} netflows: these data are essential for the results proposed in this dissertation.

The monitored network is represented in Figure 2.1. The network probe has been installed within the $AS3$. The $AS3$ reaches the Internet through a Tier2 and a Tier1 AS node and, at the same time, it is the Upstream provider for $AS1$, acting as backup link. Furthermore, the $AS3$ provides sundry services, e.g. web hosting, mail servers and end-user X-DSL connections. During the collection phase, the $AS1$ was target of many attacks and malicious activities and these data have been used to verify the effectiveness of the DoS detection metrics (chapter 4).

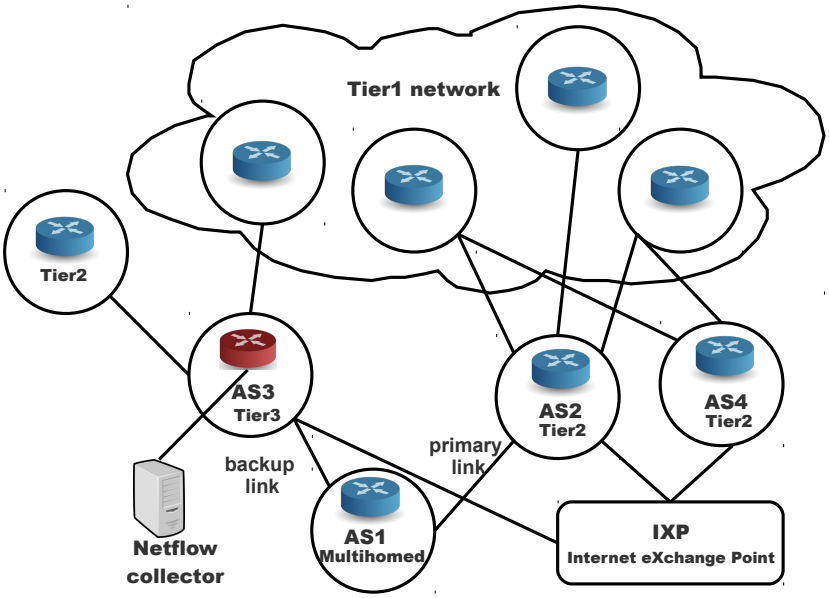


Figure 2.1: Monitored network architecture.

Since $AS3$ is composed by heterogeneous networks and services, it can be considered as a good testing case for the research activity proposed in this dissertation; furthermore, it is general enough to represent many real contexts. To show the dimension of the monitored AS node, $AS3$, table 2.1 reports the average of the exchanged traffic. Considering that it

manages thousands of unique IP addresses and that the amount of packets is around $30 Kbits/seconds$, the monitored AS node can be considered of medium size for an European member state.

Time of the day	Flows/s	Packets/s	Mbit/s
00:00AM-11:00AM	377/312	8.8K/6.4K	37/44
11:00AM-06:00PM	1.3Kb/930	21Kb/13K	113/54
06:00PM-11:59PM	764/575	14Kb/8K	80/27

Table 2.1: Monitored network: bandwidth and traffic characterization (input/output).

Netflows dataset

The *Netflow* protocol is a Cisco^{TM} technology used for monitoring IP traffic [139]. Despite the classical packet collector (packet dump), Netflow only collects data in Layers 2-4.

Netflow efficiently monitors a network, enabling services like traffic accounting, usage-based network billing, network planning, as well as Denial of Services monitoring. Netflow records are extremely compacts and representative, avoiding to maintain the packet's payload and making analysis and computation lighter. While several kind of attacks crafted in the traffic payload are able to circumvent the detection filter, traffic anomalies are still effectively observable. Netflow is recognized as a network monitoring tool: many research papers as well as professional software use Netflow protocol as source of data to query network status or get back data logs.

The typical configuration to leverage the Netflow protocol is made by a router with netflow capabilities (*netflow exporter*) and a probe (*netflow collector*) able to store received data (see Figure 2.1). Netflow records are sent as a *UDP* stream of bytes. A netflow-enabled router creates one record with only selected fields from the TCP headers of *each* transiting connection (Figure 2.2): a single netflow record is a *unidirectional* sequence of packets all sharing the 7 values source and destination IP addresses,

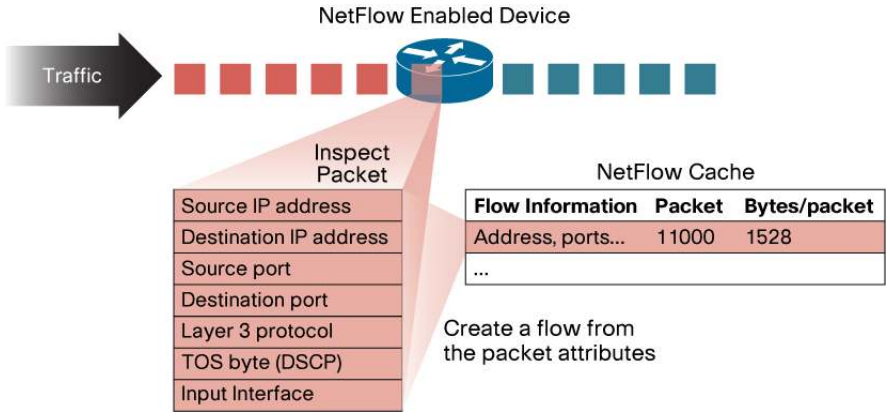


Figure 2.2: How a Netflow enabled device exports netflow records. This image is taken from Introduction to Cisco^{TM} IOS Netflow.

source and destination ports (for UDP or TCP, 0 for other protocols), IP protocol, Ingress interface (SNMP ifIndex) and IP Type of Service. Other valuable informations associated to the flow, like timestamps, duration, number of packets and transmitted bytes, are also recorded. A single flow is a record that represents the data exchanged between two hosts only in one direction, since it aggregates all the IP packets that composed a single communication session. Indeed, a single TCP connection is represented by two distinct flows in opposite directions, despite the number of IP packets or the number of exchanged bytes.

A netflow-enabled router sends to the probe a single flow as soon as the relative connection expires. This can happen when *(i)* when TCP connection reaches the end of the byte stream (FIN flag or RST flag are set); *(ii)* when a flow is idle for a specific timeout; *(iii)* if a connection exceeds long live terms (30 minutes by default).

The use of Netflow technology has several advantages with respect to the raw packet sniffing, since using just few information it is able to give a lightweight picture of monitored network. Several researches proposed netflow collectors as IDSs (Intrusion Detection Systems), [22], traffic classifiers, [42], and others security tools. To have a flavor of the advantages of a netflow dataset in terms of dimension and required effort, the following statistics can be considered: a 2 G Bytes of full netflow entries contains 110

millions of flows, 2 billions of packets and about 1,5 T Byte of exchanged data, corresponding to the data gathered in one single day. Until today, the dataset used in this dissertation consists in a collection of 48 months long netflow records (about 4 T Bytes of data) and it keeps growing.

Note that, despite the advantage of low computational requirements to process an extensive amount of data, netflows inevitably sacrifice a lot of valuable information related to traffic payload: carried attacks to single host services, virus or malware specific signatures, particular malformed packets and so on, are dropped and cannot be recovered from netflows. In this configuration, due to some privacy issues, it is not possible to access to the whole payload to perform a full packet inspection. In fact, the monitored network contains several governmental networks that exchange sensitive data and any access to their packet payloads is restricted.

Chapter 3

Relieve Internet Routing security of Public Key Infrastructure

This chapter introduces the protocol reBGP, a secure BGP protocol that mitigate prefix hijacking and invalid path attacks.

3.1 BGP fundamentals

The Border Gateway Protocol (BGP) is the IETF standard routing protocol on the Internet. It is a path vector protocol in which routers choose the route with the "best" path for inclusion in the routing table and announce the selected path to other neighbors. Selection of the best path depends on many factors, like local preference, commercial agreement or path length (refer to [126] for more details.).

Every BGP router is referred to as a "BGP speaker". Each of them is directly connected to a number of neighbors with whom it exchanges routing information and network packets. The AS node responsible for a specific network is said to be the *originator*, while all the AS nodes that can be traversed to reach that network constitute one *AS-path* for that prefix. Hence, the originator issues the *announcement* of a prefix, while other

The work described in this chapter is a joint work with L.V.Mancini, A. Spognardi, A. Villani and C. Soriente and it appeared in the International Conference on Computer Communication Networks (ICCCN2012), [94].

AS nodes along the AS-path *propagate* the announcement, appending their AS number to the received AS-path. Since a network prefix represents the identity of a network it is also called Network Layer Reachability Information (NLRI). An NLRI is composed of a length and a prefix. The length is a network mask in *CIDR* notation (e.g.: /25) specifying the number of network bits, and the prefix is the Network address for that subnet. An NLRI would look something like: 192.168.1.0/24..

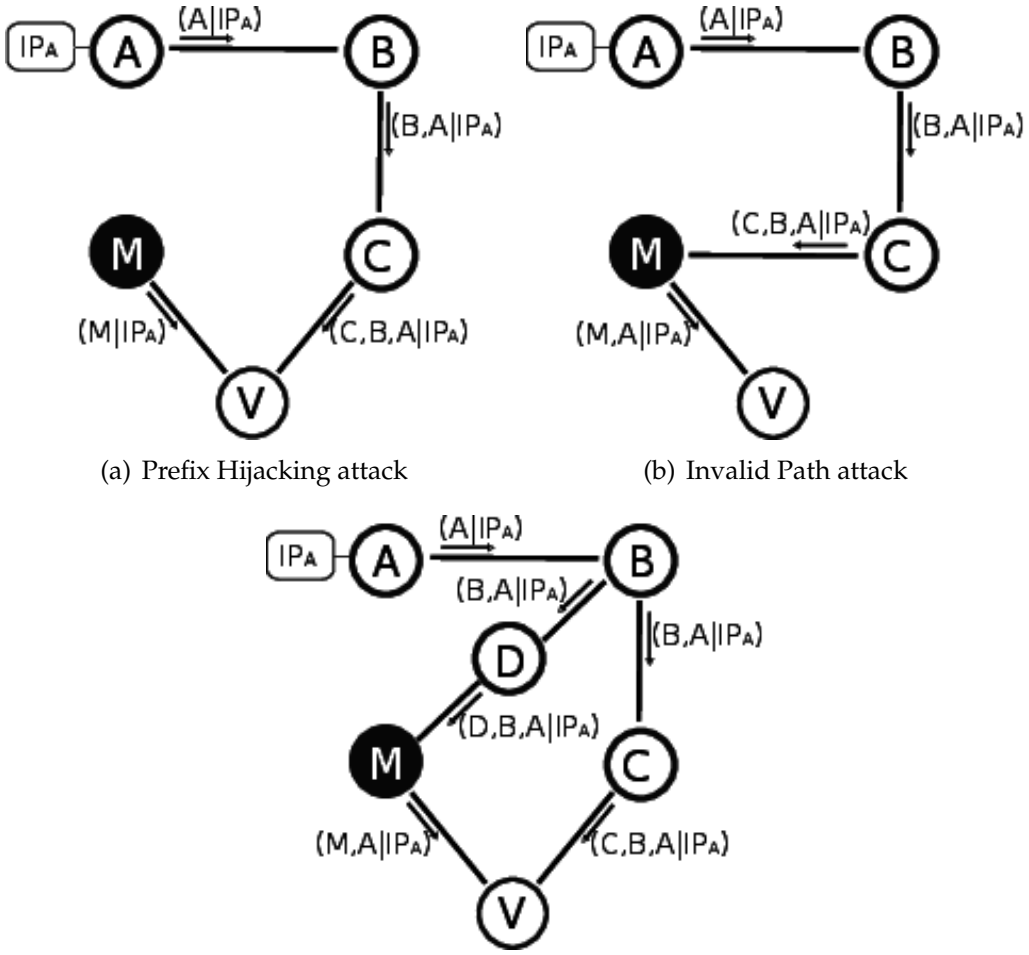
Once a network becomes unreachable for an AS node, it forwards a *withdrawal* message for that route to its neighbors: those, in turn, choose and propagate an alternative route (if any) for that network. In particular, BGP speakers exchange announcements, propagations and withdrawals within *UPDATE* messages.

3.1.1 Security threats and adversary model

The mechanism BGP uses to propagate routing data is extremely simple: an AS node receives announcements, store it and according with its routing policies, propagates them. This simplicity has determined its success but, at the same time, it is also the cause of many dangerous attacks against Internet. In fact, each speaker constructs its routing table only using the information received by its neighbors. Nothing prevents any speaker to behave maliciously and advertise to its peers prefixes it is not responsible of or routes that it cannot actually reach. This kind of behavior is the one *reBGP*, [94], seeks to mitigate.

Similarly to relevant work in the area [73, 67], this dissertation considers an active insider adversary that performs falsification attacks. The attacker can compromise any BGP speaker and inject fake BGP messages that do not reflect the real network topology. In particular, the adversary has access to all the cryptographic material the compromised router stores and can sign or encrypt any message on behalf of that router. The aim of the adversary is to subvert the correct BGP functioning in order to alter the normal routing of the traffic in Internet.

Protocol reBGP assumes that the adversary can falsify NLRI informa-



(a) Prefix Hijacking attack

(b) Invalid Path attack

(c) Blackhole attack

Figure 3.1: BGP threats: illegal prefixes and falsified AS-path attacks.

tion in order to perform a *prefix hijacking* and the AS-path attribute of an UPDATE message, in order to perform an *invalid path attack* [90].

Prefix hijacking

With this attack a compromised router can introduce UPDATE messages for networks that it does not administrate. For example, it could pretend to be the originator of a prefix 10.0.0.0/16 announcing that prefix to its neighbors (like the malicious AS node M of Figure 3.1(a) that announces IP_A owned by AS A). Or, it could announce more specific prefixes

of that network (like 10.0.0.0/18): longer prefixes, in fact, are preferred respect to smaller ones, since they are less specific (as happened in the Youtube hijack cyber attack).

Once the route eventually propagates to other AS nodes, the best path selection process for 10.0.0.0 would consider the regular path and the falsified one. In this way, the compromised router would attract all the traffic directed to 10.0.0.0 (or to a subnet) generated by the AS nodes that choose the falsified route.

Invalid path attack

In this attack the adversary issues UPDATE messages with a fabricated *AS-path*. For example the attacker can shorten a route of any NLRI simply removing some or even all the AS nodes from the AS-path attribute. This attack is depicted in Figure 3.1(b), where the AS node M drops from the AS-path for IP_A the AS nodes B and C. The invalid path attack can lead to a *blackhole* that is related to the *data plane* security issues. In particular, a *blackhole* happens when a malicious AS attracts traffic that would not flow through it, exploiting falsified AS-paths. For example, the AS node M of Figure 3.1(c) using a fake AS-path is able to attract the traffic towards IP_A from the victim AS V. Similarly, the attacker could modify the AS-path introducing some random AS in order to enlarge the path or to produce loops. In general, the effects of an invalid path attack would impact on the best route selection algorithm: shorter paths would be preferred against longer ones, while paths that include loops would be directly discarded as malformed routes.

3.1.2 Revocation in BGP

Just as in reBGP, signature-based solutions to secure BGP, e.g., [73], authenticate AS-paths by nesting the signatures of each AS on the path. In such a setting, revocation is performed by the trusted authority in charge of issuing public key certificates, that revokes the certificate of malicious or simply eliminated AS nodes. From that moment on, all announcements

made by the revoked AS are discarded by other peers. Other solutions, e.g. [67], propose to leverage the concept of "epochs", namely, they assume time is divided in periods of fixed length, after which all routes must be re-advertised. Nevertheless, a dynamic environment like the Internet requires fine-grained revocation means. For example, consider autonomous system AS_i that authorizes its neighbor AS_j to advertise its prefixes 10.0.0.0 and 11.0.0.0. Later on, AS_i changes its commercial strategy and decides that AS_j should keep advertising 10.0.0.0, but 11.0.0.0 should only be advertised by another peer, say AS_l. PKI certificate revocation is clearly a poor match in this scenario due to the large amount of communication and computational requirements. Moreover, AS nodes should retain complete control of the authorization they distribute or revoke, without resorting to a trusted third party. A possible solution would be for AS_i to sign a message that invalidates all announcements made by AS_j about 11.0.0.0 and to broadcast it. However, this would generate a considerable number of messages. reBGP tackles selective revocation embedding revocation information within AS-path announcement. In the above example, AS_i would just provide an authorization for AS_l to announce a path towards 11.0.0.0. As explained in the next sections, the authorization itself carries enough informations that allow other peers to accept the new route as the genuine one and to discard any further announcement made by AS_j on the old route for 11.0.0.0.

3.2 Cryptographic primitives

This section reviews the main cryptographic primitives used in the protocol reBGP, namely *Identity-based Aggregate Signatures*.

3.2.1 Identity-based Aggregate Signature

Traditional public key cryptography relies on one or more trusted authorities that issue certificates to bind users to public keys. Hence, before encrypting a message under one's public key, its authenticity must be ver-

ified by means of a valid certificate issued by the authority. Similarly, a certificate is required to validate the (public) verification key of a party, before verifying its signatures.

Certificate distribution and management requires a complex infrastructure known as Public Key Infrastructure (PKI). While traditional cryptography has been previously recommended to secure Internet-domain routing [73], the proposal has never been adopted because of the involvement of the PKI and complex trust relations among its components.

Identity Base (IB) Cryptography is an effective alternative to traditional Public Key cryptography that relies on certificates to endorse public key authenticity. In an IB cryptographic scheme, the identities of the parties "are" their public keys. In other words, any string can serve as a public key; corresponding private keys are managed and issued by a trusted authority, referred to as Private Key Generator (PKG). The latter is only active when issuing new keys.

The main advantages of the IBE Cryptosystems, are related to the its use of user identity's attributes (e.g email addresses or identity numbers) instead of digital certificates, for cryptographic operations. This environment significantly reduces the complexity of a cryptography system by eliminating the need for generating and managing users' certificates, so reducing communication and computational costs. Although an IBE cryptosystems avoids the need a PKI, it relies on a trusted third party, namely the Private Key Generator (PKG) which generates and issues the public/private keypairs. Furthermore, using such settings, there is no need to managing a public key infrastructure, including the CRLs (Certificate Revocation Lists).

It is relevant of mention other advantages of an IBC. In a IBC all the encryption keys are always available for all recipients. In fact, the encryption key is derived mathematically from the receiver's identity. This condition also assures the server can securely regenerate keys for recipients as needed.

Identity Based Aggregate Signatures (IBAS) allow multiple users to sign multiple messages and aggregate all signatures in a fixed-size token.

Given the identity of the signing parties, the signed messages and the aggregated signature, anyone can verify the validity of the signatures over their respective messages.

More formally, an IBAS scheme can be defined as a tuple of the form:

$$Setup, ExKey, Sign, Agg, Verify$$

where:

- $pk_{PKG}, sk_{PKG} \leftarrow Setup(\lambda)$ on input security parameter λ, outputs the PKG public/private key pair pk_{PKG}, sk_{PKG}.
- $pk_i, sk_i \leftarrow ExKey(sk_{PKG}, i)$ on input the PKG private key sk and a string i, outputs public/private key pair pk_i, sk_i.
- $\sigma \leftarrow Sign(sk_i, m)$ on input secret key sk_i and message m produces a signature on m under key sk_i.
- $\sigma \leftarrow Agg(\sigma_1, \ldots, \sigma_n)$ on input signatures $\sigma_1, \ldots, \sigma_n$ produces an aggregated signature σ.
- $\{0, 1\} \leftarrow Verify(pk_{PKG}, \sigma, m_1, ...m_n, i_1, \ldots, i_n)$ takes as input public key pk_{PKG}, a number of messages $m_1, \ldots, m_n$ with their respective signers $i_1, \ldots, i_n$ and the aggregate of their signatures σ; it outputs 1 if $\sigma \leftarrow Agg(\sigma_1, \ldots, \sigma_n)$ and, for $1 \leq i \leq n$, $\sigma_i \leftarrow Sign(sk_i, m_i)$.

Protocol reBGP uses the IBAS scheme proposed in [57] that allows multiple signers to sign multiple messages in such a way that the total verification information, apart from a description of who signed what, consists only of a short aggregate signature. In the following, the algorithms in [57] has briefly reviewed:
Setup(λ) is run by the PKG to define its secret key and public parameters:

1. Define groups G, G_T of prime order q and bilinear map $e : G \times G \rightarrow G_T$
2. Pick arbitrary generator $P \in G$

3. Randomly pick $s \in \mathbb{Z}_q$ and set $Q = sP$
4. Define Hash functions $H_1, H_2 : \{0,1\}^* \to G$ and $H_3 : \{0,1\}^* \to \mathbb{Z}_q$
5. Set $sk_{PKG} = s$ and $pk_{PKG} = G, G_T, e, P, Q, H_1, H_2, H_3$

$\mathbf{ExKey}(\mathbf{sk}_{PKG}, \mathbf{ID_i})$ is run by the PKG to compute the secret key of singer with identity ID_i:

1. Set $sk_i = \{sP_{i,0}, sP_{i,1}\}$ where $P_{i,j} = H_1(ID_i, j) \in G$

$\mathbf{Sign}(\mathbf{sk_i}, \mathbf{m_i})$ is run by signer with identity ID_i to sign an arbitrary message m_i:

1. Pick random string w
2. Compute $P_w = H_2(w) \in G$
3. Compute $c_i = H_3(m_i, ID_i, w)$
4. Pick random $r_i \in \mathbb{Z}_q$
5. Compute $\sigma_i = (w, S'_i, T'_i)$ where
   $S'_i = r_i P_w + sP_{i,0} + c_i sP_{i,1}$ and $T'_i = r_i P$.

$\mathbf{Aggregate}(\boldsymbol{\sigma_1, \ldots, \sigma_n})$ allows to aggregate an arbitrary number of signature that use the same w. The procedure can be run by any party as it requires no secrets.

1. Parse σ_i as (w, S'_i, T'_i)
2. Compute $\sigma_{1,\ldots,n} = (w, \sum_{i=1}^{n} S'_i, \sum_{i=1}^{n} T'_i)$

$\mathbf{Verify}(\mathbf{pk}_{PKG}, \sigma_{1,\ldots,n}, \mathbf{m_1}, \ldots, \mathbf{m_n}, \mathbf{ID_1}, \ldots, \mathbf{ID_n})$ is run to verify an aggregate signature $\sigma_{1,\ldots,n}$ where each singer ID_i has signed message m_i ($1 \leq i \leq n$).

1. Parse $\sigma_{1,\ldots,n}$ as (w, S_n, T_n)
2. Check that
   $e(S_n, P) = e(T_n, P_w) \cdot e(Q, \sum_{i=1}^{n} P_{i,0} + \sum_{i=1}^{n} c_i P_{i,1})$

The scheme is provably secure in the random oracle model, assuming the hardness of computational Diffie-Hellman over groups with bilinear maps.

3.3 Protocol reBGP

Protocol reBGP relies on the IBAS scheme described in the previous section. The latter guarantees the authenticity of both route announcements and AS-path announcements, with a constant overhead in terms of communication and verification.

Each announcement is signed by the sender for a specific receiver. The obtained signature is the *route endorsement* that authorizes the receiving AS to propagate those routes. This allows senders to control which peers can further propagate their announcements. In particular, when AS_i wishes to announce a route towards prefix IP_l to its peer AS_j, it signs the route AS_j, AS_i, IP_j.

Signatures are "nested" to thwart path truncation or AS replacement attacks. Hence, if AS_j wishes to further propagate the received announcement to peer AS_p, it adds AS_p on the path and signs it. In other words, AS_j signs AS_p, AS_j, AS_i, IP_l and sends the announcement to AS_p. AS_j also sends the aggregate signature resulting from the aggregation of its signature and the one received by AS_i. This signature is the route endorsement from AS_j towards AS_p.

Protocol reBGP assumes that all paths "originate" at the PKG. That is, if AS_i is assigned IP block IP_l, the PKG (i.e., the ICANN) signs the route AS_i, IP_l and sends the signature to AS_i. Clearly, AS_i owns IP_l so this hop is authenticated by the PKG. However, reBGP leverages this design choice to use a single algorithm to verify both route origination and AS-path announcements.

Since reBPG uses IB-cryptography, verification keys (i.e., the public key) do not need to be checked against certificates. Moreover, thanks to signature aggregation, communication and verification overhead is inde-

pendent on the number of signers (i.e., independent of the path length).

Protocol reBGP provides a simple yet effective mechanism to allow AS nodes to revoke announcements that were previously signed. This feature is particularly desirable in an extremely dynamic environment like the Internet, where relocation of IP blocks and modifications to agreements between AS nodes happens at a very high rate.

The basic idea is to include counters, denoted as α-values, in each signed announcement. Each AS that propagates an announcement for an IP block to a peer, adds a counter on the path before signing. Given two valid announcements (i.e., two announcements with valid signatures), $P = \ldots, AS_j, AS_{j-1}, \ldots, AS_1, IP_l$ and $Q = \ldots, AS'_j, AS_{j-1}, \ldots, AS_1, IP_l$, assume that they traverse the same AS nodes up to AS_{j-1} (i.e., $AS_j \neq AS'_j$). Any AS receiving P and Q will consider P as revoked if the α-value used by AS_{l-1} when signing P is smaller than the α-value used by AS_{l-1} when signing Q. If the two α values match, both paths are considered valid. The same technique can be used by the PKG to re-assign IP blocks. On the other side, AS must store sets of counters received in announcements by other peers for any IP block. The latter are referred to as β-values.

In the following, the operations carried out by the PKG and AS nodes in reBGP are provided.

System Setup

Protocol reBGP assumes the PKG to be the ICANN as the latter is already a trusted party in the Internet domain. At this time, the PKG sets up the parameters for an IBAS scheme as in [57] and runs $Setup(\lambda)$ to define its public/private key pair pk_{PKG}, sk_{PKG}.

AS join.

The AS nodes are identified by their number that is also used as a unique identity for IB cryptographic operations. When AS with number AS_i joins the system, the PKG runs $ExKey(sk_{PKG}, AS_i)$ to extract public/private key pairs pk_i, sk_i; the latter is securely delivered to AS_i.

AS_i also keeps two sets of counters. Set α_i has one counter for each IP block that AS_i knows a path to, such that $\alpha_i(l)$ is the α-value of AS_i related

to block IP_l. Set β_i has one counter for each AS on each path towards each IP_l that AS_i knows, so that $\beta_i(j,l)$ is the β-value stored by AS_i and related to AS_j that lies on the path $AS_i \rightsquigarrow AS_j \rightsquigarrow IP_l$. All records of both data structures are initialized to zero.

IP block assignment
Protocol reBGP treats assignment of IP blocks to AS nodes as BGP announcements from the PKG; in other words, when IP block IP_l is assigned to AS_i, the PKG "authorizes" AS_i to announce that IP block. In particular, PKG computes

1. $\alpha_{PKG}(l) = \alpha_{PKG}(l) + 1$
2. $m = AS_i, \alpha_{PKG}(l), IP_l$
3. $\sigma \leftarrow Sign(sk_i, m)$

and sends m, σ to AS_i. Note that message m states that PKG authorizes AS_i to advertise a path towards IP_l.

Route propagation
Assume that AS_i wants to advertise a route towards block IP_l to neighbor AS_j. Without loss of generality, denote with m the message that advertised[1] the route to AS_i and denote with α the corresponding signature. AS_i does the following:

1. $\alpha_i(l) = \alpha_i(l) + 1$
2. $m' = AS_j, \alpha_i(l), m$
3. $\sigma' \leftarrow Sign(sk_i, m')$
4. $\sigma^* \leftarrow Agg(\sigma', \sigma)$

and sends m', σ^* to AS_j. Note that AS_i adds $\alpha_i(l)$ to the signed message. The α-value will be used by receiving AS nodes to verify the freshness of the announcement. In particular, increasing $\alpha_i(l)$ before including it in

[1] In case AS_i is the owner of IP_l, then $m = AS_i, \alpha_{PKG}(l), IP_l$.

the message to AS_j guarantees that AS_j is the only AS that is currently authorized by AS_i to announce routes towards IP_j. In other words, all previous authorizations endorsed by AS_i to advertise a path towards IP_l are implicitly revoked.

Route verification
Assume that AS_i receives a message with a path towards IP_l, that is, the received message is

$$m = AS_i, \alpha_{i_n}(l), AS_{i_n}, \ldots, \alpha_{i_1}(j), AS_{i_1}, \alpha_{PKG}(l), IP_l$$

where $\alpha_{i_n}(l), AS_{i_n}, \ldots, \alpha_{i_1}(l), AS_{i_1}$ might have zero length if AS_i has been assigned IP_l by the PKG.

Assume also that σ is the aggregate signature for this message. AS_i does the following:

1. For $w = n-1, \ldots, 1$, check if $\alpha_{i_w}(l) \geq \beta_i(i_w, l)$; otherwise discard the message as the route is not the most up to date to reach IP_l.

2. If the path has length greater than 1, check if $\alpha_{PKG}(l) > \beta_i(PKG, l)$; otherwise the block IP_l is not assigned to AS_{i_1} anymore.

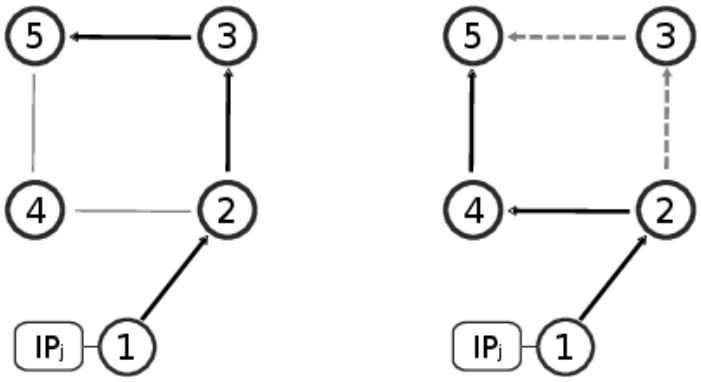
3. Check if $1 \leftarrow Verify(m, \sigma)$; otherwise discard the message as at least one signer failed to compute a valid signature.

If all checks succeed, the route is valid and AS_i can update its routing tables. This includes setting $\beta_i(j, l) = \alpha_j(l)$ for $j \in \{1, \ldots, n-1\} \cup \{PKG\}$.

The α and β values are used in reBGP for route revocation and multi-homing (Figure 3.2). In Figure 3.2(a), AS_5 reaches IP_j owned by AS_1, through AS_3 and AS_2. That is, AS_5 received the following message from A_3:

$$m = AS_5, \alpha_3(j), AS_3, \alpha_2(j), AS_2, \alpha_1(j), AS_1, \alpha_{PKG}(j), IP_j$$

Now assume that AS_2 changes its routing policy and decides that AS_4 should be its upstream peer for routes towards IP_j; at the same time AS_2



(a) (b)

Figure 3.2: Route revocation in BGP.

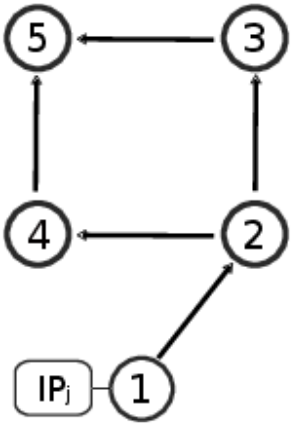


Figure 3.3: Route revocation in BGP for Multi-Homing AS.

decides to revoke this capability to AS_3. The updated topology is depicted in Figure 3.2(b). In order to endorse the new routing policy, AS_2 sings a new AS-path towards IP_j where the link AS_2, AS_4 has $\alpha_2(j)$ greater than the one used by AS_2 when it authorized the link AS_2, AS_3. In particular, the message sent by AS_2 to AS_4 is

$$\tilde{m} = AS_4, \tilde{\alpha_2}(j), AS_2, \alpha_1(j), AS_1, \alpha_{PKG}(j), IP_j$$

where $\tilde{\alpha_2}(j) > \alpha_2(j)$. Eventually, AS_4 will announce a route towards IP_j to

AS_5. The latter will detect the updated α-value used by AS_2 and acknowledge the new path as the only valid one.

Another possibility is for AS_2 to keep AS_3 as an upstream AS for paths towards IP_j and, at the same time, authorize also AS_4 to announce routes towards that IP block. This scenario, referred to as "multi-homing", is depicted in Figure 3.3. In order to do so, AS_2 sends to AS_4 the following message:

$$\tilde{m} = AS_4, \tilde{\alpha_2}(j), AS_2, \alpha_1(j), AS_1, \alpha_{PKG}(j), IP_j$$

where $\tilde{\alpha_2}(j) = \alpha_2(j)$. Each AS (including AS_5) receiving the new AS-path announcement by AS_4 will detect no changes in the α-value used by AS_2 in announcements related to IP_j and will acknowledge the new route as an alternative one.

3.3.1 Security analysis

Security of reBGP is straightforward assuming the security of the underlying IBAS scheme. As the network prefixes are assigned by the PKG through prefix endorsements, a prefix hijack attack would require the adversary to forge signatures by the PKG on arbitrary messages. Similarly, invalid path attacks require the adversary to forge signatures by other AS routers. One major drawback of IBAS is that an unsuccessful verification of an aggregate signature cannot be traced back to the signer(s) that produced invalid signatures. In other words, any party can mount a Denial of Service (DoS) attack modifying an aggregated signature so that the verification algorithm would fail. However, DoS attacks are beyond the scope of reBGP. Finally, similarly to other related works [67, 59], protocol reBGP does not consider collusion among compromised routers (i.e. the collusion attacks or tunneling attacks, [117]) and the design of a secure routing protocol that effectively counters a number of colluding nodes is still an open research problem.

3.4 Evaluation

The evaluation of the overall overheads includes the estimation of the space requirement in terms of volatile memory (Section 3.4.1), computational costs (Section 3.4.2) and time delay (Section 3.4.3), namely the expected convergence time.

Similarly to other works, [90, 158], in order to evaluate the computational overhead and the introduced delay, the SSFNet simulator [137] has been used. SSFNet provides a simulation environment for BGP as well as for other network protocols. All the measurements of reBGP are compared with S-BGP.

3.4.1 Space complexity

Although nowadays BGP routers have reasonable resources, volatile memory represents a strict constraint for routing algorithms; notably, this aspect becomes more and more relevant when routers have to maintain the routing table for all reachable network prefixes in the Internet. The most important contribution to memory occupation is related to cryptographic material, namely signatures and keys.

Signature requirements

In [73], the authors suggest to implement S-BGP with the Digital Signature Algorithm (DSA), mainly due to its smaller signature, compared to RSA. With 1024-bit keys, DSA produces 40-byte signatures, versus 128-byte signatures typical of RSA. This thesis considers S-BGP using DSA with 1024-bit keys.

In contrast, reBGP leverages elliptic curves in order to use smaller keys but still afford *computationally equivalent security* [83]. In particular, reBGP uses 140-bit keys with 120-byte signatures and offers the same security level of DSA with 1024-bit keys.

While S-BGP DSA signatures grow linearly with respect to the AS-path length, reBGP aggregated signatures are fixed in size, regardless of the

path's length. This aspect makes reBGP particularly suitable to protect path vector protocols like BGP. In order to estimate memory requirements for S-BGP and reBGP with real data, the Routing Information Base (RIB) of one of the collector routers owned by RIPEncc[2] has been considered. The RIB is the table that holds all routing information received from routing peers, namely, it contains multiple paths for each IP prefix. In particular, it has been dumped the RIB of the `rrc00` router[3], dating back to February 1st, 2012: this router received 18 IPv4 full Internet tables from 28 active peers, totaling about 7 millions of routes, with the related AS-path attribute. Figure 3.4 shows the distribution of AS-path lengths, where the maximum AS-path length was 13 (without prepend). Please note that the figure does not consider prepending since both S-BGP and reBGP verify an AS in the AS-path only once.

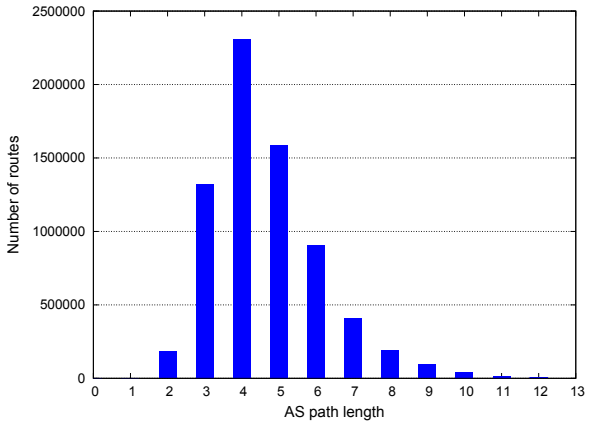


Figure 3.4: Composition of the RIB table of the `rrc00` RIPEncc router.

The equation suggest the memory requirements Q of the `rrc00` RIB table:

$$Q = \sum_{i=1}^{N} |RIB_i| \cdot s(i) \cdot sizeof(\sigma) \tag{3.1}$$

where $|RIB_i|$ represents the number of entries of length i, N is the max AS-path length, the value $s(i)$ is the number of required signatures for a

[2]Please refer to http://www.ripe.net/ for more details.

[3]The rrc00 router is part of the Routing Information Services offered by the RIPEncc(`http://www.ripe.net/data-tools/stats/ris/`)

	Memory requirement		
	Key (bits)	**Signature** (bytes)	`rrc00` **RIB** (MB)
S-BGP	1024	40	7984
reBGP	140	120	848

Table 3.1: Memory requirement comparison among S-BGP and reBGP.

prefix of length i and $sizeof(\sigma)$ is the size in bytes of a single signature. In particular, $s(\cdot)$ will be:

$$s(i) = \begin{cases} 1 & \text{with aggregate signature} \\ i & \text{otherwise} \end{cases}$$

Consistently with equation 3.1, the memory requirements of the the RIB of `rrc00`, is 7984 MB and 848 MB for S-BGP and reBGP respectively, namely a reduction of 9,4 times. Memory requirements for the signatures are summarized in Table 3.1.

Timestamp requirements

As seen, introducing the timestamps for every AS and for each prefix in the RIB enables the possibility to revoke prefix and route endorsements. Timestamps are a simple but effective mechanism that could also be applied by other secure versions of BGP. This equation evaluate the additional memory requirement T:

$$T = \sum_{i=1}^{N} |RIB_i| \cdot i \cdot sizeof(timestamp) \tag{3.2}$$

Considering timestamps of 8 bytes, the additional overhead grows around 320 MB. Combining equation 3.1 and 3.2, the total memory requirement of reBGP is 1168 MB, with a reduction of 6.8 times with respect to S-BGP.

3.4.2 Computational complexity

To evaluate the overhead introduced by the cryptographic operations, this dissertation considers a simple *chain topology* (Figure 3.5) of N Autonomous Systems and estimates the time required to perform signatures and verifications on all the nodes of the chain. In such topology, AS_1 is the

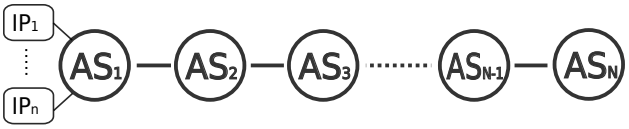


Figure 3.5: Network topology used to estimate the computational overheads of reBGP.

source node, since it is the first node of the chain and is the only one that owns and advertises some prefixes to the rest of the network. AS_N is the *sink node*, that is the last node of the chain: it only receives the advertised prefixes from the preceding node, but does not forward any message. All the other AS nodes are *transit nodes*, since they receive all the advertised prefixes by the preceding node and forward them to the following node in the chain.

Considering public key based BGP protocols (like S-BGP with DSA), the transit nodes are the most overloaded, since they handle both signature verification and generation for any received prefix. On the other hand, the source and the sink nodes only perform signatures generation and verifications, respectively. Clearly, AS_{N-1} is the node with the maximum overhead, since it has to verify every advertised prefix with the longest path and has to sign every prefix for AS_N. For this reason, the node in position $(N-1)$ is defined as *critical node*. As the above section, according to [141], the maximum AS-path length is $N = 13$.

To evaluate the overhead required by the cryptographic operations of the protocol reBGP, new software has been implemented using the *PBC Library* of the Stanford University, [91], which is an experimental free C library for elliptic curves arithmetic and pairing computation.

Since there is no publicly available version of S-BGP, its cryptographic

operations have been implemented using the *OpenSSL Library*, [111]. In order to obtain a fair comparison, a non optimized version of the OpenSSL library has been used. This fill the gap with the experimental grade of the PBC library. The reported results were obtained using an Intel(R) Core i5 540m CPU at 2.53 GHz with 3072 KB of L2 cache and equipped with 4GB of DDR3 memory and averaging the results of 100 experiments.

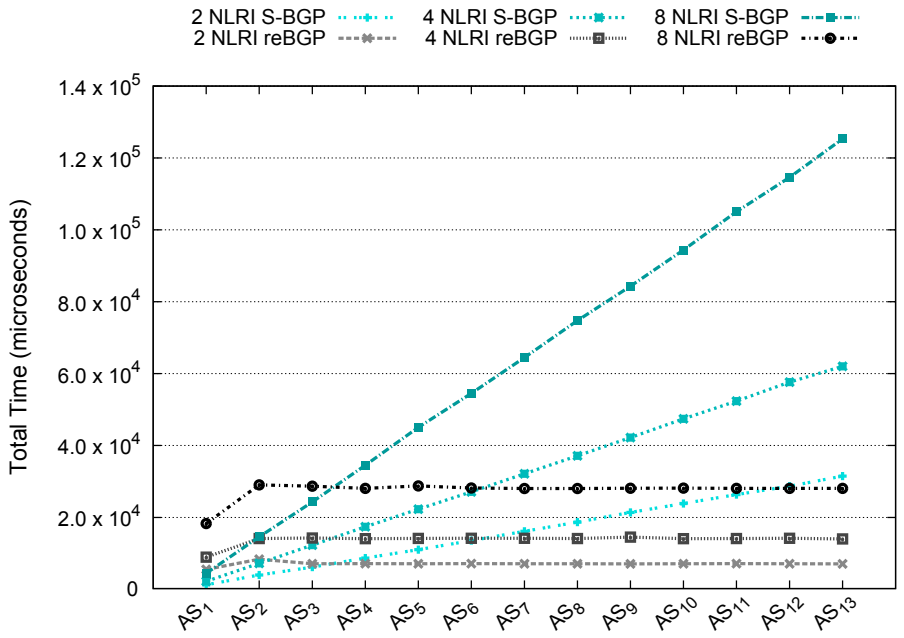


Figure 3.6: Cumulative overhead of every AS in the simulated environment.

Figure 3.6 shows in μs the cumulative overhead (i.e. signature cost plus verification cost) experienced by every AS along the chain topology. For the same AS-path length, this experiments varied the number of NLRI advertised by AS_1, setting them to 2, 4 and 8. As expected, the overhead introduced by S-BGP grows linearly, since it uses DSA: each transient node has to verify and sign one time for every hop from the source node. reBGP, instead, introduces always a constant overhead, regardless to the AS-path length. Considering the lines of reBGP, it's easy to note that the reBGP overtakes S-BGP starting from an AS-path length equal to 4. Even if a single pairing operation is more expensive than both a DSA verification and DSA signature, the continuous growth of the AS-path quickly reduces this

advantage and worsen the performances of S-BGP. It is worth noting that from Figure 3.4 there are less than 3% of the announced routes have a path length lower than 4. Furthermore, the more the AS-path length grows, the more the advantage of reBGP over S-BGP increases. The resulted values have been used as reference for the delays introduced in the SSFNet simulator to evaluate the convergence time of the routing information in the network, as shown in the next section.

3.4.3 Convergence Time

The above computational overhead estimation provides a fundamental analysis of the protocols. Since BGP is a distance vector protocol, exactly a path vector protocol, each time a router handles a message the delay introduced by the cryptographic operations would accumulate in time. The delays, then, would broaden the time needed by the Internet to converge, namely to reach a stable state in which each node (router) has the required information [60, 33, 79]. As other works, this dissertation presents an analysis of the convergence time based on SSFNet, [137], namely, a tool for scalable high-performance network modeling, simulation, and analysis. It is a Java-based and event-driven simulation package able to model large and complex networks, just like the Internet. The experiments used SSFNet with networks of different sizes that exhibit the same properties of Internet, [33, 51].

To evaluate protocol reBGP and propose a fair comparison with S-BGP and simulate the effects of the cryptographic operations on the CPU, a time delay for each message management has been added. The introduced time delay were taken from the evaluation results of the computational overhead analysis, as exposed in the above Section 3.4.2.

Before introducing experiments results, as observed in [60], it's important to note that there are two primary causes of BGP delayed convergence: the first concerns the distributed nature of BGP path selection; the second relates to the Minimum Route Advertisement Interval (MRAI) value, that is a timer that limits the minimum time interval between two

consecutive update messages sent for the same destination. While the main purpose of MRAI is to provide protection against route flapping attacks, [143], one of the effects is the increase of the convergence time. Furthermore, it allows a router to collect several UPDATEs before to send a new UPDATE to the same peer.

In order to get rid of the effects of the MRAI, this value is set to 0, while the suggested value is 30 seconds. The results show that the difference between the convergence time of BGP and its secure versions is due only to the introduced delays. The effects of MRAI, in fact, in the relatively small networks simulated with SSFNet would completely hide the delays of the cryptographic operations.

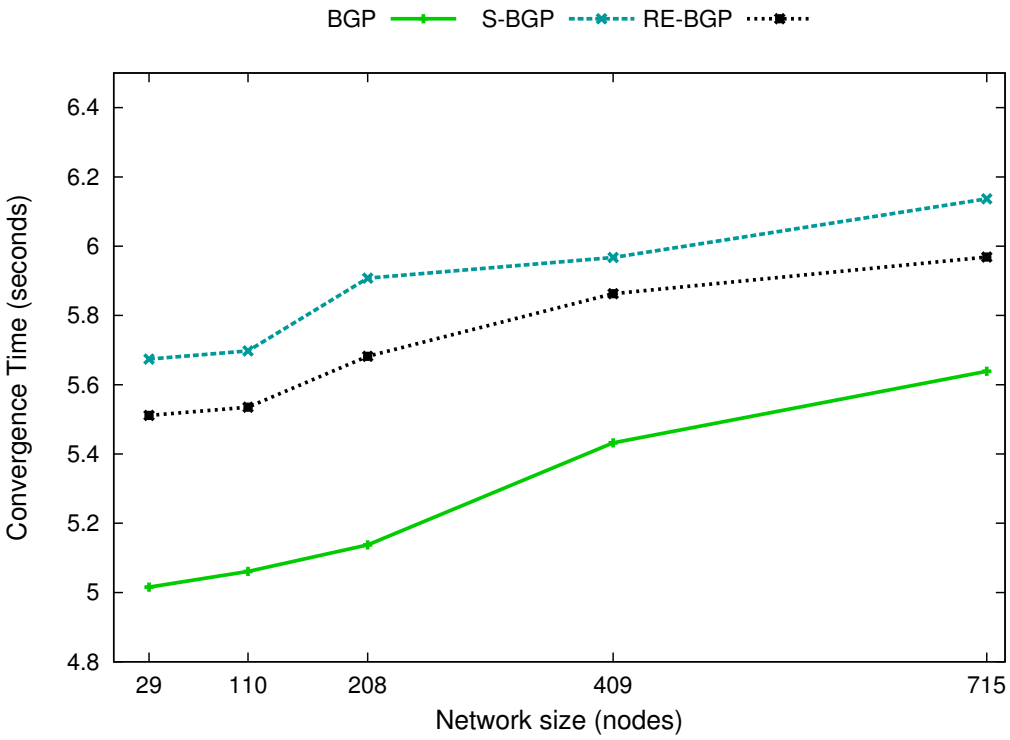


Figure 3.7: Estimated convergence time of BGP, S-BGP and reBGP.

Figure 3.7 shows the impact of reBGP and S-BGP on the convergence time with several network sizes. As said, the network topologies are part of the SSFNet project and are publicly available. These topologies describe networks with 29, 110, 208, 409, and 715 nodes (AS), in which the most

relevant Internet graph properties are preserved, [33, 51]. Thanks to its aggregate signatures, reBGP introduces smaller delays compared to S-BGP when the path length is greater than 4 (see Section 3.4.2). In this way, the total convergence time of reBGP is always smaller than the one of S-BGP.

Chapter 4

On the difficulty of Denial of Service Attacks detection

This chapter shows the effectiveness of metrics based on information theory and evaluates them using a huge data set of real network flows.

4.1 Background

One of the most critical aspect of DDoS (Denial of Service and Distributed DoS) attacks is their artlessness and simplicity.

While the synchronization of attacking entities is still performed using botnets of unaware compromised hosts, nowadays, simple word-of-mouth ways are used to coordinate volunteers attackers (e.g., chat, twitter, irc channels or others). Recently, one of the most used tool to perform DDoS is "LOIC" (Low Orbit Ion Cannon), a software originally designed to test the robustness of network services and able to quickly flood a target IP with connections. LOIC and similar tools make these activities exploitable by all Internet users: political *hacktivists*, individuals or interested groups keep increasing their use to express disagreement against private companies or public entities. Many examples can be found in the past

The work described in this chapter is a joint work with L.V.Mancini, A. Spognardi, A. Villani and it appeared in the 9th International Conference on Security and Cryptography (SECRYPT2012), [43].

years [28]. For example, in September 2010, a DDoS attack named *Operation Payback* was launched against the Motion Picture Association of America's (MPAA) web-page. Similarly, strong emphasis was given to the series of DDoS attacks against several companies which resulted in a cut off for WikiLeaks.org or to the Playstation's online store as a form of revenge against Sony's lawsuit against the PS3 hacker George Hotz. More recently, DDoS attacks have been reported to the Italian Government, the Vatican State web sites and many other international institutions. In general, every Internet Critical Infrastructure or any sensitive economic service can be considered a possible target.

The effects of DDoS attacks can be serious: in the best cases, the network services hosted by the target Autonomous System become unavailable as long as the attack activity persists; in the worst cases, the session between the target AS and its ISP breaks out, making a black hole where the packets are all dropped, eventually causing a chain reaction that amplifies the attack and spreads its effect on other AS nodes. DDoS attacks are considered really challenging and have generated a large amount of research activity. In the last decade, several works try to survey metrics, strategies and tools to protect network services and to reduce the impact of such malicious activities. At the same time, new attack flavors (ip-spoofing, low-rate attacks, botnet and others) keep raising the level of challenge.

Finally, privacy concerns and the lack of secure techniques to make data anonymous, keep researchers unable to freely share their own traffic datasets and network dumps, slowing and hindering the research on this topic.

This chapter focuses on DoS and Distributed DoS attacks that consume the bandwidth resources of a whole AS node. In the DDoS taxonomy defined by Mirkovic et al., [101], such kind of attack has code VT-4, since it generates an extremely large number of network flows, saturating all the router resources (CPU or ram or bandwidth capacity) of the AS. Increasing router resources is typically helpless against bandwidth saturation attacks, mainly for *stub* AS nodes: they usually purchase the minimal

required bandwidth, but suffer DDoS attacks from intermediate AS nodes, with much higher traffic capacity.

Typical defenses, usually adopted in *stub* AS nodes, avoid memory or CPU saturation [37], but are typically helpless against bandwidth saturation attacks: the high bandwidth pathways usually lie in the intermediate networks, while the end networks purchase only as much bandwidth as they usually need. A more effective solution is to block the malicious traffic in advance, in the upper AS nodes, before it could reach the target AS. The most used approach to distinguish malicious packets among the aggregated traffic at AS level is the adoption of information theory metrics, since they are able to make traffic anomalies to "emerge" from the whole traffic flows. As it has been shown in section 2.2, the effectiveness of proposed metrics is evaluated using synthetic traffic, where attack patterns are artificially injected.

The experiments shown in this chapter are based on the collected traffic that flowed through an important Italian Tier II AS, as shown in Figure 2.1, that plays the role of transit for some stub AS node and shares connections with other ISPs in a IXP (Internet eXchange Point). In order to protect customers privacy, no references about real IP addresses, identities or related contents are provided.

The collection probe recorded meaningful high resources network events and several attacks; these events have been used to evaluate and estimate the effectiveness of information theory metrics for DDoS attack detection, just using CISCO^{TM} NetFlow data set.

4.2 Entropy and Relative Entropy Metrics

The use of entropy analysis aims to capture fine-grained patterns in traffic distributions, that simple volume based metrics cannot identify. Interestingly, information theory based metrics enable sophisticated anomaly detections directly with the whole traffic that are difficult to provide with simpler metrics, like aggregated traffic workload, number of packets or

single host traffic. As it will be described in the next sections, the events detected by combinatorial metrics are not really predominant when observed with traditional ways: within the aggregated traffic of the monitored ISP (order of 1Gbit/s), a DDoS attack against a single VLAN is not noticeable, since the Mbits needed to perform a DDoS attack are well-hidden in the aggregate traffic, and would not determine any apparent anomaly. On the other hand, the Kullback-Leibler divergence is effectively able to notice the anomaly and to raise an alarm. To provide the same level of accuracy, any traditional metric should be continuously evaluated on every possible target, in order to detect the anomalies. Combinatorial metrics, instead, are able to detect the anomalies within the whole traffic; moreover, those are less affected by the fluctuations of traffic workload or any other quantitative measure.

The first metric evaluated is the *simple entropy*, that captures the degree of dispersal/concentration of a distribution. Then, the experiments consider two relative entropy measures, namely the *Kullback Leibler* divergence [85] and *Rényi* divergence [86].

The concept of *Entropy* was introduced by Shannon in [133]. The classic definition says that entropy is a measure of the *uncertainty associated with a random variable*. The entropy $H(X)$ of a discrete random variable X is defined as:

$$H(X) = -\sum_i p_i \log_2 p_i \tag{4.1}$$

where $p_i = P[X = i]$ is the probability that X assumes the value i.

Relative entropy (also known as *information divergence*) is a non symmetric measure of the similarity between two probability distributions P and Q and quantifies the distance between two statistical objects. The Kullback-Leibler divergence equation [85] used is:

$$D_{KL}(P||Q) = \sum_i P(i) log \frac{P(i)}{Q(i)} \tag{4.2}$$

A low D_{KL} value means a high similarity in the two probability distributions, on the other hand, high divergence values correspond to low simi-

larity. Note that, since it is not symmetric, the divergence measure can not be strictly considered a *metric* (i.e., $D_{KL}(P||Q) \neq D_{KL}(Q||P)$).

The Rényi divergence generalizes the Kullback-Leibler divergence, providing a family of metrics based on a parameter α. Formally [86]:

$$D_\alpha(P||Q) = \frac{1}{1-\alpha} \log \sum_i \frac{p_i^\alpha}{q_i^{\alpha-1}} \tag{4.3}$$

Notice that $D_{\alpha \to 1}(P||Q) = D_{KL}(P||Q)$. Intuitively, Rényi divergence with high values of α takes in higher account the more likely events, while with low values of α, it considers more equally all the events, regardless of their likelihood.

4.2.1 Metrics implementation

This section describes how to exploit netflow data to implement the above-mentioned metrics. As stated by [109], port and IP address distributions are highly correlated in network traffic. For this reason, only source and destination IP have been considered.

Network flows are aggregated into time blocks of a fixed size (1 minute by default). Let f^t be the number of flows that cross the monitored network in a time block. Let f_i^t be the number of flows that have IP_i as source (or destination) address. For each time block t, the entropy is evaluated by the following formula:

$$H(X) = -\sum_{\forall distinct\ IP\ i} \frac{f_i^t}{f^t} \log_2 \frac{f_i^t}{f^t} \tag{4.4}$$

Concerning the relative entropy metrics (Kullback-Leibler and Rényi), p_i describes the packet distribution over a time block t, while q_i describes the packet distribution of the previous time block $t-1$:

$$p_i = \frac{f_i^t}{f^t}, q_i = \frac{f_i^{t-1}}{f^{t-1}}$$

The Kullback-Leibler divergence is computed as follows:

$$D_{KL}(t||t+1) = \sum_i \frac{f_i^t}{f^t} \log \frac{\frac{f_i^t}{f^t}}{\frac{f_i^{t-1}}{f^{t-1}}} = \sum_i \frac{f_i^t}{f^t} \log \frac{f_i^t f^{t-1}}{f^t f_i^{t-1}}$$

and Rényi divergence as:

$$D_\alpha(P||Q) = \frac{1}{1-\alpha} \log\left(\sum_i \frac{(\frac{f_i^t}{f^t})^\alpha}{(\frac{f_i^{t-1}}{f^{t-1}})^{\alpha-1}} \right)$$

Notice that the experiments only consider the entries that appear in both t and $t-1$ time blocks, since the relative entropy imposes $Q(i) > 0$ for each $P(i) > 0$. Another key aspect is the choice of the parameter α in the Rényi divergence. According to the results in [86], the experiments set the value of α to 5. The time block dimension affects the relationship between detection *reactivity* and detection *sensibility*, directly influencing the results. This thesis assumes that 1 minute is a good compromise between reactivity and sensibility.

4.3 Attack and anomaly analysis

This section reports the comparison of the three metrics presented in the above section, applied to several anomalies collected in the monitored network. Remember that the netflow dataset refers to the period between September 2010 and August 2011, that has been the scenario for several DDoS episodes, in Italy and abroad. In order to make a complete and fair comparison between Entropy, Kullback-Leibler and Rényi metrics with the previous research results, the experiments evaluated the above metrics considering separately the destination and the source *IP* distributions.

The experiments considered the whole dataset of netflows and are based on the evaluation of the three metrics for all the 12 months. However, since there is no complete knowledge of the attacks happened during the

whole period, the evaluations only considered as attacks the official report of the AS network administrators. As depicted in the next figures, the reported anomalies correspond to metric fluctuations that produced peaks in their values. Once such high peaks were identified, deeper inspection was conducted in order to capture the motivations behind the anomaly. This kind of analysis produced several insights about the behaviors and limitations of the metrics.

In addition to the reported anomalies, this chapter analyzes another kind of network activity, namely the abnormal traffic generated by scheduled and automated administration activities (e.g., scheduled backups or maintenance procedures). Since those activities can make sensible service outage to users, they are usually programmed in the period that spans from 12:00am to 8:00am, when regular traffic is low and the amount of flows reaches its minimum. By observing the relative netflows, it is possible to identify sudden and relatively short mutations of the traffic pattern, resulting in a deep alteration of the metrics.

4.3.1 Sample events

This section reports the results of four sample events (E_1, E_2,E_3, E_4): Table 4.1 summarizes how the implemented metrics (Entropy H, Kullback Leibler KL and Rényi R) reacted during these events. The three metrics are evaluated both on source and destination IP with the exception of the Rényi divergence which is evaluated only on destination IP. The results do not report the Rényi on source address due to its extremely fuzzy behavior. The shortened form H_s, KL_s and H_d, KL_d, R_d refer to entropy, Kullback Leibler and Rényi respectively evaluated on source and destination IP distributions.

Table 4.1 reports how all metrics (the columns of the table) behave when each event (the rows of table) happens. If an abrupt variation to a higher value has been observed, the cell of the table says that the metric *Increases*; in the opposite case, the cell says that the metric *Decreases*. Finally, the *Unvaried* value indicates the absence of observable variation

with.

	E_1 (DoS)	E_2 (DoS)	E_3 (DDoS)	E_4 (Routines)
KL_d	*Increases*	*Increases*	*Increases*	*Unvaried*
H_d	*Decreases*	*Decreases*	*Decreases*	*Increases*
R_d	*Unvaried*	*Unvaried*	*Unvaried*	*Unvaried*
KL_s	*Unvaried*	*Increases*	*Increases*	*Unvaried*
H_s	*Decreases*	*Decreases*	*Increases*	*Decreases*

Table 4.1: Malicious events and Denial of Service metrics behavior.

In order to avoid metrics overlapping, the results of the Entropy and Kullback-Leibler with the source IP have been plotted mirrored with respect to the x axis.

E_1 — DoS attack This episode has been classified as a DoS attack. In fact, traffic statistic (table 4.2) shows that there was a single *IP* playing a primary role during the attack against one single host server. There were also few other *IP* addresses participating to the attack, with a smaller contribution.

Source	Flows
IP1	40.6%
IP2	28.3%
IP3	8.8%
IP4	0.4%
IP5	0.3%
IP6	0.3%
IP7	0.1%
...	...

Table 4.2: E_1 - traffic statistics.

All the metrics correctly detected the malicious activity, as shown by the fluctuations and the spikes of Figure 4.1. Indeed, Rényi distribution shows the lower peak.

The DoS nature of the attack is well described by the downfall of both entropy lines (in the lower part of the plot) around *4:00pm*. This behavior expresses that a small number of source addresses generates the largest

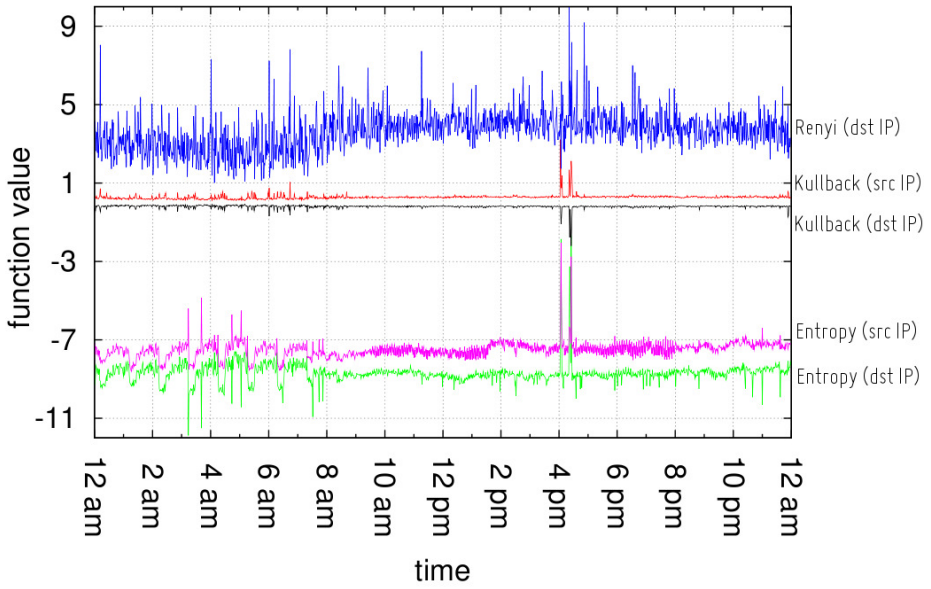


Figure 4.1: E_1 — metrics comparison for a DoS attack.

amount of connections towards a small set of destinations, namely the typical scenario of a DoS attack. At the same time, KL on destination IP grows significantly. To have a deeper insight of KL behaviors, the graphs plot the contribution of every destination IP to the final KL_s and KL_d values. The Figure 4.2 (upper) reports the first ten time-blocks since the beginning of the malicious activity. It is evident how the final value of KL_d is obtained by the contribution of one main component (the victim host), while the contribution of the other hosts is negligible. On the other hand, since during a DoS attack, only few sources generate the largest part of the traffic, each attacking host addresses many flows towards the victim host. This anomaly is perfectly captured by the peaks of lower Figure 4.2, that corresponds to the main contributors to the KL_s value.

In the same plot of Figure 4.1, it is possible to observe the anomaly described as administration activity (maintenance jobs) and labeled with E_4: before *7:00AM* indeed both source and destination entropy metrics rise and fall continuously, since they generate maximum (respectively minimum) traffic compared to high (respectively low) traffic.

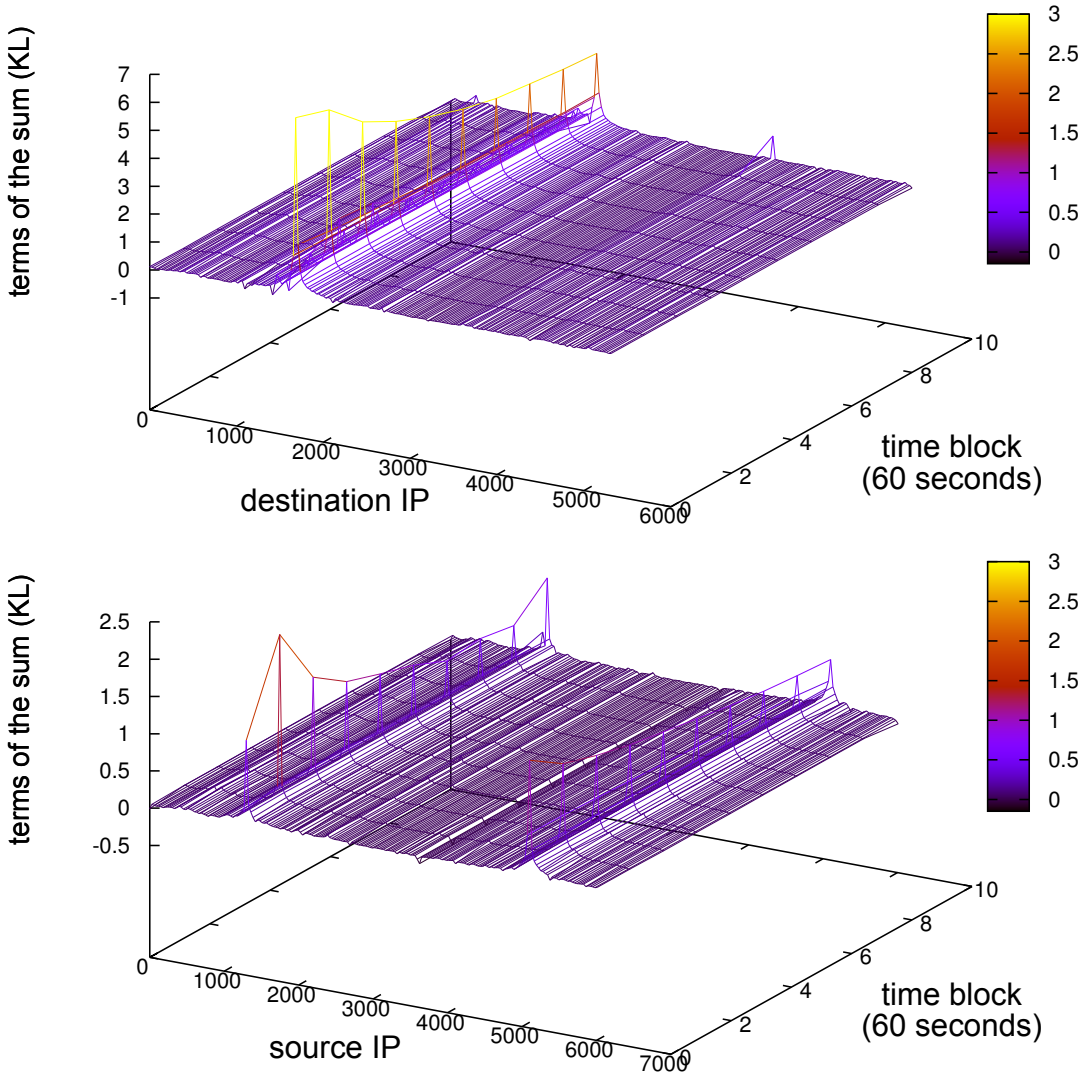


Figure 4.2: Kullback-Leibler details for E_1 on source and destination IP address.

E_2 — DoS attack In this episode, the main contribution to the attack comes from a single *IP*, and it consists in more than 50% of all the flows towards one single victim host; moreover, the five most active *IP*s have generated the 93.8% of the whole traffic (table 4.3).

In this event, the victim host does not involve a large portion of network flows, while several other services of the networks (web, mail, DNS servers etc.) generated the larger amount of flows. Nevertheless, the traf-

Source	Flows
IP1	58.7%
IP2	13.0%
IP3	12.8%
IP4	7.0%
IP5	2.3%
IP6	0.4%
IP7	0.2%
IP8	0.1%
...	...

Table 4.3: E_2 - Traffic statistics.

fic diversity expressed when the attack occurred has been well detected by both *KL* measures. This aspect represents a scalability factor of this measure and suggests that the attack is detectable among the whole aggregated traffic: the attack emerges from the traffic thanks to its informational fingerprint. As entropy line shows (Figure 4.3), the attack starts soon after *1:00PM*.

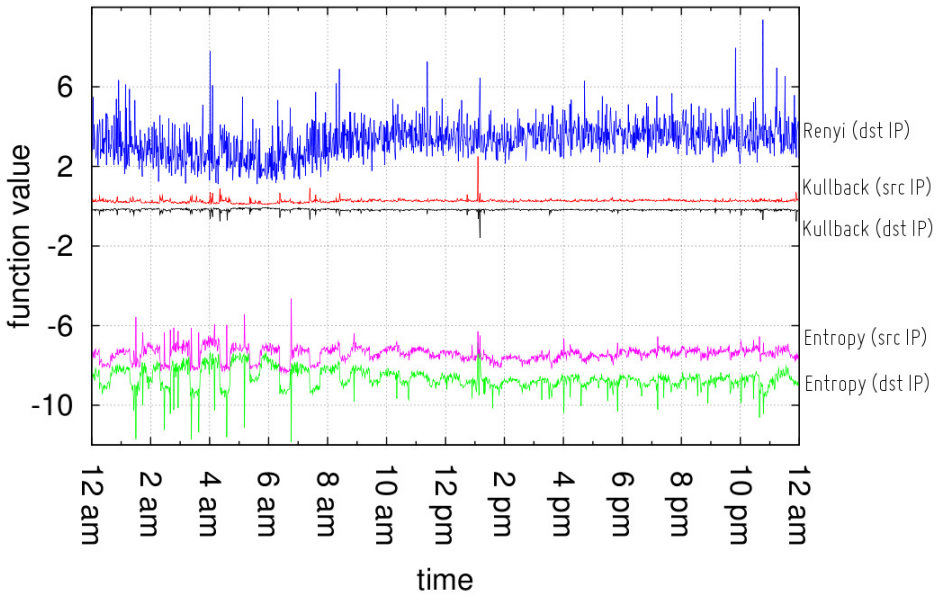


Figure 4.3: E_2 — metrics evaluations for a DoS attack.

The entropy on both attributes decreases, representing a non-uniform distribution of destination IP as well as source IP fields. The Rényi dis-

tribution reveals a small peak, but this is hidden by the fuzzy behavior it exhibits.
Even in this case it is possible to observe the perturbations due to the maintenance jobs: in the case of the entropy, the peaks are higher than the ones relative to the attack E_2, generating some false positive (as it will be clear in the following). Rényi divergence also suffers the same issue.

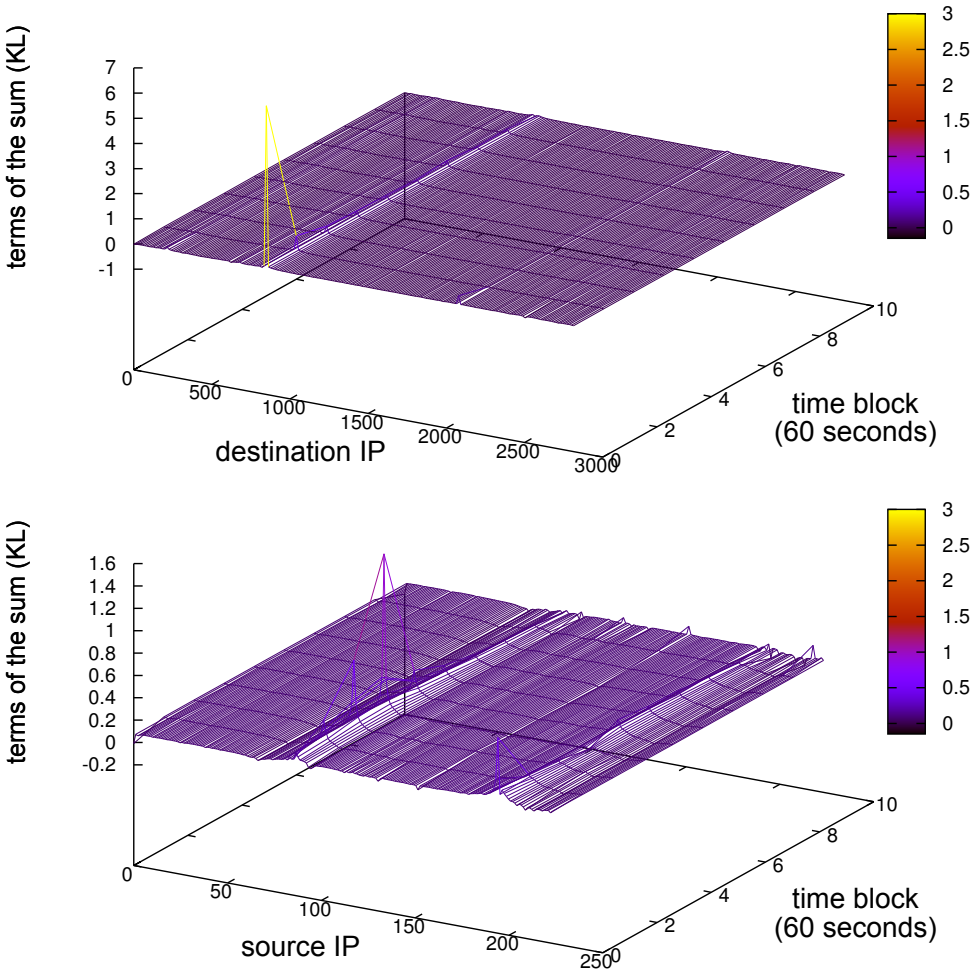


Figure 4.4: Kullback Leibler details for E_2, on source and destination IP address.

In this particular DoS attack, the intensity of malicious traffic is significantly lower than E_1, making the detection more difficult. Indeed, the entropy peaks associated to the attack are not really evident since they are

lower than the false positive of the early morning; nevertheless, the *KL* is still able to detect the anomaly. Again, the deeper representations of the *KL* contributors at the time of the attack (Figure 4.4) show how this metric correctly reveals the attack and characterizes it as a DoS.

E_3 — DDoS attack This event describes a Distributed DoS attack, characterized by a large number of attack sources. In this event the most active host generates only the 0.5% of the flows in the aggregated traffic (table 4.4).

Source	Flows
IP1	0.5 %
IP2	0.5 %
IP3	0.3 %
IP4	0.2 %
IP5	0.1 %
IP6	0.1 %
IP7	0.1 %
...	...

Table 4.4: E_3 - Traffic statistics.

This kind of attack is really different from E_1, where the most active *IP* addresses generate about half of total flows.

Figure 4.5 reports the metric behavior. As in E_1, Rényi distribution seems to generate several peaks associated to non-attack instances. The most significant example can be found around *6:00PM*. The attack started soon after *10:00PM*: both entropy metrics reveal the event and catch its DDoS nature. The abrupt growth of source *IP* entropy line suggests that there was a great amount of diversity in this field. The peak of destination *IP* entropy represents that there is an anomalous variation in the connected endpoints.

Attack dynamic is represented in upper Figure 4.6, where the roughness and quickness of the malicious event causes a jump of the *KL* value. The presence of several new entities drawn by the DDoS attack induces a continuous variation in the source *IP* distribution (see lower Figure 4.6) and, then, causes the *KL* to fluctuate constantly. As opposite to previous

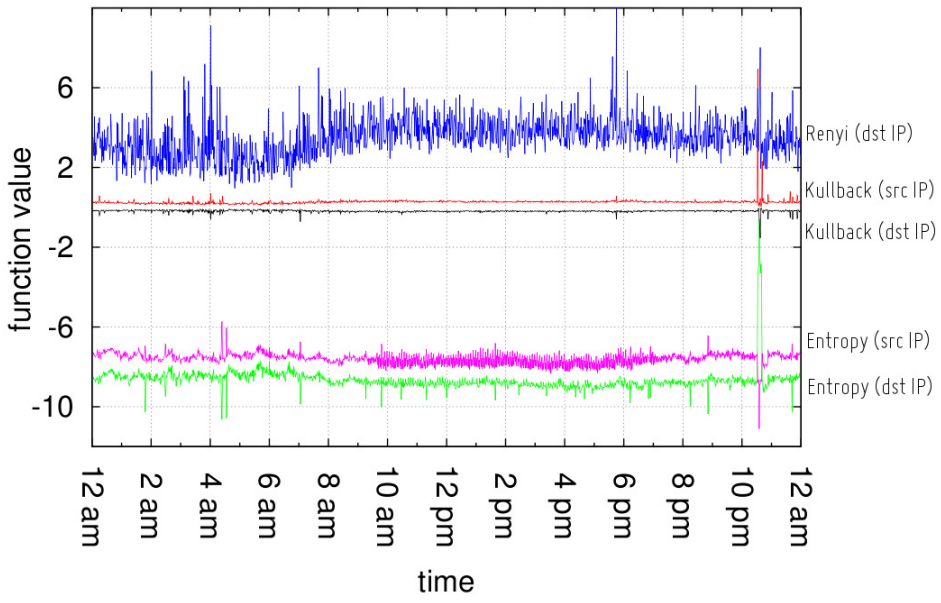


Figure 4.5: E_3 — metrics evaluations for a Distributed DoS attack.

cases, the plot shows that the variation of the *KL* metric is caused by multiple components, that contribute to its final value.

E_4 — **Maintenance jobs** In order to explore how entropy metrics are prone to false positive (see introduction of Section 4.3.1), a deep analysis of maintenance job events has been performed. These events are common to all networks, and consist in backup activities scheduled during the early hours of each days, aimed to reduce network workload and service degradation. Figure 4.7 shows how each *IP* contributes to the final *KL* value. The component's order of magnitude is clearly smaller than the other *KL* detailed graphs. *KL* values, as well as the values of entropy metrics, are sensible to traffic variation. Since the entropy metrics sense destinations (respectively sources) *IP* addresses distribution diversity, they notice a lacks of regularity in the traffic flows and increase their values. On the opposite, *KL* values warns distributions divergence, but the low level of traffic activity attenuates the final result, keeping the value of the metric below suspicious value.

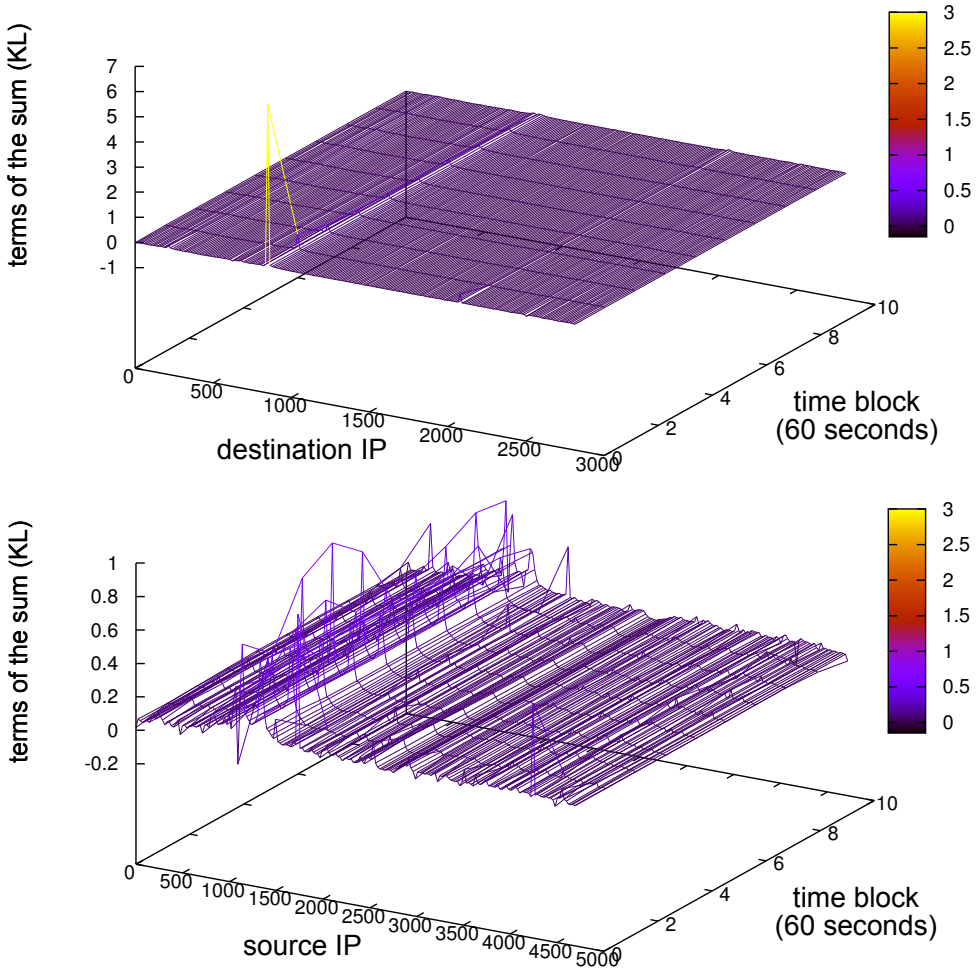


Figure 4.6: Kullback Leibler details for E_3 on source and destination IP address.

4.3.2 Metrics comparison

The experiments show that all the aforementioned metrics are able to detect the traffic alterations, but some of them are prone to a high false positive rate. In particular, it is easy to observe that the Rényi is the more unstable metric: it exhibits many spikes during the whole analysis, making very unsuitable its use for DDoS anomaly detection with netflows. Similarly, the Entropy metric shows many fluctuations, making difficult to find a feature related to DDoS attacks. The Kullback-Leibler (KL), instead, seems to have the more stable trend, showing evident spikes only

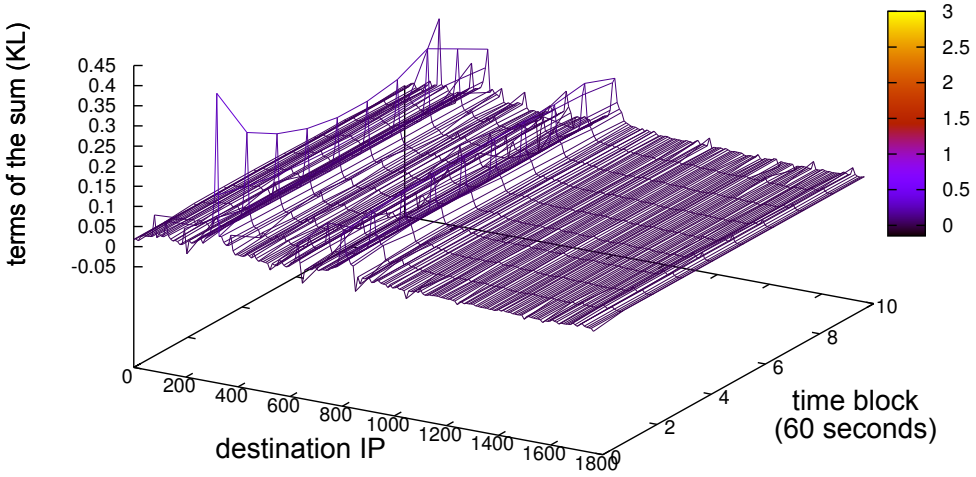


Figure 4.7: E_4 — metrics evaluations for maintenance jobs.

during the attack events. According to other studies like [150], the results showed in this dissertation also suggests that the *KL* is the most suitable information theory metric to detect DDoS.

Furthermore, it is relevant to highlight the difficulty to define a threshold value that could be used to determine if a spike corresponds to an attack or not. Threshold selection is easier with simple metrics (like packet number or packet size), but it seems much harder with statistical metrics [23, 129]. Both the Entropy and Rényi metrics show very unstable values and, in coincidence with some known attacks (like E_2 and E_3), exhibit lower values than the ones obtained during the regular traffic.

Chapter 5

Obfuscation of Sensitive Data in Network Flows

This chapter describes (k, j)-obfuscation, a novel obfuscation technique that provides formal guarantees of security and privacy.

5.1 A novel technique: (k, j)-obfuscation defense

As anticipated in Section 2.3, techniques based on a one-to-one mapping of each IP address in an encrypted value *that is consistent across the whole data set* are ineffective when an adversary gets to know the mapping among some IP addresses and their encrypted value.

On the other hand, mapping the same IP address to different encrypted values in different flows would effectively counteract those attacks, but would result in an excessive loss of information; i.e., it would be tantamount to suppress the IP address fields from released flows. This Chapter exposes a novel technique, named (k, j)*-obfuscation*, that enforces a many-to-one mapping among IP addresses and pseudo-random group-ID values, which are substituted in obfuscated flows to the real IP addresses.

The work described in this chapter is a joint work with D.Riboni, A. Villani, D.Vitali, C. Bettini and L.V.Mancini; it appeared in the 31st Annual IEEE International Conference on Computer Communications (IEEE INFOCOM 2012) [41].

With this solution, each IP address in the released flows is blurred in a set of at least k possible IP addresses.

Note that, with the above solution, an adversary may still be able to identify the real IP address in the group of possible addresses based on the fingerprint of its host. For this reason, (k, j)-obfuscation technique includes a defense against fingerprinting attacks. At first, IP addresses are grouped based on the fingerprint of their corresponding host: IP addresses whose hosts have similar fingerprint are grouped together. Then, the fp-QI values of flows are obfuscated, such that, for each obfuscated flow f^* whose source IP s belongs to an IP-group α, there exist other $j \leq k$ obfuscated flows, whose source IP belongs to α but is different from s, and that are fp-indistinguishable from f^*. This way, even if the adversary knows the hosts' fingerprint, as well as the mapping among IP addresses and group-IDs, he cannot associate each flow to less than j different IP addresses.

5.1.1 Network flows obfuscation

Let be L an original set of network flows, and by L^* the obfuscated version of L released by the data publisher. The fields of the flows include a confidential multi-value attribute $A^p = \{src\_addr, dst\_addr\}$, and a set of other fields $A^i = \{A_1, A_2, \ldots, A_m\}$ that may be used to infer A^p. In particular, as explained in Section 2.3, some flow fields may be exploited to identify A^p based on the hosts' characteristics. In order to characterize those fields, consider the notion of *fingerprint Quasi Identifier (fp-QI)*.

Definition 1 (*Fingerprint Quasi Identifier (fp-QI)*) *A field of a network flow is denoted as a* fingerprint Quasi Identifier (fp-QI) *if its value, possibly combined with external knowledge about the characteristics of the network hosts, can reduce the cardinality of the candidate set for source or destination IP addresses of the flow in L^*.*

Clearly, which flow fields act as fp-QI strongly depends on the external knowledge available to the adversary. In order to state that two flows are

indistinguishable based on the network hosts' fingerprint, consider the following notion.

Definition 2 (*Fingerprint indistinguishability*) *Two network flows are* fp—indistinguishable *if their fp-QI values are identical.*

Given a flow f, and fields A, $f[A]$ is the *projection* of f onto A; for instance, f[*src_addr*,*source_port*] is the pair ⟨source IP address, source port⟩ of f. Flows are obfuscated by a defense function $D()$ before being released.

Definition 3 ((k,j)-obfuscation function) *Let us consider $D : \mathcal{L} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathcal{L}^*$ a partial function that transforms a set of network flows by substituting each IP address with a group-ID, and by possibly obfuscating the values of the other fields of the flows. Each IP address is mapped to its IP-group by a function* group-ID*; this mapping is consistent across the whole set of flows. Let be $f^* \in L^*$ the transformation of $f \in L$ obtained by the application of function D. D is a (k,j)-obfuscation function if, for each set L of network flows, $L^* = D(L,k,j)$ satisfies the following properties:*

- p1: *Each IP-group contains at least k different IP addresses. Formally, for each group-ID g appearing in a flow $f^* \in L^*$, there exists a set A of at least k IP addresses appearing in a flow in L such that, for each $a \in A$, group-ID$(a) = g$.*

- p2: *Each flow f^* is fp-indistinguishable in a set of at least j flows in L^* originated by distinct IP addresses belonging to the same IP-group.*

$D(L,k,j)$ is undefined if the above properties cannot be satisfied; i.e., if L involves less that k different IP addresses (it is impossible to enforce p1*), or if L contains less than j flows (it is impossible to enforce* p2*).*

Table 5.1 reports a summary of the notation used in this chapter.

Table 5.1: Summary of the notation used for the (k, j)-obfuscation algorithm.

fp-QI	fingerprint Quasi Identifier (Definition 1)
src_addr, *dst_addr*	fields for source and destination IP
$f[A]$	projection of flow f onto field A
L (respectively L^*)	original (respectively obfuscated) set of network flows
$D(L, k, j)$	(k, j)-obfuscation function (Definition 3)
k	minimum number of IP addresses in a group
j	minimum number of fp-indistinguishable flows
τ	time granularity used in Algorithm 3

5.1.2 Adversary model

At each release of a set L^*, the goal of an adversary is to reconstruct, with a certain degree of confidence, the source and destination IP addresses of flows in L^*. The considered adversary model is based on the following assumptions:

1. The adversary may observe L^*.

2. The obfuscation function $D()$ is publicly known.

3. The adversary may have external information about the characteristics of the target environment, including the fingerprint of network hosts. For example, the adversary may know the topology of the network to be logged, and the set of services offered by its hosts. This knowledge determines which fields act as fp-QI.

4. The adversary may know in advance where and when the flows will be collected, and may inject flows into the network.

Note that this thesis assumes a powerful adversary, that may perform both fingerprinting and injection attacks. Those assumptions are reasonable. Indeed, some information about the logged network and hosts can be acquired *after* the release of flows; for instance, by scanning the network to locate services. Moreover, in some cases (e.g., if logs are periodically collected and released from a given network), an adversary may also know

in advance the network location and the schedules of flows collection, and send bogus messages to target hosts in the network.

5.1.3 Confidentiality guarantees

As explained in Section 2.3, most network flow anonymization techniques (e.g., Crypto-PAN [52]) are based on a function that maps the space of real IP addresses into a separate space, in order to avoid the disclosure of the real source/destination IPs of the flows. However, it has been recognized that other fields in the flows may allow an adversary to reconstruct the real source/destination IPs of some flows, thus violating privacy constraints. For this reason, several frameworks have been recently proposed to counteract the re-identification of IPs, by obfuscating (i.e., generalizing, substituting, randomizing, ...) the values of those fields. However, it was shown – both theoretically [13] and empirically [18] – that the existing obfuscation methods can be easily defeated by injecting flows in the network according to complex patterns.

The strategy (k, j)-obfuscation is based on the pragmatic assumption that, in the worst case, an adversary may be able to reconstruct the mapping among some IP addresses and their pseudo-ID value by injection attacks[1], and he may know the accurate fingerprint of some network hosts.

In the following section, the confidentiality guarantees enforced by (k, j)-obfuscation will be described, based on different assumptions about the external knowledge available to an adversary.

Defense against knowledge of the IP mapping function

As explained in Section 2.3, if an adversary discovers a mapping between the real and obfuscated IP address in one flow, and the IP address encryption is consistent across the whole set of flows, he can decrypt the same IP in any other flow in which it appears: such mappings can be easily

[1]Even (k, j)-obfuscation does not provide any defense method specifically tailored against injection attacks, its defense against fingerprinting attacks – which is based on generalization – may effectively counteract injection attacks as well.

discovered through injection. As demonstrated by the following theorem, (k, j)-obfuscation counteracts this attack, by ensuring that no less than k different IP addresses are mapped to the same IP-group.

Theorem 1 *Consider a (k, j)-obfuscation function D, a set L of original network flows, its obfuscated version $L^* = D(L, k, j)$, and an obfuscated flow $f^* \in L^*$. Suppose that the (obfuscated) source and destination IP addresses of f^* are α and β, respectively. Suppose also that an adversary got to know the function that maps original IP addresses to their group-IDs. Then, based on the knowledge of that function, he can associate the source and destination IP addresses of f to no less than $k(k-1)$ different pairs of possible addresses.*

Proof According to the property $p1$ of (k, j)-obfuscation, the cardinality of groups α and β is greater than or equal to k. There are two possible cases: either α is different from β, or α is equal to β. In both cases, the uncertainty of the adversary about the original *source IP* of f is at least k: indeed, at least k original IPs may have been substituted with group-ID α in f^*. If β is different from α, the uncertainty of the adversary about the original *destination IP* of f is also greater than or equal to k: hence, he can associate the pair $\langle src\_addr, dst\_addr \rangle$ of f to no less than k^2 different pairs of original IPs. However, if β is equal to α, the uncertainty of the adversary about the original *destination IP* of f is lower. Indeed, the adversary can exclude that the same IP is both the source and destination of the flow, since flows produced and designated to the same host are not logged by the network flow collectors considered in this work. Hence, his uncertainty about the original destination IP is greater than or equal to $k-1$. As a consequence, he can associate $\langle src\_addr, dst\_addr \rangle$ of f to no less than $k(k-1)$ different pairs of original IPs.

Defense against fingerprinting attacks

If an adversary knows the fingerprint of network hosts, he can decrease his uncertainty about the source IP of a flow. The following theorem demonstrates that the (k, j)-obfuscation technique protects against fingerprinting attacks.

Theorem 2 *Consider a (k, j)-obfuscation function D, a set L of original network flows, its obfuscated version $L^* = D(L, k, j)$, and an obfuscated flow $f^* \in L^*$. Suppose that an adversary has accurate information about the hosts' fingerprint. Then, based on this knowledge, he can associate the source IP address of f to no less than j possible addresses.*

Proof According to property $p2$ of (k, j)-obfuscation, f^* is fp-indistinguishable in a set $F^* \subseteq L^*$ of j or more flows having different original source IPs. This means that there exist at least j different hosts, which generated a flow that (after obfuscation) is equal to f^* based on fp-QI values. Hence, exploiting fingerprint knowledge, the adversary cannot associate the source IP of f to less than j different IPs.

Defense against combined attacks

A more powerful threat to consider is when an adversary knows both the IP mapping function, and the hosts' fingerprint. The following theorem demonstrates that (k, j)-obfuscation provides strong protection even under this assumption.

Theorem 3 *Consider a (k, j)-obfuscation function D, a set L of original network flows, its obfuscated version $L^* = D(L, k, j)$, and an obfuscated flow $f^* \in L^*$. Suppose that the (obfuscated) source and destination IP addresses of f^* are α and β, respectively. Suppose also that an adversary got to know the function that maps original IP addresses to their group-IDs, and has accurate information about the fingerprint of network hosts. Then, based on this information, he can associate each pair $\langle \text{src\_addr}, \text{dst\_addr} \rangle$ to no less than $j(k-1)$ different pairs of possible addresses.*

Proof In order to prove this theorem, let us refer to the proofs of Theorems 1 and 2. Let us consider the source IP. As shown in the proof of Theorem 1, by exploiting only the knowledge of the function mapping original IPs in group-IDs, an adversary cannot associate the original source IP of f to less than k different IPs; i.e., the ones that are mapped to α by that function (let be A this set of possible IPs). On the other hand, as shown

in the proof of Theorem 2, by exploiting only fingerprint knowledge, an adversary cannot associate the source IP to less than $j \leq k$ possible IPs (let be B this set of possible IPs). In order to restrict the candidate set of possible IPs, the adversary may intersect A and B. However, property *p2* of (k, j)-obfuscation ensures that all the IPs in B are mapped to the same IP-group; since the source IP of f is mapped to α, this means that also the other IPs in B are mapped to α. Hence, B is a subset of A, and, as a consequence, the intersection of A and B equals to B. Because the cardinality of B is greater than or equal to j, the adversary can associate the source IP of f to no less than j different possible IPs. Since, as shown in the proof of Theorem 1, the uncertainty about the original destination IP of f is greater than or equal to $k-1$, it follows that the adversary cannot associate each pair $\langle src\_addr, dst\_addr \rangle$ to less than $j(k-1)$ different pairs of original IPs.

Defense against linking attacks

In some cases, an adversary may be able to understand that a flow g^* is the response to a flow f^*. Different inferences may be used to *link* request and responses; for instance, by observing that a flow from α to β is immediately followed by a flow from β to α. The following theorem demonstrates that (k, j)-obfuscation is effective even against this kind of inferences.

Theorem 4 *Consider a (k, j)-obfuscation function D, a set L of original network flows, its obfuscated version $L^* = D(L, k, j)$, and two obfuscated flows $f^* \in L^*$ and $g^* \in L^*$. Suppose that the adversary got to know that g^* is the response to f^*. Suppose also that he knows the function that maps original IP addresses to their group-IDs, and has accurate information about the fingerprint of network hosts. Then, based on this information, he can associate the source/destination IP addresses of f and g to no less than $j(j-1)$ different pairs of possible addresses.*

Proof Since the adversary knows that g^* is the response to f^*, he knows that the source IP of f is equal to the destination IP of g, and vice-versa. As shown in Theorem 3, he can perform a combined attack to restrict the

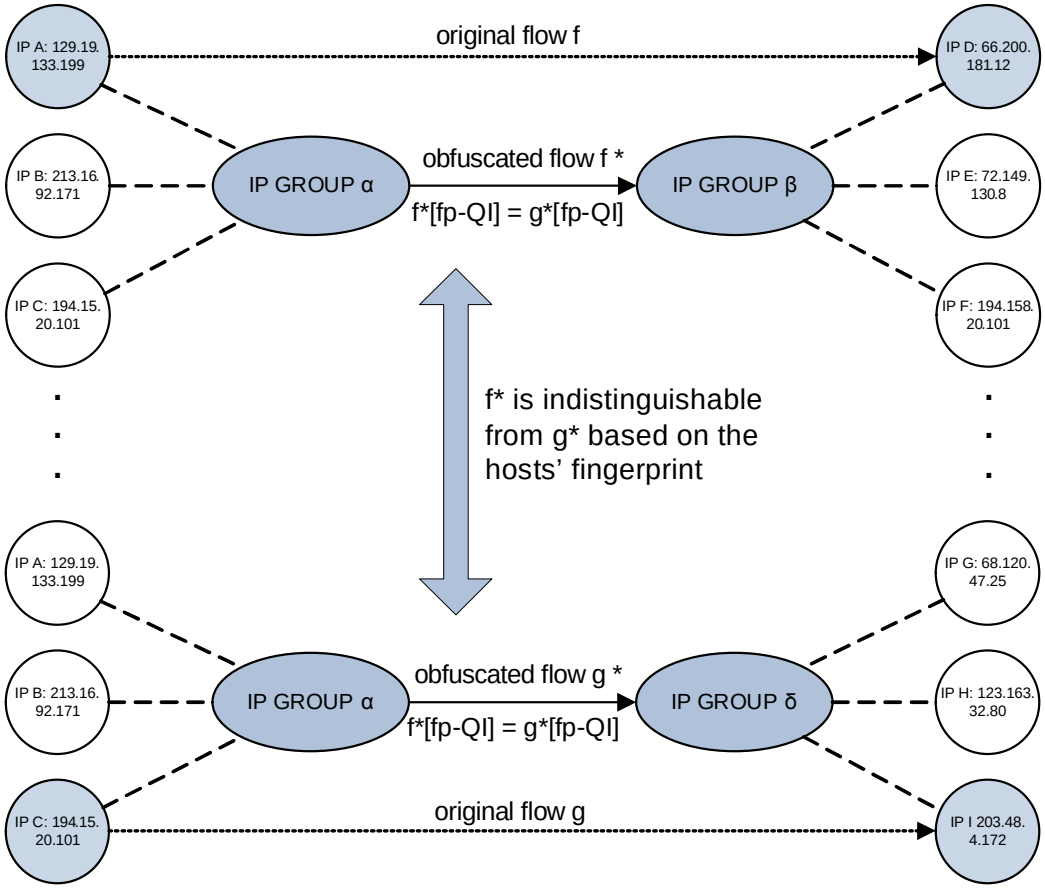


Figure 5.1: Overview of the (k, j)-obfuscation method.

candidate set of source IPs of f to a set A having cardinality greater than or equal to j. He can perform the same attack to restrict the candidate set of source IPs of g to a set B having cardinality greater than or equal to j. Hence, he cannot associate the source/destination IPs of f and g to less than j^2 different pairs of original IPs. Moreover, if the source/destination IPs of f and g are mapped to the same pseudo-ID, the set of candidate source/destination IPs is smaller: its cardinality is $j(j-1)$ (the proof is analogous to the one of Theorem 1).

Input: L: original set of network flows; *fp-QI*: set of fingerprint Quasi Identifiers; k: minimum group size; j: minimum number of fp-indistinguishable flows; τ: time granularity for enforcing fp-indistinguishability.
Output: L^*: set of obfuscated network flows.

```
1  Obfuscate(L, fp-QI, k, j, τ) begin
2      IP-groups 𝒢; IP-group identifiers 𝒢ℐ𝒟 := GroupCreation(L, fp-QI, k)
3      L := SubstituteIPs(L, 𝒢, 𝒢ℐ𝒟)
4      L* := ∅
5      foreach IP-group G_α ∈ 𝒢 do
6          L_α := GetFlows(L, G_α)
7          L*_α := Bucketize(L_α, fp-QI, j, τ)
8          L* := L* ∪ L*_α
9      end
10     return L*
11 end
```

Algorithm 1: Network flow obfuscation algorithm.

5.2 Enforcing (k, j)-obfuscation

Finding the optimal transformation of flows that satisfies (k, j)-obfuscation (i.e., the one that minimizes the generalization of fp-QI values, and the suppression of flows) is a NP-hard problem; indeed, it is well-known that even the basic problem of optimal k-anonymous generalization is NP-hard [100]. For this reason, (k, j)-obfuscation is an approximate algorithm; its pseudo-code is shown in Algorithm 1. At first (line 2), IP-groups are created by executing Algorithm 2 (Section 5.2.1). Then (line 3), the real IPs in network flows are substituted by the identifier of the IP-group they belong to. After initializing the set of obfuscated flows L^* (line 4), for each IP-group, flows generated by the hosts of its IPs are considered, then the fp-indistinguishability property is enforced by executing Algorithm 3 (Section 5.2.2), and it adds the obfuscated flows to L^* (lines 5 to 9). Finally (line 10), the set of obfuscated flows are returned.

Input: L: original set of network flows; *fp-QI*: set of fingerprint Quasi Identifiers; k: minimum group size.
Output: IP-groups: $G_1, \ldots, G_j$; IP-groups identifiers: $GID_1, \ldots, GID_j$.

GroupCreation(L, *fp-QI*, k) **begin**
  set of IP addresses $A := \{f[src\_addr], f \in L\} \cup \{f[dst\_addr], f \in L\}$
  if $|A| < k$ **then return** *null*
  foreach IP address $a \in A$ **do**
    foreach *feature* $\in$ fp-QI **do**
      feature value v_f := GetFeatureValue(L, a, *feature*)
      fingerprint feature vector $\overrightarrow{a}$:= AddFeature($\overrightarrow{a}$, v_f)
    end
    Hilbert index a_H := ComputeHilbertIndex($\overrightarrow{a}$)
  end
  sorted list of IP addresses $\overrightarrow{A}$:= SortOnHilbertIndex(A)
  group index $j := 0$
  for $i := 1$ *to* $|A|$ **do**
    if *(i % k)* = 1 **then**
      if $(i + k) < |A|$ **then**
        $j := j + 1$
        IP-group $G_j := \emptyset$
        IP-group identifier GID_j := *CSPRNG()*
      end
    end
    $G_j := G_j \cup \overrightarrow{A}_i$
  end
  return $G_1, \ldots, G_j$; $GID_1, \ldots, GID_j$
end

Algorithm 2: Fingerprint-based group creation.

5.2.1 Fingerprint-based IP-groups creation

The goal of the fingerprint-based IP-groups creation method is to enforce property *p1* of (k, j)-obfuscation while preserving the quality of obfuscated data. In order to reach this goal, IP-groups are created by grouping together IPs whose hosts have a similar fingerprint (i.e., they originate similar flows), so that fp-indistinguishability can be more easily enforced. The algorithm to group IPs takes as input the original set L of network flows, the set *fp-QI* of fingerprint Quasi Identifiers, and the minimum group size k. It returns the IP-groups and their identifiers. The pseudo-

code of the algorithm is shown in Algorithm 2; its main operations are the following:

1. If less than k IPs appear as source of a network flow in the original set L, it is impossible to create an IP-group of size greater than or equal to k. Hence, in this case, (k, j)-obfuscation cannot be enforced, and the algorithm terminates (line 3 in Algorithm 2).

2. Otherwise, for each IP, the algorithm builds a *fingerprint vector* (lines 5 to 8), in which each dimension corresponds to a statistics (mean, standard deviation, ...) about the values of an fp-QI field of its flows. This vector represents the fingerprint of the host having that source IP.

3. the algorithm maps each fingerprint vector in an integer value by exploiting the Hilbert space-filling curves [20] (line 9). A Hilbert space-filling curve is a function that maps a point in a multi-dimensional space into an integer. With this technique, two points that are close in the multi-dimensional space are also close, with high probability, in the one-dimensional space obtained by the Hilbert transformation. In this case, IPs whose hosts have a similar fingerprint are associated to close Hilbert indices.

4. the algorithm sorts IPs based on their Hilbert index (line 11), and create groups by partitioning IPs in groups of size k based on that order (lines 12 to 22); if the last group contains less than k IPs, it is merged with the previous one. Each group is identified by a value calculated by a cryptographically secure pseudo-random number generator (CSPRNG) function (line 18). Finally (line 23), it returns the IP-groups $G_1, \ldots, G_j$, as well as their identifiers $GID_1, \ldots, GID_j$.

5.2.2 Enforcing fp-indistinguishability

While the fingerprint-based group creation makes it difficult for an adversary to associate flows to IPs in a group based on their fingerprint, it

does not provide any formal privacy guarantee. Indeed, it is still possible that an IP whose host has a very specific fingerprint can be distinguished from the other ones in the same group, and correctly associated by an adversary to its flows. We protect against these attacks by generalizing fp-QI values; i.e., values of those fields that may be exploited by an adversary to identify the source/destination IP of the flow based on the hosts' fingerprint.

The (k, j)-obfuscation defense ensures that, for each obfuscated flow $f^* \in L^*$, whose source IP is $s \in \alpha$, there exist at least other $j \leq k$ flows in L^* whose source IP is $s' \neq s$, $s' \in \alpha$, that are fp-indistinguishable from f^*. When the above condition is satisfied, each flow f^* can be associated to at least j different source IPs with the same confidence: indeed, there exist at least other $j - 1$ source IPs other than s, which generated a flow indistinguishable from f^* based on fp-QI values.

Example 1 *The obfuscation method is graphically illustrated in Figure 5.1. As it can be seen, both source IPs of original flows f and g belong to group α; hence, α is substituted to the real source IPs in obfuscated flows f^* and g^*. Moreover, the values of fp-QI fields are generalized such that they assume the same values in both f^* and g^* (i.e., $f^*[\text{fp-QI}] = g^*[\text{fp-QI}]$). Hence, an adversary performing a fingerprinting attack cannot understand whether f^* has source IP A and g^* has source IP C or vice-versa.*

The bucketization the values of fp-QI fields enforces the fp-indistinguishability property. Bucketization consists in substituting the real value of a field with a multi-set of possible values. For instance, to make fp-indistinguishable a set of three flows whose number of bytes are 250, 400, and 250, respectively, It substitutes the real values in these flows with $\{250, 250, 400\}$.

The bucketization algorithm considers one group of IPs at a time. It takes as input a set L_α of original flows (whose source IPs belong to the same group α), the set of fp-QI fields, the minimum number j of fp-indistinguishable flows, and a time granule τ (for instance, *one minute*). The algorithm considers flows in slots of one time granule at a time, in order

Input: L_α: original set of network flows whose source IP belongs to IP-group α; *fp-QI*: set of fp-QI fields; j: minimum number of fp-indistinguishable flows; τ: time granularity.
Output: L^*_α: obfuscated flows.

Bucketize(L_α, *fp-QI*, j, τ) **begin**
  t_{start} := lowest timestamp of flows in L_α
  t_{end} := highest timestamp of flows in L_α
  repeat
    foreach flow $f \in L_\alpha$ *s.t.* $f[timestamp] \in [t_{start}, t_{start} + \tau)$ **do**
      fp-QI feature vector $\overrightarrow{f} = f[\text{fp-QI}]$
      Hilbert index f_H := ComputeHilbertIndex($\overrightarrow{f}$)
      sorted list of flows $\overrightarrow{F}$:= SortOnHilbertIndex($L_{\alpha,\tau}$)
    end
    group index $i := 1$
    group of IPs $G_i := \emptyset$
    number of distinct IPs in G_i $n := 0$
    for $c := 1$ *to* $|L_{\alpha,\tau}|$ **do**
      if $(\nexists f \in G_i$ *s.t.* $f[\text{src\_addr}] = \overrightarrow{F_c}[\text{src\_addr}])$ **then**
        $n := n + 1$
      end
      $G_i := G_i \cup \{\overrightarrow{F_c}\}$
      if $(n = j)$ **then** $i := i + 1$; $G_i := \emptyset$; $n := 0$
    end
    if $(0 < n < j)$ **then**
      if $i > 1$ **then** $G_{i-1} := G_{i-1} \cup G_i$; $G_i := \emptyset$
      else SuppressFlows(G_i)
    end
    $L^*_{\alpha,\tau} := \emptyset$
    foreach *group of flows* G **do**
      G^* := BucketizeFp-QI-values(G)
      $L^*_{\alpha,\tau} := L^*_{\alpha,\tau} \cup G^*$
    end
    $t_{\text{start}} := t_{\text{start}} + \tau$
  until $t_{start} > t_{end}$
  return $L^*_{\alpha,\tau}$
end

Algorithm 3: Bucketization of fp-QI fields.

to reduce computational and memory costs. This solution is needed when very large sets of flows are considered, as it does in experimental evalua-

tion. The algorithm returns the set of obfuscated flows L^*_α. Its pseudo-code is shown in Algorithm 3; the main steps are the following:

1. At first, it initializes two variables t_{start} and t_{end} with the lowest and highest timestamp of flows in L^*_α, respectively (lines 2 and 3).

2. For each flow f in the original set having source IP belonging to group α and timestamp in the interval $[t_{\text{start}}, t_{\text{start}} + \tau)$, it builds a *fp-QI feature vector* $\overrightarrow{f}$, in which each dimension corresponds to the value of an fp-QI field of the flow (line 6). Algorithm maps each fp-QI feature vector in an integer value f_H by exploiting the Hilbert space-filling curves (line 7), and sort the flows based on their Hilbert index (line 8).

3. It partitions flows in groups based on their Hilbert order, ensuring that each group contains at least j flows having distinct source IPs (lines 10 to 19). If the diversity of source IPs in the last group is less than j, then it merge them with the second-last group (line 21). If diversity cannot be enforced, these flows are suppressed (line 22).

4. For each group, it substitute the fp-QI values of their flows with buckets including the fp-QI values of each flow (lines 25 to 28). the above steps is repeated, for the subsequent time intervals, until t_{end} is reached. Finally, the set L^*_α of obfuscated flows is returned (line 31).

5.2.3 Correctness and computational complexity

Due to the typically large dimension of network flow datasets, the defense algorithm needs to have low computational complexity. The most computationally expensive task of the algorithm is sorting, which is performed both for grouping IP addresses, and for bucketizing fp-QI fields (complexity in the average case is $O(n \log n)$, where n is the maximum between the number of IP addresses appearing in the flows and the number of flows to be made fp-indistinguishable). The calculation of the Hilbert

index is an $O(1)$ operation; it is executed $(n + m)$ times, where n is the number of flows, and m is the number of distinct IP addresses appearing in flows.

Theorem 5 *Algorithm 1 correctly computes a (k, j)-obfuscation function.*

Proof In order to prove the correctness of (k, j)-obfuscation algorithm, it must be demonstrated that, for each set L of original flows, fp-QI fields, k and j parameters, the algorithm returns a set L^* of obfuscated flows that satisfy properties *p1* and *p2* of (k, j)-obfuscation (Definition 3). The above statements can be easily proved by construction. Property *p1* is enforced by Algorithm 2. Note that, if less than k IP addresses appear in the set of network flows, it is impossible to create an IP-group of size greater than or equal to k. In this case, the algorithm terminates without obfuscating flows. However, this situation is unlikely to occur, since logs of network flows typically involve thousands, if not millions, of different IP addresses. Property *p2* is enforced by Algorithm 3.

5.3 Experimental evaluation

The goal of the experiments was to evaluate the role of (k, j)-obfuscation parameters on the utility of obfuscated Netflows. Of course, higher values of k and j determine stronger confidentiality protection but lower utility of obfuscated flows.

5.3.1 Experimental setup

The experiments are supported by the *real* network data described in Section 2.5. The Algorithms 1, 2, and 3 have been implemented using *C* and *Python* programming languages. Experiments were carried out on a workstation with IA64 Core i7 930, 2.80 GHz CPU (4 cores, 8 threads), and 12 GBytes DDR3 1066MHz of RAM, running a GNU/Linux kernel 2.6.32 OS. With this experimental setup, the software were able to obfuscate Netflows collected during an entire working day in a few hours. This is an

acceptable time, since, for most applications, Netflow obfuscation can be performed off-line. An extension of the algorithms to support larger sets of Netflows will be investigated in future works. These experiments consider a model in which the adversary may have an in-depth knowledge of the network hosts' fingerprint. This experiments assume that the fp-QI fields of Netflows are: type of service (*tos*), protocol, TCP flags, number of packets, and dimension in Bytes.

5.3.2 Impact of parameter k: IP address grouping

The first set of experiments was aimed at evaluating the role of parameter k of (k, j)-obfuscation; i.e., the minimum dimension of IP address groups. In order to study the effect of k in isolation, Algorithm 2 has been applied to the original set of Netflows in order to partition IP addresses in groups of dimension greater than or equal to k. Then, each real IP address in original Netflows has been substituted with its corresponding group-ID. The value of k determines the level of obfuscation of IP addresses. Due to the high complexity of the optimal algorithm for (k, j)-obfuscation, and the very large size of the dataset, there are no comparison of (k, j)-obfuscation algorithm with the performance of the optimal one. Instead, the experiments study the impact of (k, j)-obfuscation defense algorithm on obfuscated Netflows using an information theory perspective; in particular, the experiments measure the network flows entropy. Indeed, the network flows entropy evaluation is widespread in traffic analysis, for instance, for traffic anomaly detection [61], and traffic classification [156].

The experiments consider the distribution of IP addresses in flows collected during one-minute long time windows, in order to evaluate its temporal trend, both in original and in obfuscated flows. High values of entropy are correlated to high diversity of the IP address distribution of flows. Hence, this measure is important for network analysis: for instance, a distributed denial of service attack would determine low entropy on destination IP addresses (many flows are directed to the same host), and high entropy on source IP addresses (many different hosts are performing the

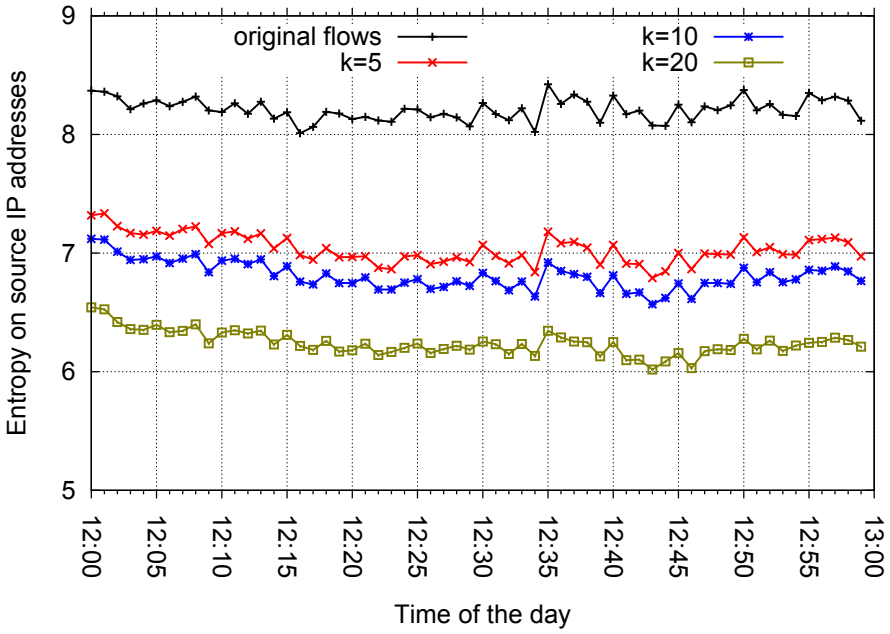


Figure 5.2: Entropy of source IP addresses distribution during one hour.

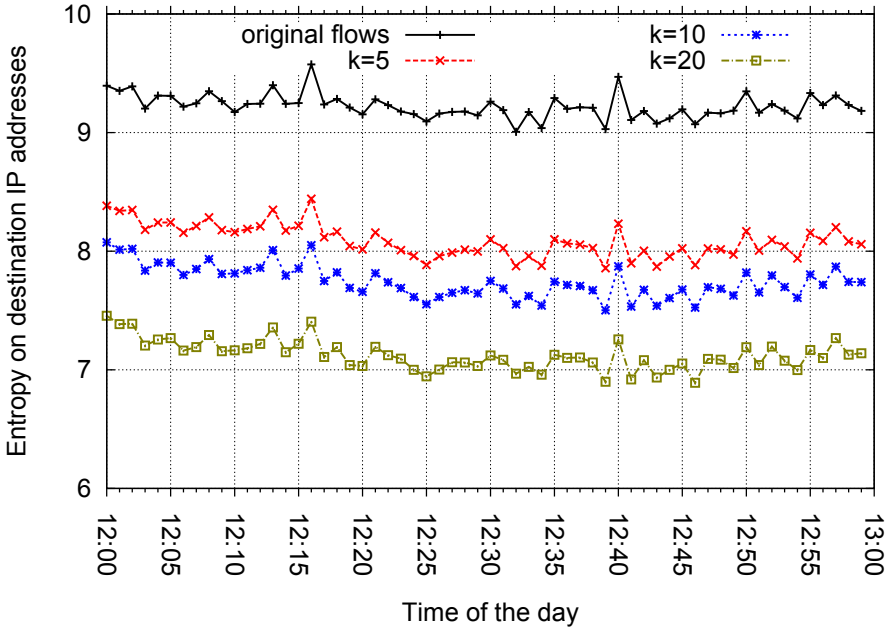


Figure 5.3: Entropy of destination IP addresses distribution during one hour.

attack). Figure 5.2 shows the result during a representative peak hour for Internet traffic (from noon to 1 PM), using the dataset of about 7.5 million

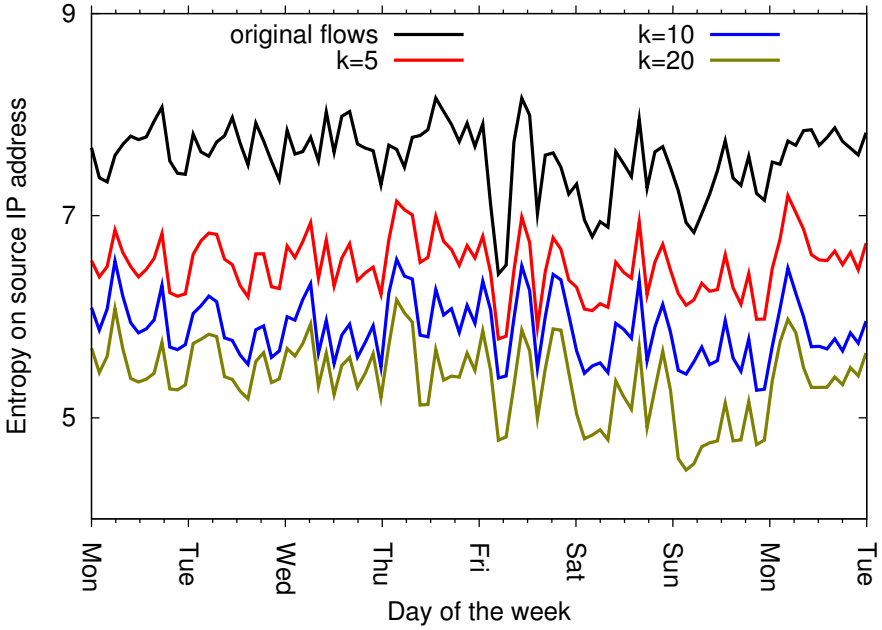


Figure 5.4: Entropy of source IP addresses distribution during 8 days.

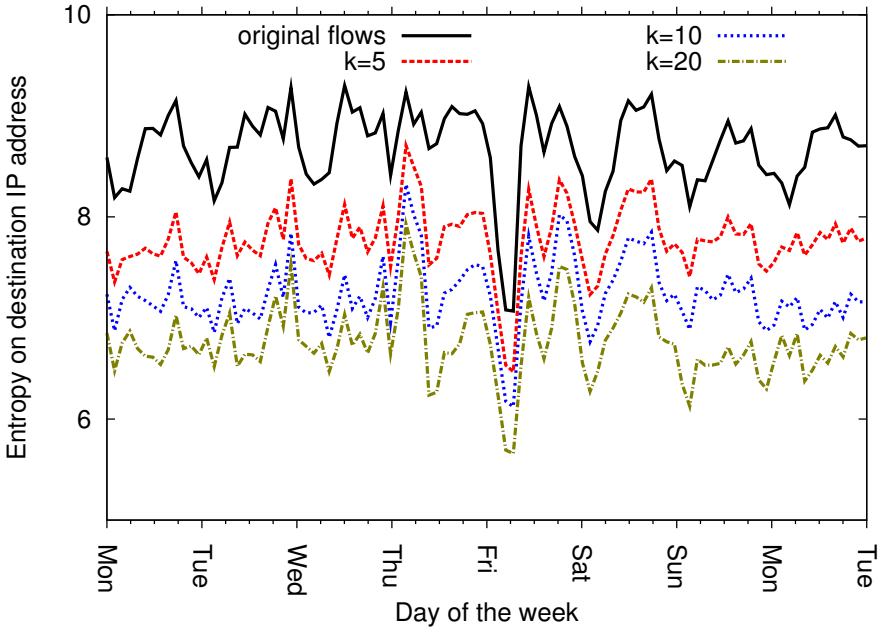


Figure 5.5: Entropy of destination IP addresses distribution during 8 days.

Netflows that were collected by the monitored network described in 2.5. Similar results have been obtained considering other time intervals; for the

sake of readability, figure 5.2 shows a one-hour sample. The experiments include the IP address grouping algorithm with values of k ranging from 5 to 20. As expected, the average value of entropy is inversely correlated to the value of k: indeed, the more IP addresses are grouped together, the less diverse the traffic and, consequently, the lower the entropy. However, algorithms for traffic analysis rely on fluctuations of the entropy value; not on its absolute value. For instance, in [61], traffic anomalies are detected by comparing the entropy in a fine-grained time window (e.g., from 12:00 to 12:01 of the current day) with its expected value (e.g., the entropy calculated on the same minute during multiple days). As it can be seen from figure 5.2, trends and temporal patterns are preserved by the transformation of real source IP addresses in group-IDs; analogous results have been obtained considering destination IP addresses. The above experiments have been repeated considering the distribution of *destination* IP addresses, obtaining analogous results, which are shown in Figure 5.3. These results were confirmed also considering the distribution of IP addresses in the dataset of about 790 million Netflows collected during 8 consecutive days; results are illustrated in figures 5.4 and 5.5.

The above results indicate that (k, j)-obfuscation technique for IP address grouping preserves both traffic diversity and data utility for algorithms based on information theory measures. For the following experiments, the value of k has been fixed to 10, since it provides a good tradeoff between confidentiality protection and data utility.

5.3.3 Impact of parameter j: fp-indistinguishability

The second set of experiments was aimed at evaluating the impact of parameter j on the data quality of obfuscated Netflows. As explained in Section 5.2, in order to reduce computational and memory costs, the algorithm to enforce fp-indistinguishability takes a temporal granularity τ as an additional parameter: Netflows are processed by Algorithm 3 in slots of one time granule at a time. Using shorter time granules demands for less computational and memory resources. However, in some cases, fp-indis-

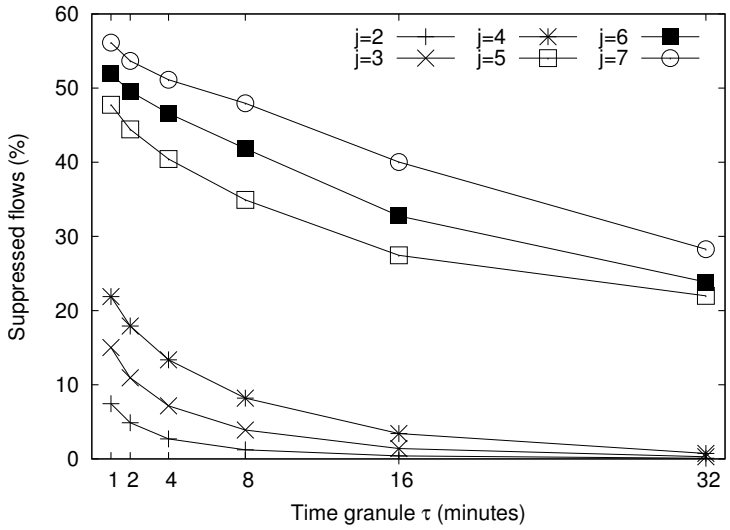


Figure 5.6: Suppressed flows ($k = 10$).

tinguishability cannot be enforced, since there is no sufficient diversity of IP addresses in flows generated during a single time granule. In those unfortunate cases, the algorithm suppresses those flows that cannot be made fp-indistinguishable.

The experiments include also the evaluations of the number of suppressed flows, using different values of j (from 2 to 7) and τ (from one minute to 32 minutes). The computational power of the experimental environment, limit the use the use of values of τ with one hour. As it can be seen in figure 5.6, with low values of j (less than 5), the percentage of suppressed flows rapidly decreases; it is close to zero with τ equal to 32 minutes. On the contrary, with higher values of j, a relevant fraction of flows ($> 20\%$) is suppressed. However, with $k = 10$, small values of j are sufficient to provide confidentiality protection. Figure 5.7 reports the adversary's confidence based on the attacks considered in Section 5.1.3. As it can be observed, with $j = 4$, the confidence of the adversary about the association between a flow and its source and destination IP addresses is below 10%.

The following experiments evaluate the utility of obfuscated Netflows in terms of the precision in answering aggregate queries. For those fields

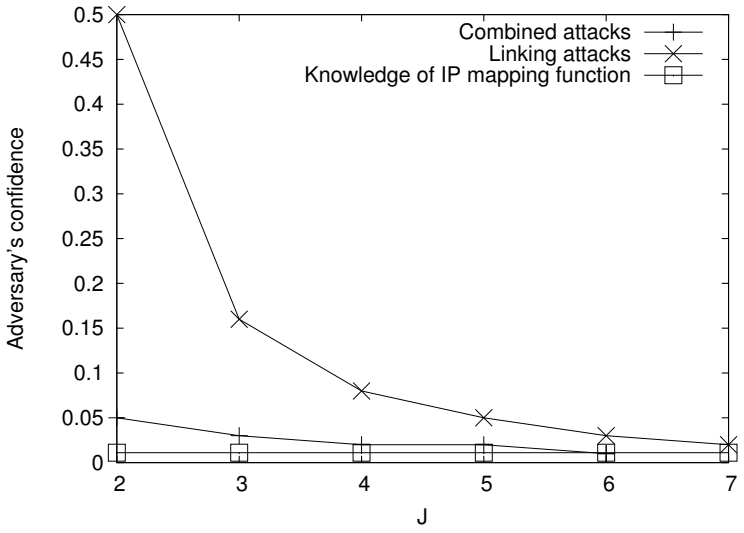
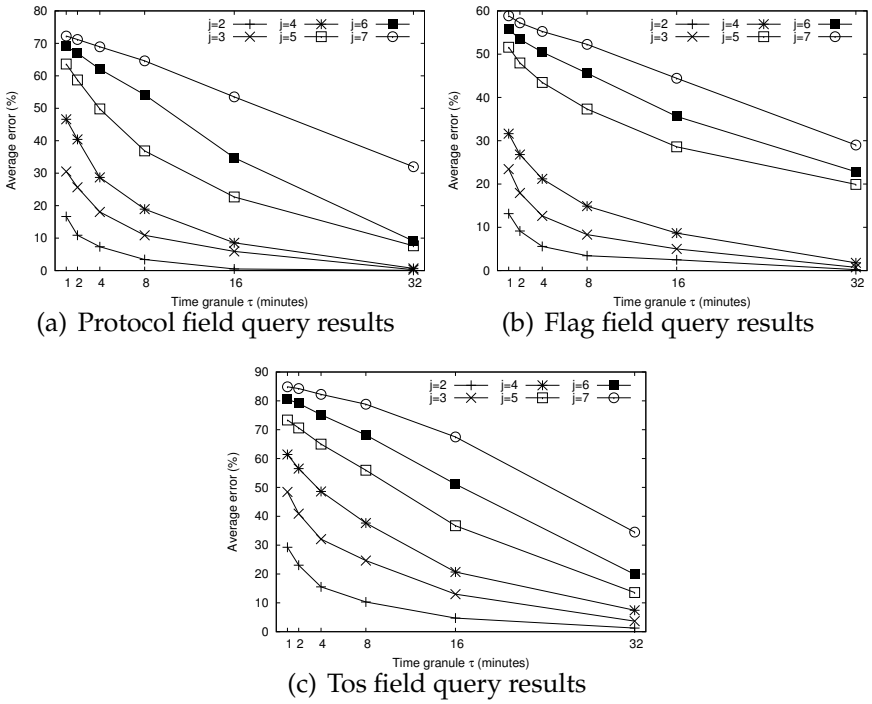


Figure 5.7: Adversary's confidence based on different attacks ($k = 10$).

having numerical domains (number of bytes and number of packets), other experiments executed the queries considering ranges of different selectivity: e.g., *"count the number of Netflows at minute t whose number of packets is between* 200 *and* 300*"*. Each queries have been performed considering each interval of dimension 100 (for the number of bytes) and 5 (for the number of packets), starting from 0, until the maximum dimension of bytes and packets in the dataset of Netflows. For those fields having non-numerical domains (tos, protocol, and TCP flags), the queries have been executed on their specific values; e.g., *"count the number of Netflows at minute t whose protocol is* TCP*"*. Summarizing, all the experiments include queries for each possible value/range, and for each minute in a one-hour time window, for a total of about 120,000 queries. Each query has been evaluated by the error rate described by the following formula:

$$e = \frac{\left|\frac{r}{t} - \frac{r'}{t'}\right|}{\frac{r}{t}}$$

where r (respectively r') is the result of the query on the original (respectively obfuscated) flows, and t (respectively t') is the total number of original (respectively obfuscated) flows.



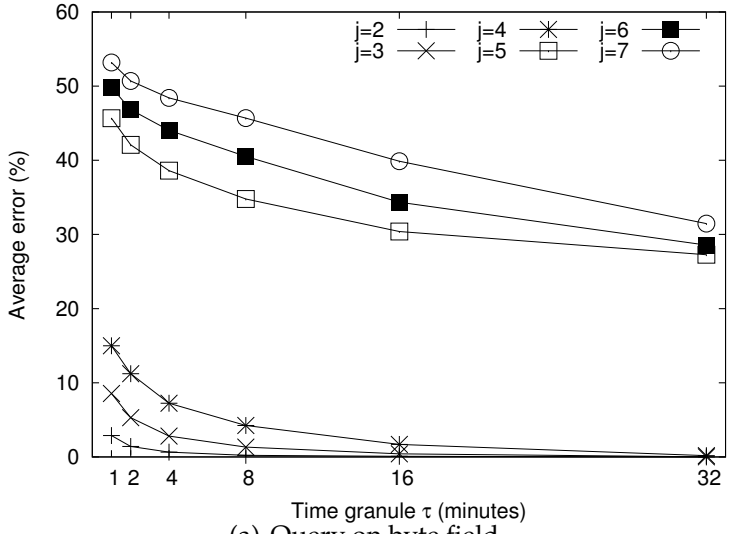
(a) Protocol field query results

(b) Flag field query results

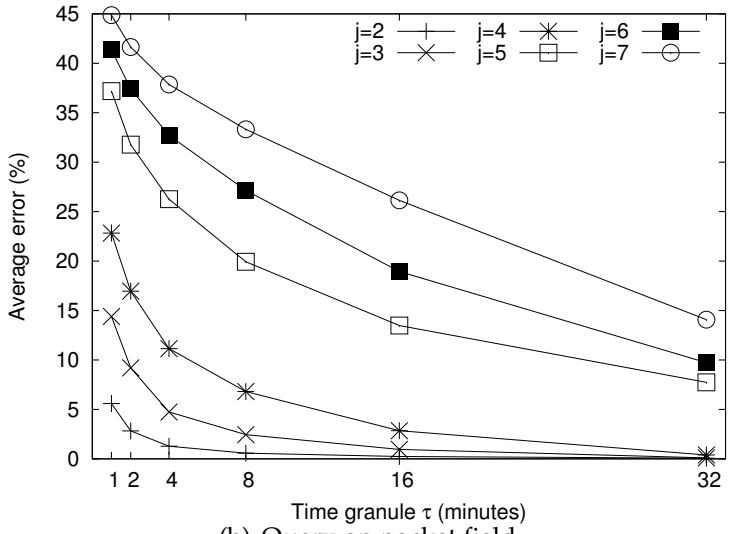
(c) Tos field query results

Figure 5.8: Average error rate for aggregate queries on obfuscated Netflows ($k = 10$).

Figure 5.8 shows the average error rate for different values of j and τ, and the different fp-QI fields having non-numerical domains. Considering flag and protocol fields, with j equal to 4 or less, and τ equal to 16 minutes or more, the average error is below 10%. The average error rate was even smaller with queries on numerical fields (results are shown in Figure 5.9). The tos field presents the larger average error; however, even with that field, the error becomes low using τ = 32 minutes and $j \leq 4$. The average error increases considerably when larger values of j are used; this is due to the large number of flows that must be suppressed to achieve fp-indistinguishability.



(a) Query on byte field



(b) Query on packet field

Figure 5.9: Average error rate for aggregate queries on obfuscated Netflows ($k = 10$).

Chapter 6

Data leakage in Machine Learning models

This chapter evidences a new kind of information leakage in the model of the Machine Learning algorithms.

6.1 Background

Machine learning classifiers are designed to make effective and efficient prediction of "patterns" from large data sets. Many applications have been proposed in the literature, e.g.: [49, 97, 140, 106, 29, 40, 104], and machine learning algorithms pervade several contexts of information technology. ML approaches (such as Support Vector machines, Clustering, Bayesian network, Hidden Markov models, etc.) rely on quite distinct mathematical concepts but generally they are employed to solve similar problems. A machine learning algorithm consists of two phases: *training* and *classification*. During the training, the ML algorithm is fed with a *training set* of samples. In this phase, the relationships and the correlations implied in the training samples are gathered inside the *model*. Afterwards, the model is used during the classification phase to classify and evaluate

The work described in this chapter is a joint work with G. Ateniese, L.V.Mancini, A. Spognardi and A. Villani and it has been submitted for publication, [56].

new data.

ML classifiers are usually able to manage a large amount of data and to adapt to dynamic environments. Their versatility makes them suitable for several important tasks. For example, classification and regression models are employed to analyze current and historical trends to make predictions in financial markets [36, 65, 119], to study biological problems [140], to support medical diagnosis [63, 88, 146], to classify network traffic or detect anomalies [26, 50, 75, 99, 104, 106].

One may think that it is safe to release a classifier, both in hardware or software, since intellectual property laws would prevent anyone from producing a similar apparatus, for example, by copying its code or design principles. However, releasing a trained classifier may be subject to unexpected information leakages that make it possible to produce a competitive product without violating any intellectual property rights.
Let us consider, for instance, a classifier $\mathcal{C}_a$ that is less effective than a classifier $\mathcal{C}_b$ produced by a competitor. The ML algorithms used in $\mathcal{C}_b$ may be publicly available or be inferred through reverse engineering. For example, commercial software products for speech recognition, such as Nuance Dragon Naturally Speaking [108], utilize widely studied Hidden Markov Models. These algorithms, along with their optimizations, are well-understood and quite standard. Thus, the common assumption is that anyone can easily replicate them. In particular, it could be assumed that the training set used for $\mathcal{C}_b$ is *superior*, in the sense that makes $\mathcal{C}_b$ more effective than $\mathcal{C}_a$ even though both implement essentially the same ML algorithms. What makes $\mathcal{C}_b$ better than $\mathcal{C}_a$ is the specific knowledge formed during the training phase, inferred by the training set. For instance, a classifier that makes stock market predictions based on neural network holds its power in the weights at its hidden layer (see Section 6.2.1). But those weights depend exclusively on the training set, hence valuable information that must be treasured.

Thus, it is fair to ask: Is it safe to release a profitable ML classifier? Would selling a software/hardware classifier reveal concrete hints about its training set, uncovering the secrets of its effectiveness and jeopardizing

the vendor?

This chapter shows that a classifier can be hacked and that it is possible to extract from it meaningful information about its training set. This can be accomplished because a typical ML classifier learns by changing its internal structure to absorb the information contained in the training data. In particular, this chapter proposes an attacks that build and train a meta-classifier that can successfully detect and classify these changes and deduce valuable information. However, it could not report on products released by commercial vendors since the missing of the legal permission to hack a proprietary product. Nevertheless, the experiments proposed in this chapter analyzed the same ML algorithms employed by commercial products. For example, experiments consider the HMM-based speech recognition engine of the open-source package VoxForge which is similar to the ones employed by commercial products, such as Nuance Dragon Naturally Speaking. Furthermore, using open-source software makes the experiments easily reproducible by others.

It is important to observe that this study is not interested in privacy leaks, but rather on discovering anything that makes classifiers better than others. In fact, the attack do not care about protecting the elements of the training set. Consider the following example: a speech recognition software recognizes spoken words better than competing products, even though they all implement the same ML algorithms. The training set is composed of commonly spoken words, thus it does not make sense to talk about privacy protection. However, it shows how to build a meta-classifier trained to reveal that, for instance, the majority of training samples came from female voices or from voices of people with marked accents (e.g., Indian, British, American, etc.). Then, we can extrapolate certain hidden *attributes* which are somehow absorbed by the learning algorithm, thus possibly uncovering the secret recipe that makes the speech recognition software stay ahead of the competition.

Therefore the type of leakage focused in this dissertation is quite different than that considered in privacy preserving data mining and statistical databases [1, 24] or differential privacy [9, 44]. Indeed, section 6.4

shows that a system providing Differential Privacy is utterly insecure in the model considered in this dissertation.

6.2 Hacking Machine Learning classifiers

This section focus on Machine Learning algorithms used for classification purposes, such as Internet traffic classifiers and speech recognition systems, or for financial market predictions. The goal is to hack a trained classifier to obtain information that was implicitly absorbed from the elements the classifier received as input.

6.2.1 Artificial Neural Networks

The *Artificial Neural Networks* (ANNs) are a category of machine learning algorithms able to solve a variety of problems in decision making, optimization, prediction, and control, learning functions from real, discrete and vector valued examples. The ANNs obtain good performances in problems where the training data is retrieved by complex sensors, such as cameras or microphones. These algorithms are also resilient to the presence of noise in the dataset. Several types of ANN have been proposed [69], but this thesis will focus on a particular family of ANNs, the ones based on *Multilayer Perceptrons*, and the related *Backpropagation algorithm* , [25], used for their training.

The basic unit of an ANN is the *Perceptron* (or neuron), a unit that takes a vector of real-valued inputs, calculates a linear combination of these inputs and then outputs 1 if the result is greater than some threshold and -1 otherwise. More formally a perceptron can be represented as a function

$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 & \text{otherwise} \end{cases}$$

where let be x_0 always set to 1 to simplify the notation, and let be net $= \sum_{i=1}^{n} w_i x_i$. Observe that $-w_0$ is the threshold value that makes the neuron

to output 1.
A single perceptron represents an hyperplane decision surface in the $n-$ *dimensional* space of instances. This kind of perceptron can only discriminate between linearly separable instances. To overcome this limitation, the sigmoid function σ is used to decide the output value:

$$\sigma(net) = \frac{1}{1 - \exp^{net}}$$

An ANN is a multi-layer network of neurons: a first *input* layer receives the input bits and provides modified inputs to a following layer, that, in turn, elaborates them and feeds a new layer, and so on. The last layer outputs the result of the ANN. The neurons that form the internal layers are called the hidden units. The core function of the network resides in the weight of the hidden units in the internal layers which are set through the backpropagation algorithm. Starting from random weights, the algorithm tunes them using a training set of input-output pairs: the inputs go forward to the network until they become output, while the errors (namely, the difference between actual and expected outputs) are back-propagated to correct the weights. The error is reduced iteratively until a minimal and tolerable error is obtained. The backpropagation of the error is inspired by the principle of gradient descent: in a nutshell, if the weight significantly contributes to the error then its adjustment will be greater.

Let's now consider a simple neural network that has to learn the identity function over a vector of eight bits, only one of them set to 1 (this example is taken from the popular book of Mitchell, [102]). The network has a fixed structure with eight input neurons, three hidden units and eight output neurons. Using the backpropagation algorithm over the eight possible input sequences, the network eventually learns the target function. By examining the weights of the three hidden units, it is possible to observe how they actually encode (in binary) eight distinct values, namely all possible sequences over three bits $(000, 001, 010, \ldots, 111)$. The exact values of the hidden units for one typical run of the backpropagation algorithm are shown in Table 6.1.

Input		Hidden Values		Output
10000000	→	.89 .04 .08	→	10000000
01000000	→	.15 .99 .99	→	01000000
00100000	→	.01 .97 .27	→	00100000
00010000	→	.99 .97 .71	→	00010000
00001000	→	.03 .05 .02	→	00001000
00000100	→	.01 .11 .88	→	00000100
00000010	→	.80 .01 .98	→	00000010
00000001	→	.60 .94 .01	→	00000001

Table 6.1: The weights of the hidden states, taken from Figure 4.7 of [102].

Basically, the hidden units of the network were able to capture the essential information from the eight inputs, automatically discovering a way to represent the inputs. Thus, it is possible to extract the (possibly sensitive) cardinality of the training set by just looking at the trained network.

6.2.2 Classification and Regression Trees

A classification or regression tree (introduce by Breiman et al. in [12] in 1984) are a prediction model which maps observations in a *decision tree*.

The observations $L = (x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ constitute the training set and are used to learn a decision tree. Both classification and regression trees deal with the prediction of a response variable y (let Y be the domain of y), given the values of a vector of predictor variables x (let X be the domain of x). If y is a continuous or discrete variable taking real values (e.g., the size of an object, the number of occurrences of certain events), the problem is called *regression*; if Y is a finite set of unordered values (e.g., the type of Iris plants), the problem is called *classification*.
The training phase produces a tree structure in which the leaves represent the class labels and the branches represent *conjunctions of features* that lead to the class labels of their leaves. Decision trees can be considered as disjunction of conjunctions of constraints on the attribute-values of instances. Each path from the tree root to a leaf corresponds to a conjunc-

tion of attribute tests, and the tree itself to a disjunction of these conjunctions, [102]. Decision trees work better when the target function has discrete output (for example "yes or no") and the data instances are represented by attribute-value pairs. Furthermore, decision trees perform well even when the training dataset contains errors or missing values. These characteristics make decision tree a suitable solution for many classification problems and in a great variety of contexts. For example, in [40] authors propose an Online Voltage Security assessment scheme for energy distribution management which uses decision trees algorithm to avoid the voltage collapse, hence, a possible large scale blackout. In the context of computer and networking security, Edith et al. in [29] proposed a packet classification for large ISP networks, in [104] authors propose and evaluate an intrusion detection system that uses decision trees and SVM also. Podgorelec et al [120], proposes decision tree models in medical decision processes, such as diagnosis of orthopedic fractures, myocardial infarction as well as a strategy of mass vaccination. While these examples show the large flexibility of decision trees, they also highlight the importance of investigating possible information leakages from such models.

The most popular implementation of decision trees is the $C4.5$ algorithm, which is an extended version of the $ID3$ algorithm, both proposed by Quinlan, respectively, in 1986 [121] and 1993 [122]. These algorithms employ a top-down, greedy search through the space of all possible decision trees. In detail, *ID3* algorithm starts the search of the decision tree answering the question: *which attribute should be used at the root of the tree?* Once the root is found, a descendant node of the root is created for each possible value, then the same question is asked recursively at each new node, until: (i) each attribute has been considered in the path through the tree, or (ii) the training examples related to a specific leaf has the same attribute values. The selection of the best attribute in each level of the tree is performed using the concept of *information gain*. In fact, the information gain measures how well a given attribute separates the training examples. Given a collection S of items, for each attribute A, $ID3$ algorithm evaluates

the gain of A with respect to S via the equation:

$$Gain(S, A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v)$$

where $H(S)$ represents the Entropy of the entire dataset and S_v is the subset of S for which attribute A has value v. Roughly speaking, this measure represents the expected reduction in entropy caused by partitioning the training set using the attribute under analysis.

ID3 algorithm performs an inductive learning method since it progressively searches for the decision tree that fits the training examples (the hypothesis space is the set of all possible decision trees). This search strategy has several effects: (i) it inhibits *ID3* in finding alternative decision trees that coherently match the same training set and, (ii) it converges to *locally* optimal solution.

The Information leakages of the decision tree models can be easily detected. Figure 6.1 shows the model obtained by the training phase of the *ID3* algorithm, using the Weka Framework, [145], and the *Iris plants* data set. The Iris dataset is perhaps the best known database in Machine Learning literature. The data set contains 3 classes of 50 instances each, where each class refers to a type of Iris plant (Setosa, Versicolour and Virginica). One class is linearly separable from the other 2; the latter are not linearly separable from each other. It is publicly available form the UC Irvine website [55].

```
petal width <= 0.6 Setosa (50.0)
petal width > 0.6
|  petal width <= 1.7
|  |  petal length <= 4.9 Versicolour (48.0/1.0)
|  |  petal length > 4.9
|  |  |  petal width <= 1.5 Virginica (3.0)
|  |  |  petal width > 1.5 Versicolour (3.0/1.0)
|  petal width > 1.7 Virginica (46.0/1.0)
```

Figure 6.1: Decision tree model obtained using *Weka* Framework, *C*4.5 algorithm and *Iris plants* dataset.

As it can be observed in the figure, decision tree algorithms produce a *plain-text* model, in which many of the characteristics of the used dataset are straightforwardly visible.

Although the Iris Datasets is quite simple and small, the above example is very useful to highlight some concerns. In fact, relevant features of the data set quickly stand out: in the training set no *Iris Setosa* plant with *petal width* greater than 0.6 centimeters can be found, while similar considerations can be made for *Virginica* and *Versicolour* plants. Moreover, other information leakage can be directly obtained, such as the *size* and the *relation* between the model classes. In fact, the number in brackets shows how many *true positive* and *false positive* instances of the training set were involved during the training phase. Similarly, it is possible to see the degree of separation between the different classes of instances: in the above example, the *Setosa* class is clearly independent from the others, while, on the other hand, *Virginica* and *Versicolour* classes have examples that share some attribute values (e.g., *petal width*). Also, notice how this kind of information can be inferred even if the true positive and false positive values were not included into the tree.

As a consequence, it's worth to note that in the case of this classifier, relevant information on the characteristics of the training data is already contained in the model; actually, this is a distinctive feature of all models that are expressed in a sort of plain-text form.

6.2.3 An attack strategy

This section devises a general attack strategy against a trained classifier that can make an attacker able to discover some statistical information about the training set.
Let be the training dataset $\mathcal{D}$ a multi-set where all the elements are couples of the form $\{(\vec{a}, l) | \vec{a} = \langle a_1, a_2, \ldots, a_n \rangle\}$; to simplify, let assume without loss of generality that $a_i \in \{0,1\}^m$, and $l \in \{0,1\}^\nu$. Each training element $\vec{a}$ is represented as a vector of n *features* (the values a_i of the vector) and has an associated classification label l. $\mathcal{C}$ is a generic machine learning classi-

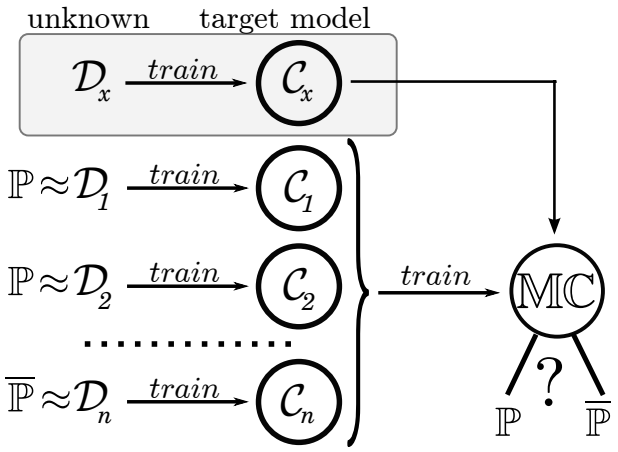


Figure 6.2: Attack methodology.

fier trained on $\mathcal{D}$: it could be an *Artificial Neural Network* (ANN), a *Hidden Markov Model* (HMM) or a simple *Decision Tree* (DT).
Let assume that $\mathcal{C}$ is disclosed after the end of the training phase. This means that the adversary cannot taint $\mathcal{C}$ during the learning process. Instead, the adversary is able to arbitrarily modify the behavior of $\mathcal{C}$ during the classification process. In fact, when $\mathcal{C}$ is *disclosed*, it includes the set of instructions for the classification task as well as the model definition; hence, both the data structures and the instruction sequences are completely in the hand of the adversary. The assumption that the adversary has complete access to the classifier is reasonable since it is possible to extract the plain classifier also from a binary executable through, for instance, dynamic analysis techniques, e.g.: [157].

Each classifier $\mathcal{C}$ can be encoded in a set of feature vectors that can be used as input to train a meta-classifier $\mathbb{MC}$. The set of feature vectors that represents $\mathcal{C}$ are denoted by $\mathcal{F}_{\mathcal{C}}$. For example, in the case of an SVM, the set $\mathcal{F}_{\mathcal{C}}$ would contain the list of all the support vectors of the classifier $\mathcal{C}$.

In Figure 6.2, $\mathcal{C}_x$ is the trained classifier that the adversary wants to examine in order to infer some statistical information about the training set $\mathcal{D}_x$. Let $\mathbb{P}$ be the property that the adversary wants to learn about the undisclosed $\mathcal{D}_x$. Let use $\mathbb{P} \approx \mathcal{D}$ to say that the property $\mathbb{P}$ is preserved by the dataset $\mathcal{D}$. For instance, in the context of medical diagnosis appli-

Input:
$\vec{\mathcal{D}}$: the array of training sets
$\vec{l}$: the array of labels, where each $l_i \in \{\mathbb{P}, \overline{\mathbb{P}}\}$
Output: The meta-classifier $\mathbb{MC}$

```
1  TrainMC(D⃗,l⃗)
2  begin
3      D_C = {∅}
4      foreach D_i ∈ D⃗ do
5          C_i ← train(D_i)
6          F_{C_i} ← getFeatureVectors(C_i)
7          foreach a⃗ ∈ F_{C_i} do
8              D_C = D_C ∪ {a⃗, l_i}
9          end
10     end
11     MC ← train(D_C)
12     return MC
13 end
```

Algorithm 4: Training of the meta-classifier.

cations, $\mathbb{P}$ could be: *the entries of the training set are equally balanced between males and females*. To discern whether $\mathbb{P} \approx \mathcal{D}_x$, the adversary can build a *meta-classifier* $\mathbb{MC}$, that is a classifier trained over a particular dataset $\mathcal{D}_{\mathcal{C}}$ composed of the elements $\vec{a} \in \mathcal{F}_{\mathcal{C}_i}$ labeled with $l \in \{\mathbb{P}, \overline{\mathbb{P}}\}$. The label is assigned according to the nature of the dataset used to train the classifier $\mathcal{C}_i$.

To train $\mathbb{MC}$ the adversary has to build the training set first. For this purpose, the adversary generates $\vec{\mathcal{D}} = (\mathcal{D}_1, \ldots, \mathcal{D}_n)$, a vector of specific datasets in such way that $\vec{\mathcal{D}}$ contains a (possibly) balanced amount of instances reflecting $\mathbb{P}$ and $\overline{\mathbb{P}}$. After this step, he trains the meta-classifier $\mathbb{MC}$ as described in Algorithm 4. The algorithm takes as input the created training sets $\vec{\mathcal{D}}$ and their corresponding labels. It starts with an empty data set (line 3). Then, it trains a classifier $\mathcal{C}_i$ on each created data set (line 5) and gets the representation of the classifier as a set of feature vectors (line 6). Then, it adds each feature vector to the dataset $\mathcal{D}_{\mathcal{C}}$ (line 8). Finally, it trains the meta-classifier using the resulting data set $\mathcal{D}_{\mathcal{C}}$ (line 11).

Next, the adversary uses the meta-classifier $\mathbb{MC}$ on $\mathcal{F}_{\mathcal{C}_x}$ to predict which class l_x the classifier $\mathcal{C}_x$ belongs to. This is already a new form of information leakage since the adversary learns whether the original training data $\mathcal{D}_x$ preserves $\mathbb{P}$ or not.

It is important to remark that with this methodology the adversary extracts *external information*, *not* in the form of attributes of the dataset $\mathcal{D}_x$. These are essentially statistical properties inferred from the relationship among dataset entries. For example, Section 6.3.1 shows how to attack a speech recognition classifier by extracting information about the accent of the speakers. This information is not supposed to be captured explicitly by the model nor it is an attribute of the training set.

To further improve the quality of the classification process, some *filters* can be applied to the set $\mathcal{D}_{\mathcal{C}}$ of models resulting from the training phase. The filters depend on the problem domain and are used to find optimal models for the property $\mathbb{P}$ and get rid of less significant entries. In some cases (as the example in Section 6.3.2), this step can be simply assimilated into the training phase of the meta-classifier. For example, section 6.3.1 will discuss a filter realized with the Kullback-Leibler divergence [92].

6.3 Case studies

This section provides two examples of attacks performed according with the methodology introduced in Section 6.2.3. These case studies probe two complex systems, one of which is largely used by software vendors and research communities. As first example, the focus of the attack is a Speech Recognition system realized by Hidden Markov Models; later, a network traffic classifier implemented by Support Vector Machines has been considered. The experiments are performed using Weka [145].
In each experiment, the meta-classifier $\mathbb{MC}$ is the Decision Tree Classifier (more details on Decision Tree are reported in 6.2.2); the experiment uses the *C*4.5's implementation, namely *J*48 module, included within the Weka framework. Clearly, the attack could be replicated using meta-classifiers

based on other ML algorithms.

The evaluation of the experiments is performed using standard metrics: (1) *recall*, that is the *true positive* rate, and (2) *precision*, that is the ratio of true positive and the total number of positive predictions of the model. Furthermore, (3) *accuracy*, namely the rate of correct predictions made by the classifier over the number of instances of the entire data set, can be easily derived from the confusion matrices in Sections 6.3.1 and 6.3.2.

6.3.1 Hidden Markov Models

Background

A Markov Model is a stochastic process that can be represented as a finite state machine in which the transition probability depends only on the current state and is independent from any prior (and future) state of the process. An *Hidden Markov Model*, introduced in [4], is a particular type of Markov Model for modeling sequences that can be characterized by an underlying process generating an observable sequence. Indeed, only the outputs of the states are observed (the actual sequence of the states of the process cannot be directly observed). One of the most elegant examples to describe HMMs was conceived by Jason Eisner [47]: During the year 2799, contemplate studying the weather in Baltimore (Maryland) for the summer of 2007 by considering the number of ice creams Jason ate every day during that summer. Only using this record (the observable sequence), is it possible to estimate with a good approximation the daily temperature (the hidden sequence). HMMs solve the *sequential learning problem* that is a special learning problem where the data domain is sequential by its nature (e.g., speech recognition problem). In Figure 6.3, a simple model M is represented that can be described by:

- a set of hidden states $Q = q_1, q_2, ..., q_m$

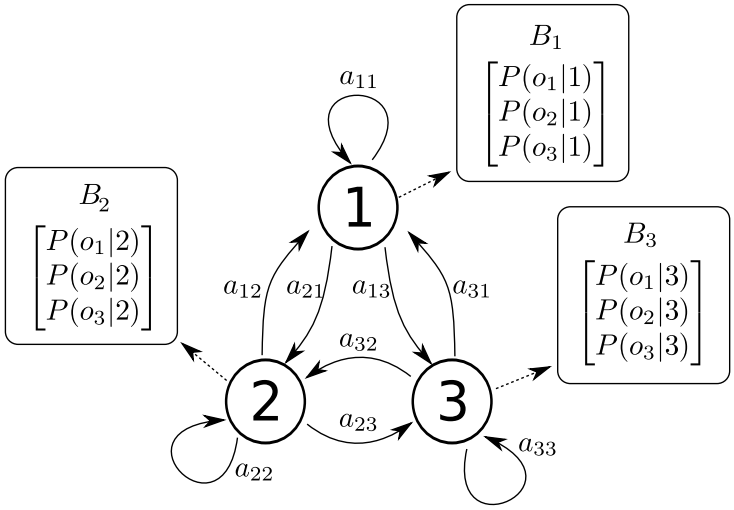


Figure 6.3: An example of Hidden Markov Model with three states..

- a transition probability matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \end{bmatrix}$$

  where the element $a_{i,j}$ represents the probability of moving from state i to state j

- an emission probability matrix $B(m \times n)$, where the element $b_{j,k}$ is the probability to produce the observable o_k from the state j, that is

$$b_{j,k} = B_j(k) = P(o_k|q_j)$$

The HMM model is based on two main assumptions. The first is the Markov assumption, namely that given a sequence $x_1, \dots, x_{i-1}$ of transitions between states, the probability of the next state depends only on the present state:

$$P(x_i = q_j|x_1, x_2, \dots, x_{i-1}) = P(x_i = q_j|x_{i-1})$$

The second is the output independence assumption, namely that given a sequence $x_1, \ldots, x_T$ of transitions between states, where $x_i = q_j$, and the observed sequence $y_1, \ldots, y_T$, the emission probability of any observable o_k depends only on the present state and not on any other state or observable:

$$P(y_i = o_k | x_1, \ldots, x_i, \ldots, x_T, y_1, \ldots, y_T) = P(o_k | q_j)$$

In Figure 6.3, three states (q_1, q_2 and q_3) are shown: the transition probabilities a_{ij}, and, for the three states, the emission probabilities (B_1, B_2, B_3 respectively) of the three observable (o_1, o_2, o_3).

The HMM models are well suited to solve three types of problems: likelihood, decoding and learning, e.g.: [72]. Likelihood problems are related to evaluating the probability of observing a given observable sequence $y_1, \ldots, y_T$, given a complete HMM model, where both matrices A and B are known. Decoding problems call for the evaluation of the best sequence of hidden states $x_1, \ldots, x_T$ that can have produced a given observable sequence $y_1, \ldots, y_T$.

Learning problems consist of reconstructing the two matrices A and B of an HMM, given the set of states Q and one (or more) observation sequence Y. When used for learning, the well-known *Viterbi algorithm* is employed to train the HMM. This algorithm exploits dynamic programming techniques and trellis graphs to provide an efficient way to find the most likely sequence of states generating a given observation sequence. For the training phase, all the transitions and emission probabilities are incrementally adjusted in order to obtain the most probable model. The standard algorithm to perform this task is the *Baum-Welch algorithm*, which derives the *maximum likelihood estimate* of the parameters of the HMM, given a data set of observation sequences.

Exploring the details of these algorithms is out of the scope of this work, but for the sake of completeness it's worth to report that the classification problem in the context of HMM is reduced to finding the solution

to the following system:

$$\arg\max_{X} \left\{ a_{x(0),x(1)} \prod_{t=1}^{T} a_{x(t),x(t+1)} b_{x(t)}(o_t) \right\}$$

where X is a specific state sequence of the model.

HMM for speech recognition

This section describes the attack to the HMM in the specific case of *Speech Recognition Engines* (SRE). The *Speech Recognition* (SR) is the process of converting a sound recorded through an acquisition hardware to a sequence of written words. The applications of SR are manifold: dictation, voice search, hands-free command execution, audio archive searching and so on; the predominant technology used to perform this task is the HMM [71], many tools are nowadays available, [81, 82].

The experiment exploited methodology described in 6.2.3 to verify whether the HMM was trained with a *biased* training set: according to the methodology described in 6.2.3, it is possible to detect, with high confidence, whether the HMM was trained *only* with people with the same accent.

There exist several types of speech recognition engines. Some consider each word simply as a sequence of phonemes ignoring the context in which a phoneme appears; others try to obtain a more accurate detection, by taking into account also the probabilities that a particular sequences of phonemes can appear in the target language. One of the main reasons that makes this process challenging is that there are many variations for the same word depending on the speaker or the environment. For instance, pronunciation, speed of speaking, or characteristics of the acquisition hardware can generate many perturbations to the signal associated to the same spoken utterance.

To recognize a speech, SREs require two types of input:

- an *Acoustic Model*, which is created by taking speech audio files, i.e.,

the *speech corpus*, and their transcriptions, and combing them into a statistical representation of the sounds that make up each word;

- and either a *Language Model* or a *Grammar File*. Both describe the set of words that the statistical model will be able to classify. However, the first model contains the probabilities of sequences of words, while the second contains a set of predefined combinations of words. For the sake of the experiment, it uses only the Language Model.

The SRE workflow is the following. An unknown speech waveform is captured by the acquisition hardware, the Pulse Code Modulation provides the digital representation of the analogical audio signal. This bit stream is now converted in *mel-frequency cepstral coefficients* (MFCCs), namely a representation of the short-term power spectrum of sounds. The MFCCs are the *observables* of a Hidden Markov Model that changes state over time and that generates one (or more) *observables* once it enters into a new state.

In this scenario, the states of the HMM are all the possible sub phones of the language while the transition matrix contains the probability for each sub phone to cycle over itself or to move to the next sub phone. The emission probabilities are the probability to observe a certain MFCC from each sub phone. The only possible transitions between the states of each phones are to themselves or to successive states, in a left-to-right fashion; the self-loops makes it possible to deal with the variable length of each phone with ease. Both transition and emission probabilities are built using the *Viterbi* algorithm, [64], over a large speech corpus.

Since the MFCC files are vectors of real-valued numbers, they are approximated by the multivaried Gaussian distribution (note that the probability to have exactly the same vector would be nearly 0). For any different state (i.e., sub phone), each dimension of the vector has a certain *mean* and *variance* that represent the likelihood of an individual acoustic observation from that state.

For the sake of the experiments, it has been built an Hidden Markov Models using the *Hidden Markov Model Toolkit* (HTK), [155]. HTK consists

of a set of library modules and tools available in C. The HTK toolkit provides a high level of modularity and is organized through a set of libraries with functions (e.g., *HMem* for memory management, *HSigP* for signal processing ,...) and a small core.

The MFCC files were gathered from the VoxForge project [144], the most important speech corpus and acoustic model repository for open-source speech recognition engines. Moreover, each speech file released by VoxForge is associated with several categories such as gender, age range, and pronunciation dialect. The aim of the experiment is to extract this information, which is implicitly correlated with the contents, even if it does not appear as an attribute in our data set.

Attack description

The main objective of this attack is to build a meta-classifier for the following property $\mathbb{P}$: *the classifier was trained only with people who speak an Indian English dialect*. It's important to emphasize that this is *external information* as introduced in Section 6.2.3: the speech dialect is *not* explicitly used during the training process, but in practice it influences the output of the classifier.

The first part of the experiment describes the encoding of the HMMs; next, it describes the decision tree of the meta-classifier; finally, it presents an improved version of the classifier that uses a *filter* to improve the classification.

To carry out the attack, the experiments retrieved 11,137 recordings from the VoxForge corpus. In particular, the experiments consider only the MFCC files in the English language. Each track comes with a form containing some meta-information (e.g., gender, age, pronunciation dialect). The corpus have been partitioned according to this meta-information, namely by the same pronunciation dialect. Starting from this partition, it has been created $\vec{\mathcal{D}}$ according to the rule defined in Section 6.2.3. Then, each classifier $\mathcal{C}_i$ have been trained as described in Algorithm 4.

Each classifier C_i, is represented in the HTK toolkit by an ASCII file

containing an HMM for each phoneme belonging to the English language. Each HMM is composed of: a transition probability matrix $A(n \times n)$ which describes the transition between hidden states and the two vectors $M = (\mu_1, \mu_2, \ldots, \mu_m)$ and $V = (\sigma_1, \sigma_2, \ldots, \sigma_m)$ that are respectively mean and variance of the output probability distribution from a given hidden state (see Sections 6.3.1). The experiments took the default HTK values (i.e., $m = 25$ and $n = 5$). To encode a single HMM, it is suffice to choose to focus only on the output distributions, that is, the couple of vectors (M, V). The idea is that all these values are initialized in the early steps of the training, according to a mean computed over the entire MFCC dataset: since all the values are iteratively refined through the HTK toolkit, then it could happen that these values are correlated in some way with the voices of the learning set and, by extension, with the pronunciation dialects. For this reason, the feature vector $\vec{a} \in \mathcal{F}_{\mathcal{C}}$ are:

$$\vec{a} = (ph, \mu_1, \mu_2, \ldots, \mu_m, \sigma_1, \sigma_2, \ldots, \sigma_m, l_i)$$

where ph is a string value representing a phoneme, $\mu_1, \mu_2, \ldots, \mu_m$ and $\sigma_1, \sigma_2, \ldots, \sigma_m$ are the output probability vectors and $l_i \in \{Indian, not\ Indian\}$ is the label of the current row. It is important to notice that this encoding gives a row in $\mathcal{D}_{\mathcal{C}}$ for each phoneme of the acoustic model. The training set was composed of $5,420$ tuples equally balanced over the two classifications considered for this experiment. The test set was composed of $1,016$ instances: 774 of these are classified as *not Indian* and the remaining 242 are classified as *Indian*. The training ended up with a very complex meta-classifier: the decision tree was composed of more than 811 nodes with 610 leaves.

Indian	not Indian	classified as
220	22	Indian
72	702	not Indian

Table 6.2: Speech recognition attack: the confusion matrix of the meta-classifier.

Table 6.2 reports the confusion matrix obtained from this experiment;

in fact, the confusion matrix shows how correctly a classifier assigned the labels to the elements of the input set. The *not Indian* classifiers are correctly classified with precision of 0.97 whereas the *Indian* classifiers are recognized with precision 0.75. Specifically, recall of *Indian* class equals 0.909 and recall of *not Indian* equals 0.907.

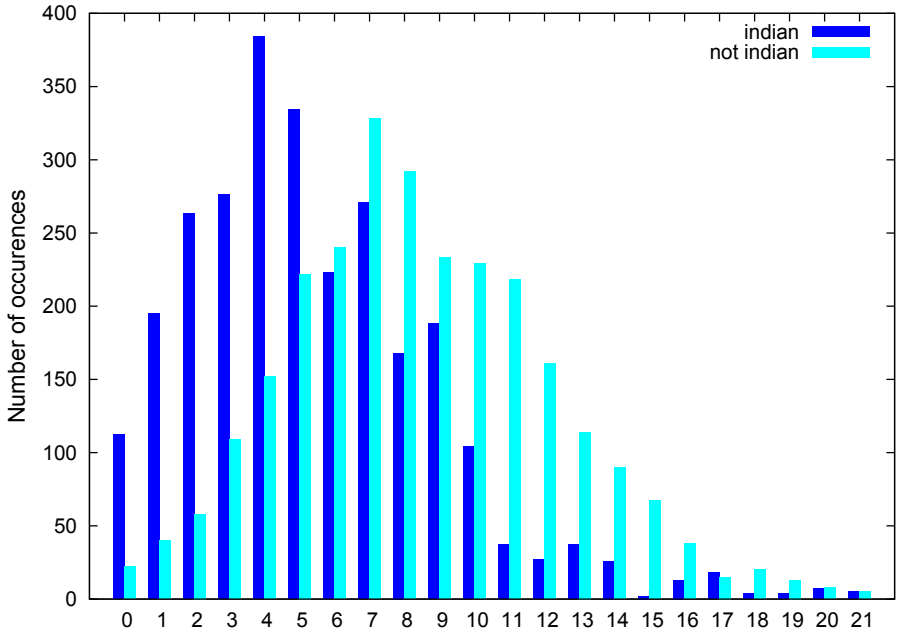


Figure 6.4: Speech recognition attack: frequency of the values of σ_2 for all phonemes in the training data of the meta-classifier.

One of the most interesting feature provided by the *C4.5* algorithm consists of the order in which the attributes decision tree appear. In fact C4.5 puts the most representative attributes at the higher level of the tree. In the experiment, one of the most representative node is σ_2. The frequencies of each value of σ_2 in the training data of the meta-classifier are represented in figure 6.4. It is easy to notice that the mean values of each distribution are considerably shifted and can be easily recognized with respect to the class. The meta-classifier is very effective in catching those differences; hence, as the experiments show, it obtains very good performances.

To further improve the quality of $\mathbb{MC}$, a *filter* to the training set $\mathcal{D}_{\mathcal{C}}$ has been applied. The goal was to extract the phonemes that *better differentiate* the language dialect. To perform this task, the experiments employed the

Kullback-Leibler (KL) divergence between the output probability distributions of the models. The KL divergence is defined as follows:

$$D_{KL}(P||Q) = \sum_i P(i) log \frac{P(i)}{Q(i)} \tag{6.1}$$

A low D_{KL} value means a high similarity of the two probability distributions, while on the other hand, high divergence values correspond to an inferior similarity. This means that the phonemes with the highest divergence are the ones which better discriminate the Indian accent from others. Since the output probabilities follow a Normal distribution, the KL divergence is computed with the following equation:

$$\mathfrak{D}_{KL}(X_i||X_j) = \frac{(\mu_i - \mu_j)^2}{2\sigma_i^2} + \frac{1}{2}\left(\frac{\sigma_i^2}{\sigma_j^2} - 1 - \ln\frac{\sigma_i^2}{\sigma_j^2}\right) \tag{6.2}$$

where $X_i \sim N(\mu_i, \sigma_i)$ and $X_j \sim N(\mu_j, \sigma_j)$.

The experiment considers 100 different training sets without *Indian* records, obtaining the relative acoustic models $\vec{\mathcal{C}} = (\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{100})$. Then, it has been built the reference learning set containing only *Indian* records, and the relative acoustic model $\mathcal{C}_r$. Then, it has been compared the distance between the output probability distributions of $\mathcal{C}_r$ with every $\mathcal{C}_i \in \vec{\mathcal{C}}$, obtaining the summed value of the divergence. Since the same phoneme state has 25 possible output distributions, it is suffice compute the mean distance value across all the distributions. Finally, just the five phonemes with the highest divergence are considered and used to rebuilt $\mathbb{MC}$.

Indian	not Indian	classified as
169	6	Indian
2	137	not Indian

Table 6.3: Speech recognition attack: the confusion matrix of the *filtered* meta-classifier.

Table 6.3 shows the confusion matrix of the *filtered* classifier. The new results are noticeably improved: the precision for the *not Indian* class is

0.98 as before whereas the precision for the *Indian* class is increased to 0.95. (Specifically: recall *Indian*: 0.986 and recall *not Indian*: 0.966.)

Also, the size of the decision tree has dropped down significantly: the resulting decision tree is composed only of 21 nodes with 11 leaves.

6.3.2 Support Vector Machines

Background

Support Vector Machines (SVM) are supervised learning methods related to *statistical learning theory* and first introduced by Boser et al. in [11]. SVMs are largely used for classification and regression analysis. In their basic form, SVMs are first trained with sets of input data classified in two classes and are then used to guess the class for each new given input. This aspect makes SVM a *non-probabilistic binary linear classifier*. Support Vector classifiers are based on the concept of *separating hyperplanes*, that are the hyperplanes in the attribute space that defines the decision boundaries between sets of objects belonging to different classes.

During the training phase, the SVM receives a set of labeled examples, each of them described by n numerical attributes (*features*) and thus represented as a set of points in a n-dimensional space. For the sake of simplicity, the following example briefly introduces how an SVM works with data represented by two attributes and mapped into two classes. The entry i of the training dataset is represented by a 2-dimensional vector $x_i = \langle x_{i1}, x_{i2} \rangle$ and belongs to one and only one class y_i:

$$\begin{aligned} (y_1, x_1), (y_2, x_2), \ldots (y_n, x_m) \\ y_j \in -1, 1 \end{aligned} \tag{6.3}$$

Let suppose that the training data is linearly separable, namely there exists a vector w and a scalar value b such that:

$$\begin{aligned} w \cdot x_i + b \geq 1 \, if \, y_i = 1, \\ or \\ w \cdot x_i + b \leq 1 \, if \, y_i = -1 \end{aligned} \tag{6.4}$$

In order to deal with sets that are not linearly separable, the training vectors x_i can be mapped into a higher dimensional space by the function ϕ, the so called *kernel function*: many kernel functions have been proposed, but the most used are linear $K(x_i, x_j) = x_i^T x_j$, polynomial $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma \geq 0$, radial basis function, RBF, $K(x_i, x_j) = exp(-\gamma \| x_i - x_j \|^2), \gamma \geq 0$ and sigmoid $K(x_i, x_j) = tanh(\gamma x_i^T x_j + r)$. The Support Vector classifier finds the optimal hyperplanes that separate the training data with a maximal *margin* in this higher dimensional space; formally it resolves the system of equations:

$$y_i(w_0 \cdot x + b_0) = 0 \tag{6.5}$$

It must be pointed out that, thanks to the nature of the training algorithm adopted by SVM, the solution of equation 6.5 can be obtained at a reasonable computational cost regardless of the kernel function adopted. Intuitively, a good separation is achieved by the hyperplane that has the largest distance - or *margin* - between the nearest training data points of different classes: these points are called the *support vectors*. Roughly speaking, the larger the margin, the lower the generalization error of the classifier.

It is easy to notice how the functional margin points determine the hyperplane of separation. This information is trivially featured by the attribute values in the training sets. Furthermore, It is worth to highlight that SVM can disclose more information when several classifiers trained with different kernel functions are provided. Since a trained SVM is represented by a set of weights and a subset of the training sample, it is not easy to obtain useful information on the characteristics of the complete training set

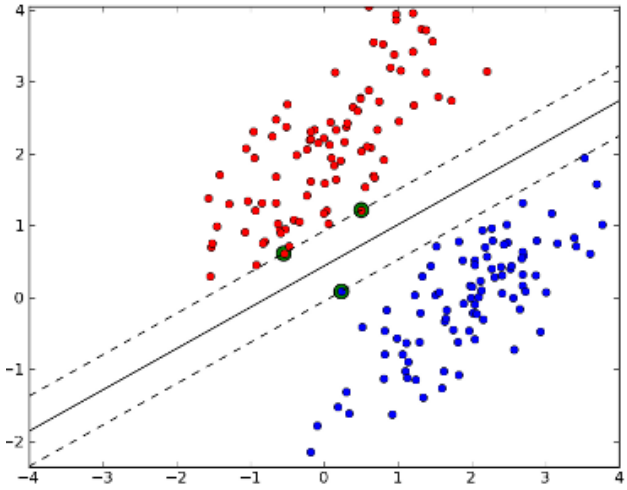


Figure 6.5: Support Vector Classifier: graphical interpretation and functional margin for a 2 class separable dataset.

directly from the SVM representation.

SVMs generated a significant research activity which extends across the limits of data mining area. Although SVMs were initially introduced to solve pattern recognition problems in an efficient way, [17]), nowadays they are suitable in several contexts. In fact, SVMs are used for intrusion detection and anomaly detection [75, 66, 26], or as part of complex systems for similar tasks [104, 103]. Other authors propose SVM-based systems for privacy-critical tasks, such as cancer diagnostic [63, 35], text categorization [70], or face recognition [62].

SVM for network traffic classification

As shown by the extensive literature on this topic [50, 49, 106, 2], network traffic classification is commonly realized by means of Machine Learning algorithms, like K-Means, HMM, decision trees, and SVM.

In order to evaluate the information leakage of SVM classifiers, the experiment set up a simple Network Traffic Classifier able to distinguish between DNS and WEB traffic. In particular, it has been considered an SVM classifier based on the *SMO* module (Sequential Minimal Optimization [118]) of the Weka framework.

The experiment uses a *real netflow* dataset, gathered by a national tier 2 Autonomous System (refer to section 2.5 for more details).

The classifier was trained using a balanced set of netflows of WEB and DNS traffic. It is worth noting that the WEB data set includes several traffic patterns. Namely, it contains the flows directed to national newspapers, advertising websites, and the Google search engine website.

During the training phase of the experiment, they have been used all the fields of the netflow entries, except the source and destination IP addresses of the tracked connections. In the literature there are examples of SVM Classifiers for traffic detection [50] able to distinguish a greater variety of network protocols; the methodology used in the experiment is similar, and can be considered appropriate to highlight the information leakage issues that are the target of this dissertation.

Furthermore, note that the classifier performs good results with high *accuracy* and *precision*: indeed, WEB and DNS connections have well-separated traffic patterns, producing well-spaced support vectors.

Attack description

The experiment investigates whether it is possible to extrapolate the type of traffic used during the construction of the SVM model. For example: *Can we infer whether Google web traffic was used in the training samples?*

The attack proceeds by creating several ad-hoc data sets with well-defined statistical properties and use them to build the meta-classifier $\mathbb{MC}$. Namely, it has been created 70 ad-hoc data sets, selecting 20.000 flows of network traffic, distinct from the original training set.

While all 70 classifiers were trained with a non-specific DNS traffic, the first half of the classifiers were trained using WEB traffic directed only to Google search engine (property $\mathbb{P}$). For the remaining 35 classifiers, it has been used WEB traffic without any netflow directed to Google search engine (property $\overline{\mathbb{P}}$).

Each classifier was trained using a *polynomial kernel function* of degree 3 and was encoded by the list of the support vectors it contains, namely

a set of points $(y, \vec{x})$ in the $n-$dimensional space ($\vec{x} = \{x_1, x_2, \ldots, x_n\}$). The training samples of the classifier $\mathbb{MC}$ are composed of all the support vectors of the 70 classifiers, labeled according to the property $\mathbb{P}$ or $\overline{\mathbb{P}}$ used for training:

$$\mathcal{D}_{\mathcal{C}} = \bigcup_{\mathcal{C}_i} \{(y, \langle x \rangle, label)\}$$

The experiment evaluates the performance of $\mathbb{MC}$ using the cross validation strategy, a method that divides the data into k mutually exclusive subsets (namely, the "folds") of approximately equal size. With the cross validation, the accuracy estimated is the average accuracy for the k folds.

Google	not Google	classified as
2312	101	Google
92	2786	not Google

Table 6.4: Internet traffic classifier attack: the confusion matrix of the meta-classifier.

Table 6.4 summarizes the experiment results: with respect to the *Google* class, results present a *precision* of 0.954 and a *recall* of 0.932. On the other hand, it correctly classify *not Google* instances with a *precision* of 0.943 and a *recall* of 0.962.

As in the example with the HMMs, the experimental results show that we were able to build an effective meta-classifier that infers whether the training set given as input includes also a specific type of traffic.

6.4 Differential Privacy

This section shows that Differential Privacy [44] is ineffective against the attack strategy presented in this chapter. More specifically, the information leakage presented with the previous experiments, sits outside the adversary model considered by differential privacy.

Differential privacy, [44, 1, 9], protects against unintentional disclosure of potentially sensitive information related to a single record of a database D. In other words, differential privacy maximizes the accuracy of queries

from statistical databases and, at the same time, minimizes the ability to identify single records. To protect the privacy of database records, differential privacy opts for basically three approaches:

1. The first is to obfuscate the original database D and transform it into D'. This strategy is completely ineffective in the model assumed, since D' is the database actually used during training and it is exactly what the adversary in the defined model is after. That is, the adversary is not interested in D, or any of its records, but it is rather eager for any information on D', i.e., anything that is the result of the transformations applied by differential privacy.

2. Another approach is to train a classifier and then add noise to the output. This is also ineffective since, in this model, the adversary has complete access to the classifier and could just disable the instruction that adds noise.

3. The third approach is more subtle. It consists of adding noise during training, thus effectively obfuscating the learning process. This approach is still ineffective against the adversary since, intuitively, the final classifier must anyway converge to classify correctly the training set. Thus, the noise must be somehow restrained and its effect can easily be mitigated (see below).

It may be unclear why the third approach above fails to provide any protection against the adversary. Hence, the following experiments show how to extract sensitive information from a classifier trained within the framework SulQ introduced in [9]. The SulQ authors improved several standard classifiers to provide differential privacy. The main idea consists of adding a small amount of noise, according to a Normal Distribution $N(0, \sigma)$, to any access to the training set. The variance of N regulates the privacy property provided by differential privacy.

The experiments against the differential privacy use the K-Means algorithm, which is the most popular clustering algorithm briefly introduced in section 6.4.1.

6.4.1 K-Means

Clustering is the task of partitioning unstructured data in such a way that objects with an high level of similarity fall into the same partition. Clustering is a typical example of unsupervised learning models where examples are unlabeled, i.e., they are not pre-classified. The *K-Means* algorithm, [96], is one of the most common methods in this family and it has been used in many applications, e.g.: [99, 152, 48, 7]. For example, in [99] the authors developed a real-time traffic classification method, based on *K-Means*, to identify SSH flows from statistical behavior of IP traffic parameters, such as length, arrival times and direction of packets.

In *K-Means* both training and classification phases are very intuitive. During the learning process, the algorithm partitions a set of n observations into k clusters. Then, the algorithm selects the *centroid* (i.e., the barycenter, or geometric midpoint) of every cluster as a representative for that set of objects. More formally, given a set of observations $(x_1, x_2, \ldots, x_n)$, where each observation is a *d-dimensional* real vector, *K-Means* partitions the n observations into k sets $(k \leq n)$ $S = \{S_1, S_2, \ldots, S_k\}$ in order to minimize the within-cluster function:

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\| \tag{6.6}$$

where μ_i is the mean of points in S_i.

To classify a given data set of *d-dimensional* elements with respect to k clusters, *K-Means* runs a learning process that can be summarized by the following steps:

1. Randomly pick k initial cluster centroids;
2. Assign each instance x to the cluster that has a centroid nearest to x;
3. Recompute each cluster's centroid based on which elements are contained in it;
4. Repeat Steps 2 and 3 until convergence is achieved.

6.4.2 Hacking models secured by Differential Privacy

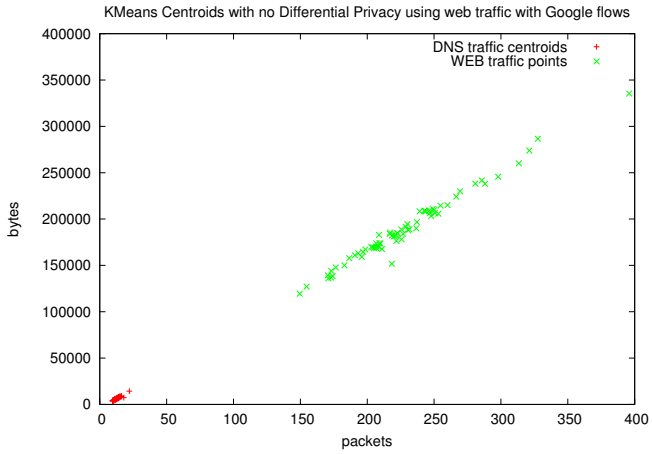
In order to hack the models secured by differential privacy two network traffic classifier based on *K-Means* have been implemented. Both classifiers have been trained with the same data set of the SVM experiment (section 6.3.2). The first classifier directly uses the Euclidean distance as metric to revise the *centroids* in the iterative refinement phase (equation 6.6). The second classifier implements a privacy preserving version of *K-Means*, providing differential privacy (the implementation is consistent with the definition provided within the SulQ framework, [9]). The version of *K-Means* presented in SulQ modifies the update rule of the centroids. In particular it defines a metric that uses approximated values of any sum and any count of points in the training set[1].

The experiment uses 70 training sets for both classifiers and it gets 70 distinct centroids. Recall that the objective of these experiments is to recognize whether there was Google traffic within the traces.

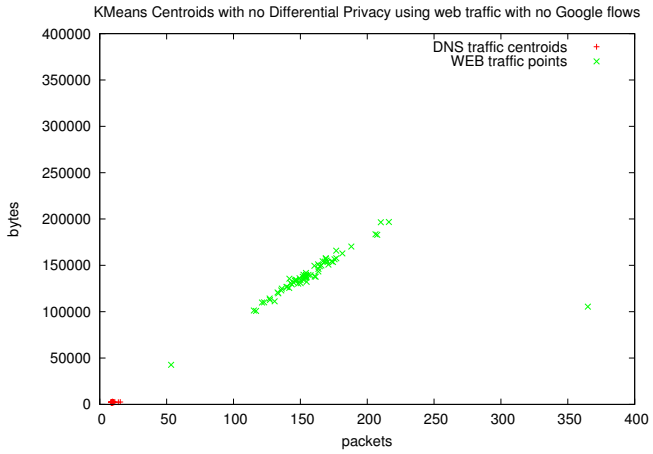
Figure 6.6(a) shows the centroids of the K-Means model obtained using a Classifiers that does not implements differential privacy and that includes web traffic directed to `Google.com`. Figure 6.6(b) represents the model obtained with the same classifier trained without traffic directed to `Google.com`. It is easy to see that the positions of the centroids are quite different, in fact, it's easy to distinguish between these two cases.

Figures 6.7(a) and 6.7(b) plot the centroids of the classifier which uses differential privacy. Even in this case, an adversary can easily distinguish whether there is `Google.com` traffic or not.

[1] As the original *K-Means*, the update rule is iterated until some convergence criterion has been reached, or a fixed number of iterations have been applied.

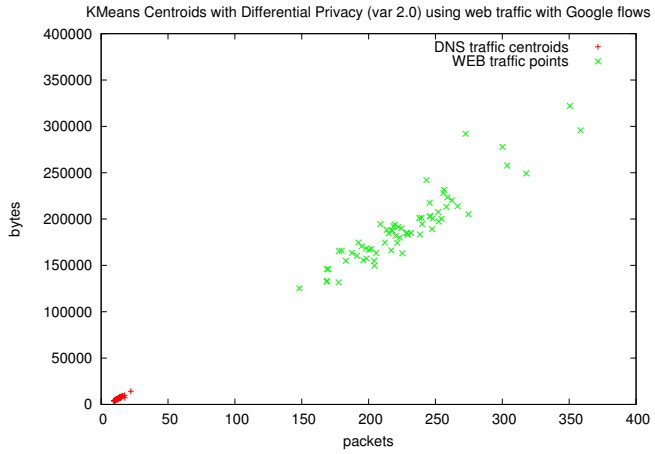


(a) Training set contains Web traffic directed to Google.com

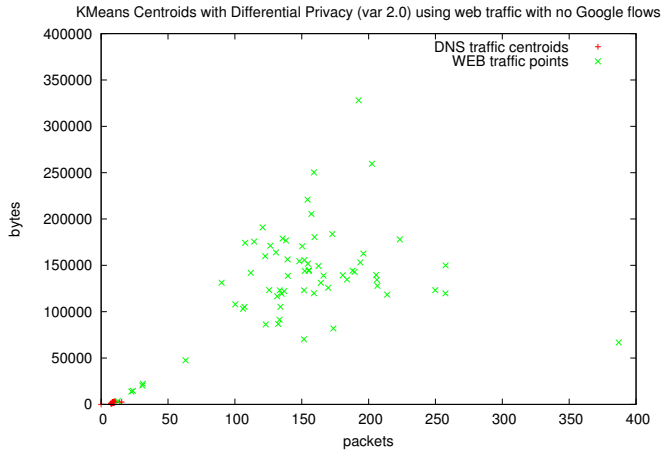


(b) Training set does not contain Web traffic directed to Google.com

Figure 6.6: K-Means: Centroids of the Internet Traffic Classifier *without* Differential Privacy.



(a) Training set contains Web traffic directed to Google.com



(b) Training set does not contain Web traffic directed to Google.com

Figure 6.7: K-Means: Centroids of the Internet Traffic Classifier *with* Differential Privacy.

Chapter 7

Conclusions

This dissertation contributes to the research related to the Internet cyber security and dealing with several issues in this domain.

First, this dissertation focuses on the security issues of the Border Gateway Protocol, BGP, which is responsible of the *control plane*. The security of Internet Routing Protocol BGP has become one of the main interests of Internet community. In this dissertation, chapter 3 describes an efficient and secure version of BGP, reBGP, that avoids the use of Public Key Infrastructures, relieving each AS nodes of the introduced overheads. The fundamentals of reBGP are the Identity Based Aggregate Signatures to get rid of public key distribution and the use of timestamps to enforce prefix and route revocations. Section 3.4 evaluates the overheads introduced in real case scenarios, and shows that the proposal outperforms S-BGP, namely the well known secure version of BGP. Protocol reBGP does not require any PKI, requires less volatile memory, introduces smaller computational complexity and brings out a smaller delay compared to S-BGP.
Next directions of this research include the evaluations of improvements (e.g., the reduction of the memory occupancy) and the design and implementation of more features, like AS number revocations. In fact, the revocations of the Internet resources are a critical aspect of BGP. Usually, traditional revocation means are based on Certificate Revocation Lists (CRL). In CRL-based revocation systems, a revocation authority (most of the time

the same authority that issues public key certificates) periodically updates and broadcasts a list of revoked public keys. A public key is valid as long as it does not appear in the latest list of revoked keys. Timely distribution of the revocation lists is key to an effective revocation means. However, CRL size and management do not fit the large scale of the Internet infrastructure. In particular, CRL suffers from message size and verification overhead linear in the number of revoked public keys. These issues must be redefined in the reBGP settings. Furthermore, the adoption of a new routing protocol requires the analysis of the additional costs.

Chapter 4 addresses the issue of Denial of Service Detection, and shows the effectiveness of several metrics based on Information Theory using a *real* and representative dataset obtained within the research activities of the ExTrABIRE project. The analysis were carried out using the CISCO^{TM} Netflows as data sources. In terms of Denial of Service Detection, one of the most interesting challenges is the formal definition of a threshold value, whose correctly distinguishes legitimate and malicious activities. Additional future works are:

- based on *real* dataset, the analysis of metrics and strategies borrowed by other academic fields, e.g. statistic or machine learning based approaches;
- the analysis of metrics and strategies using obfuscated data sets, according with the (k, j)-obfuscation technique.

It's important to note that, using real data which include real attacks, it is very hard to outline the ROC curve (Receiver Operating Characteristic, namely the plots of the true positive rate against the false positive rate) that well represents the effectiveness of the proposed strategy. This happens simply because we don't know the base truth for all the attack events.

Furthermore, this dissertation addresses the security and privacy concerns of obfuscation of sensitive information in real dataset of network flows. Chapter 5 formally models this issues, and proposes a novel defense technique, the (k, j)-obfuscation. Differently from previous proposals, the (k, j)-obfuscation provides formal guarantees under realistic as-

sumptions about the adversary's knowledge. An extensive experimental evaluation with a large set of real network flows shows that (k, j)-obfuscation also preserves the utility of network data. The (k, j)-obfuscation technique opens sundry interesting future directions:

- many networking and security tasks can be *re-thought* based on obfuscated datasets, for instance, quality of service (QoS), traffic classification, anomaly detection and more;
- the (k, j)-obfuscation technique can be extended in such a way that the network flows owner releases the obfuscated datasets incrementally;
- the (k, j)-obfuscation technique can be extended to different adversary models; in particular, one in which the hosts fingerprint may change over time.

Finally, this thesis introduces a novel approach to extract meaningful information from machine learning classifiers, using a meta-classifier. While previous works investigated privacy concerns of the single records of databases, the attack strategy proposed in chapter 6 focuses on the statistical information correlated to the *entire* training set used during the learning phase. Section 6.3 empirically shows that several ML classifiers suffer from a new class of information leakage that is not captured by privacy-preserving models, such as Privacy Preserving Data Mining or Differential Privacy. In particular, the practical attacks showed successfully distinguish the accents of users involved in the corpus of a speech recognition engine and also distinguish the traffic used during the creation of a Internet traffic Classifier. Since this kind of leakage has been never studied before, this outcome opens many future directions in terms of theoretical analysis and practical attacks. Theoretical directions are related to the quantification of the leakages, formal definition of the attack and the limits of such approach. On the opposite side, since the Machine Learning Algorithms pervade many Information Technology tasks, the strategy proposed in this dissertation can treat other contexts.

Bibliography

[1] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *Proceedings of the ACM Special Interest Group on Management of Data (SIGMOD)*, volume 29, pages 439–450, New York, NY, USA, May 2000. ACM.

[2] Tom Auld, Andrew W. Moore, and Stephen F. Gull. Bayesian neural networks for internet traffic classification. In *IEEE Transactions on Neural Networks*, volume 18, pages 223–239, 2007.

[3] Hitesh Ballani, Paul Francis, and Xinyang Zhang. A study of prefix hijacking and interception in the Internet. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM 2007)*, volume 37, pages 265–276, New York, NY, USA, August 2007. ACM.

[4] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state Markov chains. In *Annals of Mathematical Statistics*, volume 37, pages 1554–1563, 1966.

[5] BBC. China denies hijacking a huge chunk of US net traffic. `http://www.bbc.co.uk/news/technology-11773146`.

[6] BBC. Pakistan lifts the ban on Youtube. `http://news.bbc.co.uk/2/hi/technology/7262071.stm`.

[7] Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, Augustin Soule, and Kave Salamatian. Traffic classification on the fly. In *ACM*

SIGCOMM Computer Communication Review, volume 36, pages 23–26, New York, NY, USA, April 2006. ACM.

[8] BGPMon. `http://bgpmon.net/blog/?p=282`.

[9] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the sulq framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles Of Database Systems (PODS)*, pages 128–138, New York, NY, USA, 2005. ACM.

[10] Alexandra Boldyreva, Craig Gentry, Adam O'Neill, and Dae Hyun Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, pages 276–285, New York, NY, USA, 2007. ACM.

[11] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

[12] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth Publishing Company, Statistics/Probability Series, Belmont, California, U.S.A., 1984.

[13] Tønnes Brekne and André Årnes. Circumventing IP-address pseudonymization. In *Proceedings of International Conference on Communications and Computer Networks (ICCCN), Marina del Rey, USA*, pages 43–48. IASTED/ACTA Press, 2005.

[14] Tønnes Brekne, André Årnes, and Arne Øslebø. Anonymization of IP traffic monitoring data: Attacks on two prefix-preserving anonymization schemes and some proposed remedies. In *5th Workshop on Privacy Enhancing Technologies*, volume 3856 of *LNCS*, pages 179–196. Springer, 2006.

[15] Kyle Brogle, Sharon Goldberg, and Leonid Reyzin. Sequential aggregate signatures with lazy verification for s-bgp. ePrint Technical Report 2011/222, 2011.

[16] Bezawada Bruhadeshwar, Sandeep S. Kulkarni, and Alex X. Liu. Symmetric key approaches to securing BGP - a little bit trust is enough. In *IEEE Transactions on Parallel and Distributed Systems*, volume 22, pages 1536–1549, 2011.

[17] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. volume 2, pages 121–167, Hingham, MA, USA, June 1998. Kluwer Academic Publishers.

[18] Martin Burkhart, Dominik Schatzmann, Brian Trammell, Elisa Boschi, and Bernhard Plattner. The role of network trace anonymization under attack. In *ACM SIGCOMM Computer Communication Review*, volume 40, pages 5–11, 2010.

[19] K. Butler, T.R. Farley, P. McDaniel, and J. Rexford. A survey of BGP security issues and solutions. *Proceedings of the IEEE*, 98(1):100 –122, January 2010.

[20] A. R. Butz. Alternative algorithm for Hilbert's space-filling curve. In *IEEE Transactions on Computers*, volume 20, pages 424–426, 1971.

[21] Haowen Chan, Debabrata Dash, Adrian Perrig, and Hui Zhang. Modeling adoptability of secure BGP protocols. In *In Proceedings of the ACM SIGCOMM*, 2006.

[22] Yi-Tung F. Chan, Charles A. Shoniregun, and Galyna A. Akmayeva. A netflow based internet-worm detecting system in large network. In Pit Pichappan and Ajith Abraham, editors, *Third IEEE International Conference on Digital Information Management (ICDIM)*, pages 581–586. IEEE, 2008.

[23] C. I. Chang, Y. Du, J. Wang, S. M. Guo, and P. D. Thouin. Survey and comparative analysis of entropy and relative entropy thresholding

techniques. In *Vision, Image and Signal Processing, IEE Proceedings -*, volume 153, pages 837–850, 2006.

[24] Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In *Processing of Neural Information Processing System (NIPS)*, pages 289–296, 2008.

[25] Yves Chauvin and David E. Rumelhart, editors. *Backpropagation: theory, architectures, and applications*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1995.

[26] Rung Ching Chen, Kai-Fan Cheng, and Chia-Fen Hsieh. Using rough set and support vector machine for network intrusion detection. In *Proceedings of the First Asian Conference on Intelligent Information and Database Systems (ACIIDS)*, volume abs/1004.0567, 2009.

[27] Kim-Kwang Raymond Choo. High tech criminal threats to the national information infrastructure. In *Jorunal of Information Security technical Repor*, volume 15, pages 104–111, Oxford, UK, UK, August 2010. Elsevier Advanced Technology Publications.

[28] Cisco Systems. Cisco 2010 Annual Security Report, Highlighting global security threats and trends. `http://www.cisco.com/en/US/prod/vpndevc/annual_security_report.html`, 2010.

[29] Edith Cohen and Carsten Lund. Packet classification in large isps: design and evaluation of decision tree classifiers. In *ACM SIGMETRICS Performance Evaluation Review - Performance evaluation*, volume 33, pages 73–84, New York, NY, USA, June 2005. ACM.

[30] CASPUR Consortium and Department of Computer Science Università Sapienza of Rome. Extrabire — exchanged traffic analysis for a better internet resiliency in europe. `http://www.extrabire.eu/`, 2010-2012.

[31] Scott E. Coull, Fabian Monrose, Michael K. Reiter, and Michael Bailey. The challenges of effectively anonymizing network data. In

Conference for Homeland Security, Cybersecurity Applications and Technology, pages 230–236. IEEE Computer Society, 2009.

[32] Scott E. Coull, Charles V. Wright, Fabian Monrose, Michael P. Collins, and Michael K. Reiter. Playing devil's advocate: Inferring sensitive information from anonymized network traces. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 2007.

[33] J.H. Cowie, D.M. Nicol, and A.T. Ogielski. Modeling the global internet. In *Journal of Computing in Science Engineering*, volume 1, pages 42 –50, January/February 1999.

[34] Reza Curtmola, Aniello Del Sorbo, Giuseppe Ateniese, and Aniello Del. On the performance and analysis of dns security extensions. In *Proceedings of Cryptology and Network Security (CANS)*, pages 288–303. SpringerVerlag, 2005.

[35] Christos Davatzikos, Dinggang Shen, Zhiqiang Lao, Zhong Xue, and Bilge Karaçali. Morphological classification of medical images using nonlinear support vector machines. In *IEEE International Symposium on Biomedical Imaging: Nano to Macro (ISBI)*, pages 587–590, 2004.

[36] Vasant Dhar. Prediction in financial markets: The case for small disjuncts. In *ACM Transactions on Intelligent Systems and Technology (TIST)*, volume 2, pages 19:1–19:22, New York, NY, USA, April 2011. ACM.

[37] Roberto Di Pietro, Stefano Chessa, and Piero Maestrini. Computation, memory and bandwidth efficient distillation codes to mitigate dos in multicast. In *International Conference on Security and Privacy in Communication Networks (SECURECOMM)*, pages 13–22, Washington, DC, USA, 2005. IEEE Computer Society.

[38] Roberto Di Pietro and Luigi V. Mancini. *Intrusion Detection Systems*. Springer-Verlag, 2008.

[39] Roberto Di Pietro, Gabriele Oligeri, Claudio Soriente, and Gene Tsudik. Intrusion-Resilience in Mobile Unattended WSNs. In *The proceedings of the 29th Annual IEEE International Conference on Computer Communications (INFOCOM 2010)*, pages 2303–2311. IEEE, 2010.

[40] Ruisheng Diao, Kai Sun, Vijay Vittal, Robert J. O'Keefe, Michael R. Richardson, Navin Bhatt, Dwayne Stradford, and Sanjoy K. Sarawgi. Decision Tree-Based Online Voltage Security Assessment Using PMU Measurements. In *IEEE Transactions on Power Systems*, volume 24, pages 832–839, May 2009.

[41] D.Vitali C.Bettini D.Riboni, A.Villani and L.V.Mancini. Obfuscation of sensitive data in network flows. In *the 31st Annual IEEE International Conference on Computer Communications (IEEE INFOCOM 2012), Orlando, FL*, 2012.

[42] Thomas Dübendorfer, Arno Wagner, and Bernhard Plattner. A framework for real-time worm attack detection and backbone monitoring. In *proceedings of 1st IEEE International Workshop on Critical Infrastructure Protection (IWCIP 2005)*, 2005.

[43] A. Spognardi R. Battistoni D.Vitali, A. Villani and L.V. Mancini. DDoS detection with information theory metrics and netflows: a real case. In *Proceedings of the 9th International Conference on Security and Cryptography (SECRYPT)*, 2012.

[44] Cynthia Dwork. Differential Privacy. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, volume 4052 of *LNCS*, pages 1–12. Springer, 2006.

[45] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Proceedings of the Third conference on Theory of Cryptography (TCC)*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006.

[46] U.S.-CHINA Economic and Security Review Commission. Report to congress of the U.S.-CHINA economic and security review commission, 2010. `http://www.uscc.gov/annual_report/2010/annual_report_full_10.pdf`.

[47] Jason Eisner. An interactive spreadsheet for teaching the forward-backward algorithm. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics - Volume 1*, ETMTNLP '02, pages 10–18, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[48] Jeffrey Erman, Martin Arlitt, and Anirban Mahanti. Traffic classification using clustering algorithms. In *Proceedings of the SIGCOMM workshop on Mining network data*, pages 281–286, New York, NY, USA, 2006. ACM.

[49] Jeffrey Erman, Anirban Mahanti, Martin Arlitt, Ira Cohen, and Carey Williamson. Offline/realtime traffic classification using semi-supervised learning. In *Jorunal of Performance and Evaluation*, volume 64, pages 1194–1213. Elsevier Science Publishers B. V., October 2007.

[50] Alice Este, Francesco Gringoli, and Luca Salgarelli. Support vector machines for tcp traffic classification. In *Computer Networks*, volume 53, pages 2476 – 2490, 2009.

[51] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the Internet topology. In *SIGCOMM Compututer Communication Review*, volume 29, pages 251–262, August 1999.

[52] Jinliang Fan, Jun Xu, Mostafa H. Ammar, and Sue B. Moon. Prefix-preserving IP address anonymization: measurement-based security evaluation and a new cryptography-based scheme. In *Journal of Computer Networks*, volume 46, pages 253–272, 2004.

[53] Laura Feinstein and Dan Schnackenberg. Statistical approaches to DDOS attack detection and response. In *In Proceedings of the DARPA Information Survivability Conference and Exposition*, pages 303–314, 2003.

[54] Michalis Foukarakis, Demetres Antoniades, and Michalis Polychronakis. Deep packet anonymization. In *Proceedings of the Second European Workshop on System Security (EUROSEC)*, pages 16–21. ACM, 2009.

[55] A. Frank and A. Asuncion. Machine learning repository, 2010. http://archive.ics.uci.edu/ml.

[56] A. Spognardi A. Villani D. Vitali G. Ateniese, L.V.Mancini. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. Submitted for publication, 2012.

[57] Craig Gentry and Zulfikar Ramzan. Identity-based aggregate signatures. In *Proceedings of The International Conference on Practice and Theory in Public Key Cryptography (PKC)*, pages 257–273, 2006.

[58] Sharon Goldberg, Shai Halevi, Aaron D. Jaggard, Vijay Ramachandran, and Rebecca N. Wright. Rationality and traffic attraction: incentives for honest path announcements in bgp. In *proceedings of the ACM SIGCOMM Compututer Communication Review*, volume 38, pages 267–278. ACM, August 2008.

[59] Sharon Goldberg, Michael Schapira, Peter Hummon, and Jennifer Rexford. How secure are secure interdomain routing protocols. In *proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 87–98, 2010.

[60] Timothy G. Griffin. An experimental analysis of BGP convergence time. In *In Proceedings of the 9th International Conference on Network Protocols (ICNP)*, pages 53–61, 2001.

[61] Yu Gu, Andrew McCallum, and Donald F. Towsley. Detecting anomalies in network traffic using maximum entropy estimation. In *proceedings of ACM SIGCOMM IIMC*, pages 345–350. USENIX Association, 2005.

[62] Guodong Guo, Stan Z. Li, and Kapluk Chan. Face recognition by support vector machines. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*, FG '00, pages 196–, Washington, DC, USA, 2000. IEEE Computer Society.

[63] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. In *Journale of Machine Learning*, volume 46, pages 389–422, Hingham, MA, USA, March 2002. Kluwer Academic Publishers.

[64] J.F. Hayes. The viterbi algorithm applied to digital data transmission. *Communications Magazine, IEEE*, 40(5):26 –32, may 2002.

[65] Ypke Hiemstra. Linear regression versus backpropagation networks to predict quarterly stock market excess returns. In *Journal Computational Economics - Special issue on computational finance*, volume 9, February.

[66] Wenjie Hu, Yihua Liao, and V. Rao Vemuri. Robust anomaly detection using support vector machines. In *In Proceedings of the International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc.

[67] Yih-Chun Hu, Adrian Perrig, and Marvin Sirbu. Spv: secure path vector routing for securing BGP. In *SIGCOMM Compututer Communication Review*, volume 34, pages 179–192. ACM, August 2004.

[68] John Mc Hugh. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory. volume 3, pages 262–294, New York, NY, USA, November 2000. ACM.

[69] Anil K. Jain, Jianchang Mao, and K. Mohiuddin. Artificial neural networks: A tutorial. *IEEE Computer*, 29:31–44, 1996.

[70] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Journal of Machine Learning*, Lecture Notes in Computer Science.

[71] B. H. Juang and L. R. Rabiner. Hidden markov models for speech recognition. In *Technometrics*, volume 33, pages 251–272. American Society for Quality Control and American Statistical Association, August 1991.

[72] Daniel Jurafsky and James H. Martin. Speech and language processing (2nd edition).

[73] Stephen Kent, Charles Lynn, Joanne Mikkelson, and Karen Seo. Secure border gateway protocol (S-BGP). *IEEE Journal on Selected Areas in Communications*, 18:103–116, 2000.

[74] Stephen T. Kent, Charles Lynn, Joanne Mikkelson, and Karen Seo. Secure border gateway protocol (S-BGP) - real world performance and deployment issues. In *Proceedings of the Network and Distributed System Security Symposium, (NDSS)*, pages 1–1, 2000.

[75] Latifur Khan, Mamoun Awad, and Bhavani Thuraisingham. A new intrusion detection system using support vector machines and hierarchical clustering. In *The Very Large Data Bases Journal (VLDB)*, volume 16, pages 507–521. Springer-Verlag New York, Inc., October 2007.

[76] Justin King, Kiran Lakkaraju, and Adam J. Slagell. A taxonomy and adversarial model for attacks against network log anonymization. In *proceedings of the ACM Symposium On Applied Computing (SAC)*, pages 1286–1293. ACM, 2009.

[77] Klaus A. Kuhn, James R. Warren, and Tze-Yun Leong. A new machine learning classifier for high dimensional healthcare data. In

MEDINFO 2007 - Proceedings of the 12th World Congress on Health (Medical) Informatics - Building Sustainable Health Systems, Studies in Health Technology and Informatics. IOS Press, August.

[78] Kaspersky Lab. Flame. `http://www.kaspersky.com/about/news/virus/2012/Kaspersky_Lab_and_ITU_Research_Reveals_New_Advanced_Cyber_Threat`.

[79] Craig Labovitz, Abha Ahuja, Abhijit Bose, and Farnam Jahanian. Delayed internet routing convergence. In *proceedings ACM SIGCOMM*, pages 175–187, 2000.

[80] Anna T. Lawniczak, Bruno N. Di Stefano, and Hao Wu. Detection & study of DDoS attacks via entropy in data network models. In *Proceedings of the Second IEEE international conference on Computational intelligence for security and defense applications*, CISDA'09, pages 59–66, Piscataway, NJ, USA, 2009. IEEE Press.

[81] A. Lee, T. Kawahara, and K. Shikano. Julius — an open source real-time large vocabulary recognition engine. In *Proceedings of Eurospeech conference*, pages 1691–1694, Aalborg, Denmark, 2001.

[82] K.-F. Lee and H.-W. Hon. Large-vocabulary speaker-independent continuous speech recognition using hmm. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 123–126, apr 1988.

[83] Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 14:255–293, 1999.

[84] John Leyden. 4chan launches ddos against entertainment industry. `http://www.theregister.co.uk/2010/09/20/4chan_ddos_mpaa_riaa/`, 2010.

[85] K. Li, W. Zhou, and S. Yu. Effective metric for detecting distributed denial-of-service attacks based on information divergence. In *IET Communications*, volume 3, pages 1851–1860, 2009.

[86] Ke Li, Wanlei Zhou, Shui Yu, and Bo Dai. Effective DDOS attacks detection using generalized entropy metric. In *Proceedings of the 9th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP)*, pages 266–280, Berlin, Heidelberg, 2009. Springer-Verlag.

[87] Liying Li, Jianying Zhou, and Ning Xiao. DDoS attack detection algorithms based on entropy computing. In *Proceedings of the 9th international conference on Information and communications security*, ICICS'07, pages 452–466, Berlin, Heidelberg, 2007. Springer-Verlag.

[88] Ming Li and Zhi-Hua Zhou. Improve Computer-Aided Diagnosis With Machine Learning Techniques Using Undiagnosed Samples. In *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, volume 37, pages 1088–1098, November 2007.

[89] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Proceedings of International Conference on Data Engineering (ICDE)*, pages 106–115. IEEE Computer Society, 2007.

[90] Qi Li, Mingwei Xuy, Jianping Wuy, Xinwen Zhangz, Patrick P. C. Leex, and Ke Xuy. Enhancing the trust of internet routing with lightweight route attestation. In *6th ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, pages 92–101, 2011.

[91] PBC library. The pairing-based cryptography library. Website, 2006. `http://crypto.stanford.edu/pbc/`.

[92] Jianhua Lin. Divergence measures based on the shannon entropy. In *IEEE Transactions on Information theory*, volume 37, pages 145–151, 1991.

[93] Yehuda Lindell and Benny Pinkas. Secure multiparty computation for privacy-preserving data mining. In *IACR Cryptology ePrint Archive*, volume 2008, page 197, 2008.

[94] A.Spognardi A.Villani D.Vitali L.V.Mancini, C.Soriente. relieve internet routing of the public key infrastructure. In *International Conference on Computer Communication Networks (ICCCN)*, 2012.

[95] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *Proceedings of International Conference on Data Engineering (ICDE)*. IEEE Computer Society, 2006.

[96] J. B. Macqueen. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[97] George D. Magoulas and Andriana Prentza. Machine learning in medical applications. In *Machine Learning and Its Applications*, pages 300–307, 2001.

[98] Ratul Mahajan, David Wetherall, and Tom Anderson. Understanding BGP misconfiguration. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, pages 3–16, New York, NY, USA, 2002. ACM.

[99] Gianluca Maiolini, Andrea Baiocchi, Alfonso Iacovazzi, and Antonello Rizzi. Real time identification of ssh encrypted application flows by using cluster analysis techniques. In *Proceedings of the 8th International IFIP-TC 6 Networking Conference (NETWORKING)*, pages 182–194, Berlin, Heidelberg, 2009. Springer-Verlag.

[100] Adam Meyerson and Ryan Williams. On the complexity of optimal K-anonymity. In *proceedings of PODS'04*, pages 223–228. ACM Pub., 2004.

[101] Jelena Mirkovic and Peter Reiher. A taxonomy of DDOS attack and DDOS defense mechanisms. In *SIGCOMM Compututer Communications*, volume 34, pages 39–53, New York, NY, USA, April 2004. ACM.

[102] T. Mitchell. *Machine Learning*. McGraw-Hill Education (ISE Editions), 1st edition, October 1997.

[103] S. Mukkamala, G. Janoski, and A. Sung. Intrusion detection using neural networks and Support Vector Machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*.

[104] Snehal A. Mulay, P.R. Devale, and G.V. Garje. Article:intrusion detection system using support vector machine and decision tree. volume 3, pages 40–43, June 2010. Published By Foundation of Computer Science.

[105] Ripe NCC. Youtube hijacking: a RIPE NCC RIS case study. `http://www.ripe.net/internet-coordination/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study`.

[106] T.T.T. Nguyen and G. Armitage. A survey of techniques for internet traffic classification using machine learning. *Communications Surveys Tutorials, IEEE*, 10(4):56 –76, quarter 2008.

[107] Giseop No and Ilkyeun Ra. An efficient and reliable DDOS attack detection using a fast entropy computation method. In *Proceedings of the 9th international conference on Communications and information technologies*, ISCIT'09, pages 1223–1228, Piscataway, NJ, USA, 2009. IEEE Press.

[108] Inc Nuance Communications. `http://www.nuance.com/dragon/index.htm`.

[109] George Nychis, Vyas Sekar, David G. Andersen, Hyong Kim, and Hui Zhang. An empirical evaluation of entropy-based traffic anomaly detection. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, IMC '08, pages 151–156, New York, NY, USA, 2008. ACM.

[110] P.C. van Oorschot, Tao Wan, and Evangelos Kranakis. On interdomain routing security and pretty secure BGP (psBGP). volume 10, New York, NY, USA, July 2007. ACM.

[111] OpenSSL. The open source toolkit for SSL/TLS. Website, 1998. `http://www.openssl.org`.

[112] Shunsuke Oshima, Takuo Nakashima, and Toshinori Sueyoshi. DDoS detection technique using statistical analysis to generate quick response time. In *Proceedings of the International Conference on Broadband, Wireless Computing, Communication and Applications*, BWCCA '10, pages 672–677. IEEE Computer Society, 2010.

[113] Shunsuke Oshima, Takuo Nakashima, and Toshinori Sueyoshi. Early DoS/DDOS detection method using short-term statistics. In *Proceedings of the International Conference on Complex, Intelligent and Software Intensive Systems (CISIS)*, pages 168–173. IEEE Computer Society, 2010.

[114] V.K. Pachghare and P. Kulkarni. Pattern based network security using decision trees and support vector machine. In *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, volume 5, pages 254 –257, april 2011.

[115] Ruoming Pang, Mark Allman, Vern Paxson, and Jason Lee. The devil and packet trace anonymization. *Computer Communication Review*, 36(1):29–38, 2006.

[116] Xiu-Li Pang, Yu-Qiang Feng, and Wei Jiang. A spam filter approach with the improved machine learning technology. In *Natural Computation, 2007. ICNC 2007. Third International Conference on*, volume 2, pages 484 –488, aug. 2007.

[117] P. Papadimitratos and Z.J. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of the SCS Commnication Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, pages 193–204, 2002.

[118] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schoelkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. 1998.

[119] Darius Plikynas and Yusaf H. Akbar. Application of modified neural network weights' matrices explaining determinants of foreign investment patterns in the emerging markets. In *Proceedings of the 4th Mexican international conference on Advances in Artificial Intelligence*, MICAI'05, pages 721–730, Berlin, Heidelberg, 2005. Springer-Verlag.

[120] Vili Podgorelec, Peter Kokol, Bruno Stiglic, and Ivan Rozman. Decision trees: an overview and their use in medicine. *Journal of Medical Systems*, 26:445–463, 2002.

[121] J. R. Quinlan. Induction of decision trees. volume 1, pages 81–106, Hingham, MA, USA, March 1986. Kluwer Academic Publishers.

[122] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[123] Barath Raghavan, Saurabh Panjwani, and Anton Mityagin. Analysis of the spv secure routing protocol: weaknesses and lessons. In *SIGCOMM Compututer Communication Review*, volume 37, pages 29–38, New York, NY, USA, March 2007. ACM.

[124] Fahmida Y. Rashid. Paypal, postfinance hit by dos attacks, counterattack in progress. `http://www.eweek.com/c/a/Security/PayPal-PostFinance-Hit-by-DoS-AttacksCounterAttack-in-Progress-860335/`, 2006.

[125] Y. Rekhter and T. Li. A border gateway protocol 4 (BGP-4). In *RFC 1771*, Mar. 1995.

[126] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), January 2006. Updated by RFC 6286.

[127] Ripe NCC. YouTube Hijacking: a RIPE NCC RIS case study. online report at www.ripe.net, Mar 2008.

[128] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Transaction on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[129] Anjali Sardana, Ramesh Joshi, and Tai-hoon Kim. Deciding optimal entropic thresholds to calibrate the detection mechanism for variable rate DDOS attacks in ISP domain. In *Proceedings International Conference on Information Security and Assurance (ISA)*, pages 270–275. IEEE Computer Society, 2008.

[130] John Godfrey Saxe. Blind men and an elephant. `http://en.wikipedia.org/wiki/Blind_men_and_an_elephant`.

[131] Alan Schwartz. *SpamAssassin*. O'Reilly Media, Inc., 2004.

[132] Vyas Sekar and Jacobus Van Der Merwe. Lads: Large-scale automated ddos detection system. In *In proceedings of USENIX Annual Technical Conference (ATC)*, pages 171–184, 2006.

[133] Claude E. Shannon. A mathematical theory of communication. In *The Bell system technical journal*, volume 27, pages 379–423, 1948.

[134] Adam J. Slagell, Kiran Lakkaraju, and Katherine Luo. FLAIM: A multi-level anonymization framework for computer and network logs. In *proceedings of the USENIX Large Installation System Administration Conference conference (LISA)*, pages 63–77. USENIX, 2006.

[135] Xiaomu Song, George Iordanescu, and Alice M. Wyrwicz. One-class machine learning for brain activation detection. In *CVPR*, 2007.

[136] Augustin Soule, Kavé Salamatian, and Nina Taft. Combining filtering and statistical methods for anomaly detection. In *In ACM Internet Measurement Conference*, 2005.

[137] SSFNET. Scalable simulation framework—network models. Website, 1999. `http://www.ssfnet.org`.

[138] Martin Suchara, Ioannis Avramopoulos, and Jennifer Rexford. Securing BGP incrementally. In *Proceedings of the ACM Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, pages 52:1–52:2, New York, NY, USA, 2007. ACM.

[139] Cisco Systems. Cisco Systems NetFlow Services Export Version 9. `http://tools.ietf.org/html/rfc3954`, 2004.

[140] Adi L Tarca, Vincent J Carey, Xue-wen Chen, Roberto Romero, and Sorin Drăghici. Machine learning and its applications to biology. volume 3, page e116. Public Library of Science, 06 2007.

[141] University of Oregon. Route Views Project.

[142] Vassilios S. Verykios, Elisa Bertino, Igor Nai Fovino, Loredana Parasiliti Provenza, Yucel Saygin, and Yannis Theodoridis. State of the-art in privacy preserving data mining. In *proceedings of the Special Interest Group on Management of Data (SIGMOD)*, volume 33, pages 50–57, New York, NY, USA, March 2004. ACM.

[143] C Villamizar, R Chandra, and R Govindan. BGP route flap damping. volume 9, pages 131–138. IETF, 1998.

[144] voxforge. `http://www.voxforge.org`.

[145] Weka Machine Learning Project. Weka. URL http://www.cs.waikato.ac.nz/m̃l/weka.

[146] M. Wernick, Yongyi Yang, J. Brankov, G. Yourganov, and S. Strother. Machine learning in medical imaging. In *Signal Processing Magazine, IEEE*, volume 27, pages 25 –38, july 2010.

[147] Russ White. Securing BGP Through Secure Origin BGP. In *The Internet Protocol Journal*, volume Volume 6. Cisco Systems, sep 2003.

[148] Wikipedia. Stuxnet. `http://en.wikipedia.org/wiki/Stuxnet`.

[149] Edmund L. Wong, Praveen Balasubramanian, Lorenzo Alvisi, Mohamed G. Gouda, and Vitaly Shmatikov. Truth in advertising: lightweight verification of route integrity. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing (PODC)*, pages 147–156, 2007.

[150] Y. Xiang, K. Li, and W. Zhou. Low-rate DDOS attacks detection and traceback by using new information metrics. In *Information Forensics and Security, IEEE Transactions*, volume 99. IEEE Press, 2011.

[151] Xiaokui Xiao and Yufei Tao. *m*-invariance: towards privacy preserving re-publication of dynamic datasets. In *proceedings of Special Interest Group on Management of Data (SIGMOD)*, pages 689–700. ACM, 2007.

[152] Guowu Xie, M. Iliofotou, R. Keralapura, M. Faloutsos, and A. Nucci. Subflow: Towards practical flow-level traffic classification. In *The 31st Annual IEEE International Conference on Computer Communications (INFOCOM)*, pages 2541–2545, march 2012.

[153] Ting-Fang Yen, Xin Huang, Fabian Monrose, and Michael K. Reiter. Browser fingerprinting from coarse traffic summaries: Techniques and implications. In *proceedings of the Detection of Intrusions and Malware & Vulnerability Assessment conference (DIMVA)*, volume 5587 of *LNCS*, pages 157–175. Springer, 2009.

[154] Heng Yin, Bo Sheng, Haining Wang, and Jianping Pan. Keychain-based signatures for securing BGP. In *IEEE Journal on Selected Areas in Communications*, pages 1308–1318, 2010.

[155] Steve Young. A review of large-vocabulary continuous-speech. In *Signal Processing Magazine, IEEE*, volume 13, page 45, September 1996.

[156] Jing Yuan, Zhu Li, and Ruixi Yuan. Information entropy based clustering method for unsupervised internet traffic classification. In

proceedings of IEEE International Conference on Communications (IIC), pages 1588–1592. IEEE Computer Society, 2008.

[157] D. Xu Z. Lin, X. Zhang. Automatic reverse engineering of data structures from binary execution. In *Proceedings of the Network and Distributed System Security Symposium, (NDSS)*. The Internet Society, 2010.

[158] Meiyuan Zhao, Sean W. Smith, and David M. Nicol. Aggregated path authentication for efficient BGP security. In *In ACM Conferernce on Computer and Communication Security (CCS)*, pages 128–138, 2005.