



AMERICAN UNIVERSITY OF BEIRUT

THWARTING TRAFFIC CLASSIFICATION  
FOR PRIVACY PRESERVATION

by

LOUMA AHMAD CHADDAD

A dissertation  
submitted in partial fulfillment of the requirements  
for the degree of Doctor of Philosophy  
to the Department of Electrical and Computer Engineering  
of the Maroun Semaan Faculty of Engineering and Architecture  
at the American University of Beirut

Beirut, Lebanon  
April 2021

# AMERICAN UNIVERSITY OF BEIRUT

## THWARTING TRAFFIC CLASSIFICATION FOR PRIVACY PRESERVATION

by

LOUMA AHMAD CHADDAD

Approved by:

---

Dr. Ayman Kayssi, Professor

Electrical and Computer Engineering

Chairman



---

Dr. Ali Chehab, Professor

Electrical and Computer Engineering

Advisor

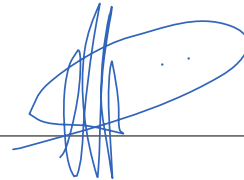


---

Dr. Imad Elhajj, Professor

Electrical and Computer Engineering

Member of Committee



---

Dr. Ramzi Haraty, Professor

Lebanese American University

Member of Committee



---

Dr. Wassim Itani, Professor

University of Houston-Victoria

Member of Committee

Date of dissertation defense: April 22, 2021

# AMERICAN UNIVERSITY OF BEIRUT

## DISSERTATION RELEASE FORM

Student Name: Chaddad \_\_\_\_\_ Louma \_\_\_\_\_ Ahmad \_\_\_\_\_  
Last First Middle

Master's Thesis       Master's Project       Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes after: **One \_\_\_ year from the date of submission of my dissertation .**

**Two \_\_\_ years from the date of submission of my dissertation .**

**Three \_\_\_ years from the date of submission of my dissertation .**

*Louma Chaddad*

*01/05/2021*

Signature

Date

# Acknowledgements

Firstly, I would like to express my deep gratitude to my advisor Prof. Ali Chehab, who expertly guided me through my PhD study and related research. I am fully indebted for his understanding, wisdom, encouragement, and for pushing me farther than I thought I could go. I would like also to thank Prof. Ayman Kayssi and Prof. Imad Elhajj for their continuous support during the last four years.

I would like to thank the rest of my thesis committee: Prof. Ramzi Haraty and Prof. Wassim Itani, for their insightful comments and encouragement.

I would like also to thank AUB University Research Board, and TELUS Corp., Canada for funding this research.

I would like to thank my close friends for their invaluable friendship, and my colleagues for the stimulating discussions, and for creating a nice atmosphere in lab environment.

Last but not the least, I acknowledge with a deep sense of reverence my parents: Ahmad and Ilham, my brothers: Ali, Mouhamad, and Amer, and my precious kids: Sara and Ali.

# An Abstract of the Dissertation of

Louma Ahmad Chaddad for Doctor of Philosophy  
Major: Electrical and Computer Engineering

Title: Thwarting Traffic Classification for Privacy Preservation

Research proved that supposedly secure encrypted network traffic is actually threatened by privacy and security violations from many aspects. This is mainly due to flow features leaking evidence about the user activity and data content. With the increasing popularity of machine learning techniques, the traditional means of encrypting packets are no longer feasible approaches from the privacy perspective. A passive adversary can monitor traffic patterns that remain intact after encryption, such as timing, size, direction, and count of packets in a specific network flow. Using these features, he can build classifiers and detect instances of application protocols. Hence, traffic analysis is considered a big threat for the privacy of Internet users. In this thesis, we aim at understanding if and how complicated it is to obfuscate the information leaked by traffic features. We define a security model of the typical thwarting system against malicious traffic analysis. Then, we propose practical techniques to prevent traffic feature leaks. These methods consist of modifying the flow's statistical characteristics to mislead traffic classifiers. However, they could impose overhead in terms of processing and memory resources. This fact is not acceptable for devices that have limited means, nor is it acceptable for applications with interactive dynamic nature. Thus, having efficient security is key while decreasing computation and storage overhead on the devices, and reducing latency on the traffic. Additionally, given that there are different types of heterogeneous apps, the obfuscation system needs to be dynamic and scalable. For these reasons, we define the optimized privacy-leak thwarting technique resulting in a tradeoff between privacy and complexity overhead. We propose a mathematical model for network obfuscation, and we formulate analytically a constrained optimization problem that treats maximization of network obfuscation while minimizing overhead as a cost.

The aim of our optimization is to decrease the security risks of statistical traffic analysis attacks by optimally obfuscating an app traffic. We propose dynamic algorithms to solve the optimization problem of traffic obfuscation by selecting the target app and the length from the target app to mutate to. We analyze the full privacy protection of our solutions using both analytical and experimental models. First, we suggest metrics for quantitative privacy measurement to measure obfuscation system's resilience to traffic analysis attacks. And then, we assess their effectiveness through extensive simulations on real-world data traces. Finally, we propose metrics based on information theory to explain the empirical results of obfuscation models. We also suggest criteria for selecting tunable parameters to achieve best results of the obfuscation model. Our measures evaluate, for any obfuscation system, the right choice of features to mutate, as well as the right choice of target applications to mutate to.

# Contents

<b>Acknowledgements</b>	<b>v</b>
<b>Abstract</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Contributions . . . . .	3
1.2 List of Publications . . . . .	4
1.3 Thesis roadmap . . . . .	5
<b>2 From Traffic Analysis to Traffic Obfuscation</b>	<b>6</b>
2.1 Traffic Analysis: State of the Art . . . . .	7
2.1.1 Port Based . . . . .	7
2.1.2 Payload Based . . . . .	8
2.1.3 Statistical Based . . . . .	8
2.1.4 Behavioral Based . . . . .	9
2.1.5 Hybrid methods . . . . .	10
2.2 Traffic Analysis Applications . . . . .	11
2.2.1 Traffic Analysis Methods/Attacks on Smartphones . . . . .	11
2.2.2 Traffic Analysis Methods/Attacks on Websites . . . . .	12
2.3 Traffic Obfuscation: State of The Art . . . . .	12
2.3.1 Anonymization . . . . .	12
2.3.2 Mutation . . . . .	14
2.3.3 Morphing . . . . .	15
2.3.4 Tunneling . . . . .	16
2.3.5 Other Techniques . . . . .	17
2.4 Website Traffic Anonymization . . . . .	17
2.5 Discussion: Key Findings and Limitations . . . . .	18
2.6 Network Traffic Modeling . . . . .	19
2.7 Conclusion . . . . .	19
<b>3 Classification Attack</b>	<b>20</b>
3.1 Traffic Analysis . . . . .	20
3.2 Machine Learning based Classification . . . . .	21



3.2.1	Decision Trees . . . . .	25
3.2.2	Random Forest . . . . .	25
3.2.3	Support Vector Machine (SVM) . . . . .	25
3.2.4	K-nearest neighbor (k-NN) . . . . .	26
3.3	Threat Model . . . . .	26
3.4	Classifier Adversary Model . . . . .	27
3.5	Classification Attack Empirical Evaluation . . . . .	28
3.5.1	Dataset . . . . .	28
3.5.2	Feature selection . . . . .	28
3.5.3	Model Training and Classification Results . . . . .	29
3.6	Discussion . . . . .	30
3.7	Conclusion . . . . .	32
<b>4</b>	<b>Obfuscation for Better Secrecy</b>	<b>34</b>
4.1	<i>AdaptiveMutate</i> . . . . .	34
4.1.1	Packet Length Mutation . . . . .	35
4.1.2	Inter-arrival time mutation . . . . .	36
4.1.3	Lengths and IAT mutation . . . . .	37
4.1.4	Practical considerations . . . . .	37
4.1.5	Obfuscation experiment and results . . . . .	40
4.1.6	Validation on MTU quantization . . . . .	44
4.1.7	Validation on <i>AppScanner</i> . . . . .	44
4.1.8	Adaptive real-time approach . . . . .	45
4.2	Anonymization Through Probabilistic Distribution . . . . .	46
4.2.1	System Overview . . . . .	46
4.2.2	Experiment Evaluation . . . . .	47
4.3	Mutation for Similar Probability Distribution . . . . .	54
4.3.1	Model Overview . . . . .	54
4.3.2	Experimental Evaluation . . . . .	56
4.4	Discussions . . . . .	58
4.5	Conclusion . . . . .	60
<b>5</b>	<b>Obfuscation Optimization</b>	<b>61</b>
5.1	Problem Statement . . . . .	61
5.2	Analytical Solution . . . . .	63
5.3	<i>Opriv</i> . . . . .	64
5.4	Experimental Evaluation . . . . .	66
5.4.1	Analytical Solution Implementation . . . . .	67
5.4.2	<i>Opriv</i> Evaluation . . . . .	67
5.4.3	Discussion . . . . .	68
5.5	Conclusion . . . . .	70

<b>6</b>	<b>Metrics to compare and enhance obfuscation efficiency</b>	<b>76</b>
6.1	Comparison Criteria and Methodology . . . . .	76
6.1.1	Defense Evaluation . . . . .	77
6.2	Information Theory . . . . .	77
6.2.1	Degree of traffic masking effectiveness . . . . .	78
6.2.2	Kullback-Leibler divergence . . . . .	80
6.2.3	Traffic divergence . . . . .	81
6.3	Conclusion . . . . .	82
<b>7</b>	<b>Conclusion</b>	<b>83</b>

# List of Figures

1.1	Problem Definition . . . . .	3
2.1	Traffic Analysis versus Traffic Obfuscation techniques . . . . .	7
2.2	Mutation techniques . . . . .	16
2.3	Morphing technique . . . . .	16
3.1	Packet captured from an encrypted SSH session . . . . .	21
3.2	Internet traffic classification process . . . . .	23
3.3	Threat Model . . . . .	27
3.4	Mobile Traffic Classification Attack . . . . .	28
4.1	Block scheme of mutation operation . . . . .	35
4.2	Lengths in bytes of the first 100 packets of 20 different flows in app traffic . . . . .	38
4.3	IAT in seconds of the first 100 packets of 20 different flows in app traffic . . . . .	39
4.4	<i>AdaptiveMutate</i> scheme . . . . .	45
4.5	App traffic packet sizes distribution . . . . .	47
4.6	Distribution fits with respect to <i>Skype</i> packet lengths . . . . .	48
4.7	Distribution fits with respect to <i>Viber</i> packet lengths . . . . .	49
4.8	Mutation Algorithm . . . . .	55
4.9	App Traffic Packet Sizes Distribution . . . . .	56
5.1	Camouflage Scheme . . . . .	64
5.2	OPL 12.9.0.0 Model . . . . .	65
5.3	OPL 12.9.0.0 Data . . . . .	66

# List of Tables

2.1	Traffic obfuscation techniques . . . . .	13
3.1	Experiment dataset . . . . .	29
3.2	Feature Set for statistical classification . . . . .	30
3.3	Features . . . . .	31
3.4	Dataset accuracy . . . . .	31
3.5	Confusion matrices for SVM, BT, KNN, and RF . . . . .	32
4.1	Cosine similarity of length vectors . . . . .	39
4.2	Cosine similarity of inter-arrival time vectors . . . . .	39
4.3	<i>AdaptiveMutate</i> results using first version of packet length mutation . . . . .	40
4.4	<i>AdaptiveMutate</i> results using second version of packet length mutation . . . . .	41
4.5	<i>AdaptiveMutate</i> results using mutation of IATs . . . . .	41
4.6	<i>AdaptiveMutate</i> results using mutation of lengths and IATs of packets . . . . .	42
4.7	Overhead resulting from padding . . . . .	42
4.8	Performance Time Delay Resulting from Mutating the IATs . . . . .	42
4.9	Confusion of <i>Skype</i> Traffic Mutated to <i>Game</i> by RF Classifier . . . . .	42
4.10	Packet lengths Distribution Fit Parameters and Goodness-of-fit . . . . .	49
4.11	Confusion of Source app mutated to Normal distribution of <i>Skype</i> packet sizes by Quadratic SVM . . . . .	50
4.12	Confusion of Source app mutated to Normal distribution of <i>Skype</i> packet sizes by Bagged Tree . . . . .	50
4.13	Confusion of Source app mutated to Normal distribution of <i>Skype</i> packet sizes by Fine KNN . . . . .	51
4.14	Confusion of Source app mutated to Normal distribution of <i>Skype</i> packet sizes by Random Forest . . . . .	51
4.15	Overhead resulting from mutating to Normal distribution of <i>Skype</i> packet sizes . . . . .	51
4.16	Confusion of Source app mutated to Poisson distribution of <i>Viber</i> packet sizes by Quadratic SVM . . . . .	52

4.17	Confusion of Source app mutated to Poisson distribution of <i>Viber</i> packet sizes by Bagged Tree . . . . .	52
4.18	Confusion of Source app mutated to Poisson distribution of <i>Viber</i> packet sizes by Fine KNN . . . . .	52
4.19	Confusion of Source app mutated to Poisson distribution of <i>Viber</i> packet sizes by Random Forest . . . . .	52
4.20	Overhead resulting from mutating to Poisson distribution of <i>Viber</i> packet sizes . . . . .	53
4.21	Prediction of mutated <i>Facebook</i> packet lengths to itself using Normal or Poisson distributions . . . . .	53
4.22	Prediction of mutated <i>Game</i> packet lengths to itself using Normal or Poisson distributions . . . . .	54
4.23	Prediction of mutated <i>WhatsApp</i> packet lengths to itself using Normal or Poisson distributions . . . . .	54
4.24	Prediction of mutated <i>YouTube</i> packet lengths to itself using Normal or Poisson distributions . . . . .	54
4.25	Results of <i>Skype</i> Mutation . . . . .	58
4.26	Results of <i>Facebook</i> Mutation . . . . .	58
4.27	Results of <i>Game</i> Mutation . . . . .	58
4.28	Results of <i>WhatsApp</i> Mutation . . . . .	58
4.29	Results of <i>Viber</i> Mutation . . . . .	59
4.30	Results of <i>YouTube</i> Mutation . . . . .	59
4.31	Confusion of <i>Game</i> mutated into <i>Viber</i> using Our Algorithm . . . . .	59
5.1	Average Execution Time for each instance of optimization for the different applications . . . . .	66
5.2	Optimization Results of <i>Skype</i> Source App using Analytical Solution	68
5.3	Optimization Results of <i>Facebook</i> Source App using Analytical Solution . . . . .	69
5.4	Optimization results of <i>Game</i> Source App using Analytical Solution	70
5.5	Optimization Results of <i>WhatsApp</i> Source App using Analytical Solution . . . . .	71
5.6	Optimization Results of <i>Viber</i> Source App using Analytical Solution	72
5.7	Optimization Results of <i>YouTube</i> Source App using Analytical Solution . . . . .	73
5.8	Results of Apps Mutation using Analytical Solution . . . . .	73
5.9	f1-score of Apps Mutation using Analytical Solution . . . . .	74
5.10	Sample of length mapping of <i>Skype</i> App using <i>OPriv</i> . . . . .	74
5.11	Sample of length mapping of <i>Facebook</i> App using <i>OPriv</i> . . . . .	74
5.12	Sample of length mapping of <i>Game</i> App using <i>OPriv</i> . . . . .	74
5.13	Sample of length mapping of <i>WhatsApp</i> App using <i>OPriv</i> . . . . .	75
5.14	Sample of length mapping of <i>Viber</i> App using <i>OPriv</i> . . . . .	75
5.15	Sample of length mapping of <i>YouTube</i> App using <i>OPriv</i> . . . . .	75

5.16	Results of apps mutation using <i>OPriv</i> . . . . .	75
6.1	Obfuscation effectiveness . . . . .	78
6.2	Degree of traffic masking effectiveness . . . . .	79
6.3	Kullback-Leibler divergence of packet lengths probabilities . . . . .	81
6.4	Kullback-Leibler divergence of IAT probabilities . . . . .	81
6.5	Length Traffic Divergence TD between original and mutated traffic using <i>OPriv</i> . . . . .	82

# Chapter 1

## Introduction

The explosive growth in network applications is perhaps the biggest technical phenomena in recent years. In fact, featured applications have dramatically impacted our societies and play an integral part in our daily lives answering those I-want-to-know, I-want-to-do, I-want-to-go, and I-want-to-buy questions. Apps are becoming the main practice of digital interaction mainly because they are simple and easy to use, and they provide a multitude of new functionalities and attractive features. On the other hand, network applications can easily boost businesses by promoting them and increasing their visibility. Consumers in today's business environment are on the move and they rely on network application platforms. Whether they use mobile phones, tablets, laptops or any other mobile devices, they have all the information they want.

Over the past few years, internet-connected consumer devices have rapidly increased in popularity and availability. It has been shown that the adoption of mobile platforms is increasing at an exponential rate. The prevailing use of these devices has been accompanied by the extensive use of apps. Also, the emergence of the Internet of Things (IoT) networks has connected new types of devices and applications. Therefore, new applications are evolving every day to connect our private resources to the Internet.

The advent of network-based applications is leading to a huge growth and to a tremendous amount of data traffic. Big-data scientists are processing these large amounts of network traffic data to produce broad benefits. In fact, statistical traffic analysis is becoming nowadays an attractive tool of developing algorithms in order to evaluate and manage Internet network traffic. Traffic analysis is the process of extracting high level information from communications even when the actual message data cannot be read. By definition, it is a network engineering technique that consists of examining statistical features of flow packets (e.g., packet sizes, inter-arrival times, packet directions, etc.) and building classifiers using machine-learning algorithms to infer traffic information. Research proved that traffic classification is essential in network security to delineate security strategies, monitor botnet propagation, filter traffic, etc. Network

administrators take advantage of traffic analysis techniques for the purposes of intrusion detection to detect abnormal traffic [1, 2, 3, 4], and to differentiate malicious traffic flows [5, 6]. In addition, a proper deployment of traffic analysis provides valuable insights for control and management of resources in TCP/IP networks through capacity planning, allocation, diagnostic checking, and provisioning [7, 8]. Traffic identification can be used for traffic engineering to optimize the bandwidth allocation and routing plan among applications for different quality of service requirements. Other uses include imposing precise rules on users to access the network in order to apply institutional strategies. Engineers can use this information to build robust networks and avoid possible delays [9]. To this end, traffic analysis is used to support Internet-based services including banking, health, military, government, electrical systems, and transportation.

Traffic analysis can also be misused to launch attacks to infer knowledge on network traffic by means of exploiting side-channel information leakage. The popularity of apps has blurred the gap separating private and unrestricted information about individuals. People usually download applications in line with their habits, religion, health, activities, etc. Accordingly, knowing the specific applications installed by a user can give much personal information about him. To ensure privacy and anonymity in general, apps are implementing encryption and people are resorting to use anonymous communication systems such as Tor [10], JAP [11], etc. However even with the growing usage of these tools to protect traffic content, traffic analysis has become a common attack threatening privacy, anonymity, and confidentiality. These types of attacks have been known for decades causing major privacy breach on military systems, banking, health-care, etc. In fact, an adversary can misuse traffic analysis in order to conclude a user's online activity, which is typically private and includes sensitive information. These privacy breaches, also referred to as 'side-channel information leaks' range from predicting users' locations [12], distinguishing the downloaded web pages [13]; identifying language in an encrypted VoIP conversation [14]; obtaining records of an encrypted VoIP chat [15]; or identifying critical information about the underlying type of applications [16, 17, 18].

One of the biggest worries in networking is to provide perfect security, in particularly user privacy. Accordingly, the security community nowadays is concerned about using network traffic analysis to breach users' privacy [19]. A typical solution to these concerns is to simply encrypt networking data to hide it from a potential attacker seizing the traffic. Yet, this alone is not sufficient; ciphering does not hide all relevant information of a packetized flow. Numerous features of the encrypted network traffic, such as packet lengths, inter-arrival times, direction of packets, their count per flow, and many others, can still give information about the traffic and the exact applications being used. Various studies [20, 21, 22] have demonstrated that an attacker is able to disclose sensitive information about a network application user even in the presence of strong encryption.

In brief, current implementations of network-based applications offer limited



security assurances against traffic analysis. Traffic analysis nowadays is considered a major threat of confidentiality of network applications. Additionally, tools for protecting information traffic privacy still fall short of what is required in terms of strong privacy protection and low performance overhead.

In this Thesis, we take steps to fill this gap and we aim at investigating of privacy protection against traffic analysis to answer the question: can we optimally protect users' privacy without deteriorating the Internet traffic behavior? Our goal is to guard against an eavesdropper monitoring the network and using statistical traffic analysis to infer users' sensitive data. We aim for a traffic obfuscation system that strikes a good balance between efficiency in obstructing application tracing and maintaining minimal performance impact. The proposed solution needs to be efficient in the first place, to have low computational and storage overhead, and to have reduced latency on the traffic. Additionally, it needs to be dynamic, scalable, and able to deal with big data. Figure 1.1 depicts our formal statement of the problem. Moreover, we aim to establish a structured and comprehensive framework to measure and compare between obfuscation systems.

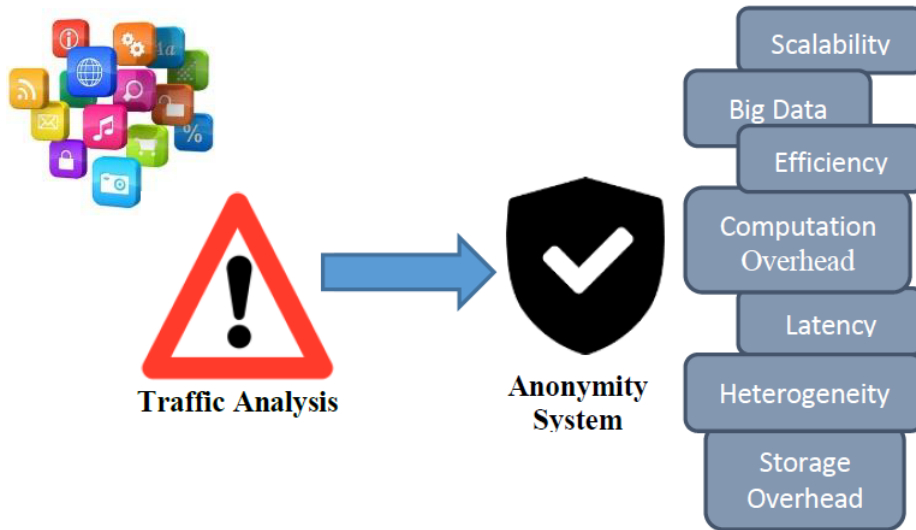


Figure 1.1: Problem Definition

## 1.1 Contributions

The contributions of this thesis can be summarized as follows:

1. Proving a passive traffic analysis attack is possible, involving only sparse information. The attack involves the identification of specific applications using machine learning on encrypted network traffic.

2. Defining formally the problem of privacy against traffic classification, thus finding what the ideal obfuscation system should do.
3. Proposing different obfuscation algorithms by exploring different concepts of anonymization, mutation, morphing, etc.
4. Defining the mathematical formulation of network traffic obfuscation to find an ideal masking algorithm that minimizes overhead cost.
5. Verifying the benefits achieved by an optimal solution. Our experimental evaluation on real dataset revealed that our technique is successful in defending against the attack without degrading users' quality of service.
6. Proposing new information theoretic metrics for quantitative privacy measurement using entropy.
7. Suggesting criteria for selecting tunable parameters to achieve best results of the obfuscation model.
8. Conducting a comparative study between obfuscation algorithms.

## 1.2 List of Publications

My thesis work has resulted in a number of publications as listed:

1. O. Salman, L. Chaddad, I. H. Elhajj, A. Chehab and A. Kayssi, "Pushing intelligence to the network edge," 2018 Fifth International Conference on Software Defined Systems (SDS), Barcelona, 2018, pp. 87-92.
2. L. Chaddad, A. Chehab, I. H. Elhajj, and A. Kayssi, "App traffic mutation: Toward defending against mobile statistical traffic analysis," IEEE INFOCOM 2018- IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, Honolulu, 2018, pp. 27-32.
3. L. Chaddad, A. Chehab, I. H. Elhajj, and A. Kayssi, "AdaptiveMutate: a technique for privacy preservation," Digital Communications and Networks, 2019, 5(4), 245-255.
4. L. Chaddad, A. Chehab, I. H. Elhajj, and A. Kayssi, "Network Obfuscation for Net Worth Security," 2020 Seventh International Conference on Software Defined Systems (SDS), IEEE, Paris, 2020, pp. 83-88.
5. L. Chaddad, A. Chehab, I. H. Elhajj, and A. Kayssi, "Mobile traffic anonymization through probabilistic distribution," 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), IEEE, Paris, 2019, pp. 242-248.

6. L. Chaddad, A. Chehab, I. H. Elhajj, and A. Kayssi, "Optimal Packet Camouflage against Traffic Analysis", ACM Transactions on Privacy and Security, 2021.
7. L. Chaddad, A. Chehab, I. H. Elhajj, and A. Kayssi, "OPriv: Optimizing Privacy Protection for Network Traffic", In review.

### 1.3 Thesis roadmap

This thesis consists of 5 chapters, besides the introduction chapter. These chapters are organized as follows:

- Chapter 2 presents a comprehensive literature review on the traffic classification and obfuscation techniques. In this chapter, we present the different methods of classification. In addition, we review the existing traffic obfuscation techniques. Moreover, in this chapter, we include some necessary background information on the main concepts needed for understanding the rest of the chapters including network traffic properties, and concepts of information theory.
- In Chapter 3, we present our problem definition and the attack scenario that illustrates the need to thwart network traffic classification to provide security. We demonstrate that activities of an application user can be inferred from generated traffic volumes alone, even when the traffic is protected with end-to-end encryption.
- Chapter 4 presents our preliminary algorithms for network obfuscation that we evaluate against traffic analysis attacks in terms of privacy protection, network delay, and traffic overhead.
- In Chapter 5, we present the mathematical formulation of the optimization problem of traffic obfuscation. Then, we present our proposed optimal solution using both analytical and solver packages.
- In Chapter 6, we consider new metrics based on entropy to enhance and compare between obfuscation systems. In addition, we use these metrics to conduct a quantitative privacy assessment of the proposed solution models.
- Finally, we conclude in Chapter 7, with an overall discussion on the findings and limitations. Also, we open the floor for future research perspectives.

## Chapter 2

# From Traffic Analysis to Traffic Obfuscation

Statistical traffic analysis has gained considerable attention from the academic and industrial research communities. It proved to be efficient in identifying security threats and providing effective management of network assets. In fact, studies proved it practical when used for applications identification [23, 24, 25], anomaly detection [26, 27], user fingerprinting [28, 29, 30], etc. Using specific patterns of the packets' sizes and their timing, one can build a classifier based on machine learning techniques to extract the needed information about the network traffic. Yet, network traffic analysis could always be used by a passive adversary and is considered nowadays a major threat with serious impact on users' privacy [19]. In fact, one can use traffic analysis to infer app users' activities from network app traffic metadata. For instance, many features like traffic rates, source and destination addresses, packet sizes, and interpacket times are easily accessible and can be used to give out information about the traffic and the exact applications being used. A typical solution to these concerns is to simply encrypt networking data to hide it from a possible attacker capturing the traffic. However, this alone is not enough; ciphering does not hide all relevant information of a packetized flow. Despite broad adoption of transport layer encryption, traffic metadata of packetized flow is sufficient for a passive network adversary to infer users' app sensitive activities. The attack involves inferring times and types of user application activity from app traffic patterns. Various studies [20, 21, 22] have validated that an attacker is able to disclose sensitive information about a network application user, even in the presence of strong encryption.

In this chapter, we review first Internet traffic analysis methods. A deep understanding of the machine learning classification attack helps in designing a solid security solution. We investigate next privacy protection against Traffic Analysis and the need for the development of thwarting traffic analysis tools. We term this Traffic Obfuscation. In Figure 2.1, we summarize techniques used in both fields.

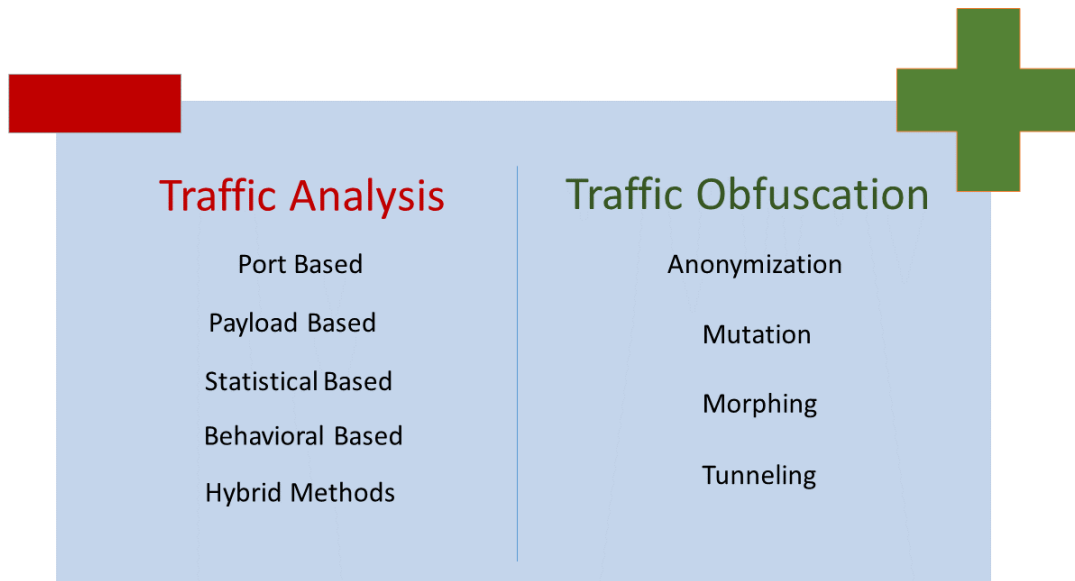


Figure 2.1: Traffic Analysis versus Traffic Obfuscation techniques

## 2.1 Traffic Analysis: State of the Art

Over the last decade, internet traffic classification has been the subject of intensive study given its use in network management and security. Many approaches have been proposed and can be categorized into 4 types: port-based, payload-based, statistical-based, and behavioral-based [31]. In addition, some studies applied hybrid approaches in network traffic classification.

### 2.1.1 Port Based

Port based method is the oldest and most common method for traffic classification. It consists of checking port numbers at the TCP/IP header to identify applications. Then, traffic is classified according to the port used. This method uses only packet headers (or flow identifiers). Essentially, well-known port numbers for specific services are assigned by the Internet Assigned Numbers Authority (IANA) [32]. Port numbers can be easily retrieved and are usually not encrypted, thus making flow classification fast. This method can be used for applications that exploit particular protocols such as HTTP, ICMP, FTP, and POP3. The implementation of a port based classification system is quite simple. However, this method became unreliable and inaccurate because port numbers can be altered and because many applications tend to use dynamic port numbers allocation [33, 34]. In addition, the use of port-based method is very limited especially because many applications do not have their ports registered to IANA.

Additionally, this approach fails on tunnels where NATing process changes source and destination port numbers in the packet header [35].

### 2.1.2 Payload Based

Payload based, known also as deep packet inspection (DPI), consists of inspecting each packet individually to find specific signatures. DPI compares the information extracted from the packets with a set of already known signatures. It entails identifying string patterns of the application and performs classification on this basis. For instance, the authors in [36] present FLOWR that can identify apps with an accuracy of 86-95%. FLOWR learns information in HTTP header that can be used as signature of mobile apps authors. In [37], the authors use raw based features of the content of application layer payload to classify traffic into different application protocols. Some of the well known deep packet inspection tools are nDPI, L7-filter, Libprotoident, PACE, and NBAR, etc. This method is reliable and lightweight and can be used dynamically because it does not include much overhead in terms of processing and storage. Hence, it is widely implemented in commercial products. However, payload-based classification techniques fail when privacy policies and laws prevent accessing to the packet content. That would be the case of encrypted traffic, NAT networks, and Virtual private networks. Also, DPI tools need to be constantly updated and otherwise, they will fail in the prediction of a new protocol. In addition, these methods touch users' privacy.

### 2.1.3 Statistical Based

To overcome DPI and port-based limitations, classification using Machine Learning methods have been proposed [38, 39]. In fact, statistical classification approaches avoid issues related to access need to packet payloads by using payload-independent parameters such as packet length, inter-arrival time, and flow duration. The main idea behind statistical-based methods is to find statistical variations between flows of different networking data classes. The literature presents many classification techniques that identify encrypted traffic using machine learning [40]. These traffic analysis methods leverage the statistical features of network flows and build classifiers based on metadata such as timing, traffic patterns, traffic direction, and volume. The majority of these techniques rely on packet length feature [41, 42, 43, 44, 45, 46, 47, 48]. In fact, packet length reveals characteristics of the underlying class and as such is useful in discriminating between different types of network traffic. In [31], the authors rely on features such as the count of packets and bytes, the minimum, maximum, average and standard deviation of the inter-arrival time of packets in a flow, and the transport layer protocol. Using this technique, authors classify packets with 90% accuracy. However, the detection rate can be affected since inter-arrival time depends on the network. Crotti et al. use in [49] statistical traffic characteristics such as packet size and inter-

arrival time to classify traffic based on the used protocols: POP3, HTTP, and SMTP. The authors in [50] compare five widely utilized machine-learning algorithms (Adaboost, Naïve Bayes, C4.5, Bayesian Network and Naïve Bayes Tree) to classify Internet traffic into different protocols (FTP, Telnet, SMTP, DNS, and HTTP). In [51], the authors build four classification algorithms, namely J48, Random Forest, k-NN, and Bayes Net with a set of 111 features to identify applications (Facebook, Twitter, and Skype). K-NN gives the best results with 93.94% of accuracy with  $k=1$ . In [52], the authors adopt an unsupervised traffic flow classification using time interval based features. Singh et al. build in [53] 5 classifiers and test them on real Internet traffic. Bayes Net was the most successful classifier with 88.13% of accuracy. In [54], the authors study different supervised and unsupervised techniques for traffic classification. Supervised classifiers are based on Naive Bayesian, Neural Networks, and Decision Trees. On the other hand, the clustering techniques studied are DBSCAN, Expectation Maximization based approach, and K-means. The authors in [55] classify encrypted traffic using different machine learning algorithms including Naive Bayesian, Support Vector Machine, C4.5, and Multilayer Perceptron. However, the features used by the machine learning algorithms play the integral role in affecting their accuracy. To address this, deep learning based methods can extract hidden features from raw data packets using a neural network [56]. The authors in [57] present a better Convolutional Neural Network model suitable for unlimited length dataset. In [58], Lotfollahi et al. apply Stacked Auto Encoder machine and Convolutional Neural Network to classify network traffic. Deep learning was applied for the first time to classify Internet traffic by authors in [59] using Stacked Auto Encoder (SAE) method. They analyze the contents of the captured packets of each TCP session (1000 bytes) to discriminate traffic into different applications protocols. The accuracy reaches 99% for some protocols. However, this solution needs a lot of computation because it uses the whole flow payload to make the prediction.

#### 2.1.4 Behavioral Based

Behavioral-based methods attempt to identify patterns among end-to-end communications in a network. This technique calculates the statistical features that characterize the behavior of an application, and then uses these features to build classifiers or clustering methods to identify the class of the application. Behavioral patterns are commonly represented through graph modeling. Graph theory is used to discover connected hosts, number of connections, and ports, etc. [60]. The work in [61] demonstrates an overview of the state of the art in this area. For instance, the work in [62] suggests a technique to find Peer-to-peer communities, where the network connections are characterized using graphs. The nodes represented using their IP addresses and ports, and the connection are characterized by the number of packets exchanged between nodes. Then, a multinomial classifier is built to map a specific graph to known networks. The authors in [63] identify

Peer-to-peer traffic by studying traffic behavior. First, they use K-means model to cluster similar flows together. Then, they characterize the clusters using a Traffic Dispersion Graph, where the nodes are symbolized using their IP addresses and the link between them using the registered flows. At the end, the applications' names are detected using specific rules based on features of the graph. On the other hand, the authors in [64] use the Traffic Analysis Graphs to reveal the behavior of diverse applications. In the graph, the nodes are represented using their IP addresses and the edges are the flows of interest that represent hosts traffic activities for example. Reference [65] constructs bipartite graphs and calculates their similarity matrix. Then, K-means clustering algorithm is built using this matrix. This method can be used to detect traffic anomalies. The authors present in [66] BLINC, a system that uses multi-level behavior of the traffic such as analyzing interaction between hosts, protocol usage, and per-flow average packet size. BLINC develops “graphlets” that describe the normal usage patterns of a variety of network applications; these structures are then used in conjunction with host information to predict the application associated with a given flow independently of the port numbers used. The results show an ability of classifying 80%-90% of the traffic with 95% accuracy. In [51], the authors build a profile of an app using features from traffic traces that associate to an application. Experimental results show that the suggested technique has high accuracy on two P2P applications: BitTorrent and PPLive. Authors in [67] combine both host-level identification and flow-level identification to classify applications using statistical behavior analysis. BitTorrent, HTTP, SMTP, and FTP traffic traces are used in the experiments demonstrating that the proposed technique based on Decision tree can recognize the applications precisely and with small delay.

### 2.1.5 Hybrid methods

The authors in [68] apply a hybrid technique where they combine packet payload signatures and well known port numbers. The main idea is to relate the unknown traffic to known traffic using a co-clustering algorithm combining source IP addresses, destination IP addresses, and destination port numbers. In [69], the authors classify network traffic using hybrid mechanisms including the Extreme Learning Machine, Feature Selection and Multi-objective Genetic Algorithms. They achieve 96% accuracy by using Multi-objective Genetic Algorithms to optimize Extreme Learning Machine classifier and to choose the best feature selection algorithm. In [70], Dong et al. use port-based, payload-based, Bayesian and SVMs methods to build a network classifier achieving over 95% average accuracy.



## 2.2 Traffic Analysis Applications

The issue of identifying applications, by just investigating the generated networking traffic, has been extensively researched over the past years. By identifying patterns in the encrypted traffic, it is possible to conclude users' actions such as spoken phrases [71], web browsing [72, 73], identification [74], motion and location [44], and behavior [43, 45].

At first sight, it appears that traffic analysis on different applications, devices, and services are all the same. While there are some similarities, such as the use of IP addresses/ports for end-to-end communication, there are distinctions in the way and category of traffic. In the rest of this section, we look specifically at traffic analysis methods/attacks on smartphones, and on websites.

### 2.2.1 Traffic Analysis Methods/Attacks on Smartphones

S. Dai et al. [47] present Android apps identification based on Deep Packet Inspection (DPI). Their technique assumes that 70% of apps don't use HTTPS which is not accurate for existing apps in stores. Moreover, applying deep packet analysis on the mobile device would consume battery and CPU resources. The authors of [74] show that the identities of apps used on a smartphone can be inferred by examining 3G/UMTS sidechannel information like volume of data and their timing. In [75], the authors propose a model for classifying service usages of mobile messaging apps by extracting features related to packet length and time delay. In [42], the authors implement a Random Forest classifier using timing and frame size based features to identify mobile apps from their encrypted traffic. In [76], the authors propose to classify the mobile apps traffic based on Convolutional Neural Networks (CNNs) by identifying the abstract app signature from its traffic. Yet, their method is not scalable since it requires the app to use HTTP, which is not the case for most apps. Authors in [17] present a promising methodology for classification of mobile apps' encrypted traffic and have recently extended this work in [41]. They create AppScanner framework based on machine learning algorithms to infer Android apps installed on a mobile device. They use statistical properties of lengths of raw encrypted flows as features. For each flow, they consider 3 series of packet lengths relative to incoming packets, outgoing packets, and bi-directional. Then, they compute for the different series: minimum, maximum, mean, median absolute deviation, standard deviation, variance, skew, kurtosis, percentiles, and the count of elements in the series. This results in 54 features for each flow that can be narrowed down to only 40 as discussed in [17]. They build a random forest classifier and assessed 110 apps from Google Play Store. The results show an average accuracy of 96% in identifying apps outperforming state of the art methods. One drawback of this method is that it uses a considerable number of features making it computationally expensive.

## 2.2.2 Traffic Analysis Methods/Attacks on Websites

A lot of work in the literature aims at identifying the contents of websites using traffic analysis. In this context, an attacker can identify web pages using their side-channel information, and reveal sensitive information of a user, such as their browsing histories. Cheng and Avnur carry out the first website fingerprinting attack in 1998 [77]. They exploit the unique size of each downloaded object to fingerprint the page visited by users. Bissias et al. [78] build profiles for well-known webpage using the packet size and inter-arrival time distributions. Then, they compare using cross correlation the degree of similarity between an observed traffic and the profiles built. They realize a 25% accuracy. Herrmann et al. [13] introduce multinomial naive bayes classifier that correctly identifies sites with an accuracy of 90%. In [79], the authors represent a traffic trace by its cumulated sum of packet sizes. Then, they use Support Vector Machine (SVM) built on discriminative features of the traffic traces. The authors in [72] use packet timing information only on the link. They apply DTW algorithm to classify between traffic traces with time sequences and realize a success rate that exceeds 90%.

## 2.3 Traffic Obfuscation: State of The Art

As outlined previously, extensive experiments and research gave evidence that an encrypted network traffic leaks information about the flow or about the user activity to an observer through detectable features. Consequently, the proliferation of smart networking applications presents unprecedented challenges for preserving user's privacy. There is currently a new research direction focusing on traffic masking to thwart classification in order to protect users' security [80, 81, 82]. Known as traffic watermarking techniques or traffic obfuscation techniques, we review in this section the proposed literature in this field.

Traffic analysis is considered in this case an attack that threatens the user's privacy. Malicious adversaries can employ a passive traffic classification attack to compromise privacy without even being detected by intrusion detection systems. According to the literature, many practices can be used to obfuscate traffic and hinder the ability of an adversary to learn from the network traffic. Different systems have been developed in order to ensure confidentiality, performance, and efficiency of networks. In the following, we group the classification-obfuscation methods into four types: anonymization, mutation, morphing, and tunneling. We classify some of the reviewed work in Table 3.1

### 2.3.1 Anonymization

Anonymization consists of hiding key information that provides important information for traffic classification such as IP addresses, port numbers, MAC ad-

Table 2.1: Traffic obfuscation techniques

Authors	Year	Attacker Model	Accuracy Before	Defense Method	Accuracy After	Defense Approach	Ap-
Fu et al. [83]	2003	Bayes	100%	Dummy packets insertion	50%	Anonymization	
Dyer et al. [84]	2012	Liberatore and Levine Classifier	87%	Maximum Transmission Unit (MTU) padding	41%	Mutation	
Dyer et al. [84]	2012	Herrmann et al. Classifier	98%	Random padding	40%	Mutation	
Dyer et al. [84]	2012	Herrmann et al. Classifier	98%	Random MTU padding	11%	Mutation	
Dyer et al. [84]	2012	Herrmann et al. Classifier	98%	Linear	73%	Mutation	
Dyer et al. [84]	2012	Herrmann et al. Classifier	98%	Exponential	61%	Mutation	
Qu et al. [85]	2012	Naive Bayes	65%	Elephants and mice padding	28%	Mutation	
Wright et al. [86]	2009	Naive Bayes	98%	Morphing using convex optimization techniques	63%	Morphing	
Zhang et al. [48]	2013	SVM and NN	83%	Traffic demultiplexing	44%	Other	
Juarez et al. [87]	2016	K-NN	91%	Adaptive padding	20%	Mutation	
Imani et al. [88]	2018	CNN model	98%	Random padding	60%	Mutation	
Lu et al. [89]	2018	Optimal attacker	94%	DynaFlow	44%	Morphing & Mutation	
Sirinam et al. [90]	2018	Deep Fingerprinting based on Deep learning	90%	BuFLO	12%	Mutation	
Liberatore et Levine [91]	2006	Naive Bayes classifier	98%	packet padding to MTU	7%	Mutation	
Panchenko et al. [92]	2011	Support Vector Machines (SVM)	80%	Camouflage	4%	Anonymization	

dresses. In this context, multi-path routing [93, 94] and NATing [95] can be used for anonymizing the communicated traffic.

Also, TOR is a well-known system for anonymous communication based on the second generation of the onion routing model [96]. The TOR process transmits data between communicating peers via a cascade of *Onion Routers*. Initially, the communication initiator creates shared secret keys with the proxies. The communication initiator encrypts the data in layers, beginning with the key shared with the last node on path of proxies, then in a reverse successive order using the keys shared with the in-between nodes and finally using the key shared with the first node of the path. Then, the first node decrypts the encrypted messages to determine the next hop in the path and sends it the data. The second node repeats this process and sends the data to the third node. This process is reiterated for all the nodes along the path. At the end, the last node in the path decrypts the first layer of encryption and sends the decrypted original data to their planned

receiver. Hence, no node along the path can conclude both the original source and the real intended destination of the data. This scheme is supposed to guarantee anonymity against the peer as well as eavesdropping attackers that could snoop on the traffic flow. However, in the recent past, it has been verified that even such systems are vulnerable to detection by machine-learning techniques [97, 98, 99] which have proved efficient in classifying TOR traffic. Recently, some efforts have been made to improve TOR security by using TOR bridges and Pluggable Transport Protocols [100, 101].

Other proposed anonymization techniques consist of inserting additional dummy packets and transmitting them to conceal the real traffic. For instance, [83] proposes the use of heavy traffic to hinder the adversary’s ability to tamper with the links. Also, the proposed defenses in [102], [103], and [104] deliberately drop some packets, known as defensive dropping, and in [105] inject artificial delay intentionally. In these methods, the authors use dummy packets judiciously in their defense model without drastically degrading performance. Their model detects the packets “on-the-fly” and predicts the traffic characteristics such as throughput and inter-packet arrival times, and consequently sets the rate of dummy traffic injection.

### 2.3.2 Mutation

Traffic mutation relies on changing the flows’ statistical characteristics to confuse a classifier and to make it difficult to identify the original traffic [106]. It consists of properly modifying the packet sizes and/or the packet Inter-Arrival Times (IAT); and consequently, the statistics of the overall conveyed traffic become considerably dissimilar from the original one. Padding and fragmentation are the techniques used to hide packet size information. On the other hand, traffic shaping and buffering aim to hide the interarrival time information. These methods have a great impact in reducing the accuracy of statistically based traffic classifiers. For the packet size mutation, five methods are described as follows:

- ***Maximum Transmission Unit (MTU) padding:*** it is a well-known mutation technique that consists of padding all the flow packets to the maximum payload size MTU [84]. This method completely hides packet size information but creates a huge bandwidth overhead.
- ***Random padding:*** this method consists of padding the packet to a size randomly chosen to be between its current size and MTU [107].
- ***Random MTU padding:*** let  $L$  be the original length, the number of added bytes is randomly selected from 1 to  $MTU - L$  [108].
- ***Linear padding:*** this technique consists of increasing each packet to the nearest multiple of a system parameter [109].

- ***Exponential padding:*** this technique consists of padding the packets to the next largest power of two or the Maximum Transmission Unit (MTU), whichever is smaller. In this case, the transformation graph will have a plateau after a certain value of current length [109].
- ***Elephants and mice padding:*** this technique consists of padding the packets to a certain value  $c$ , if their size is less than  $c$ , if not, they are padded to the MTU [110].

The linear, exponential and mice/elephants techniques showed the worst results in privacy protection. Random MTU padding and Fixed-length have similar results in privacy protection, but the former generates less overhead than the latter [84].

Techniques based on interarrival time mutation are summarized below as listed in [111]:

- ***Constant Interarrival Time:*** this method consists of sending the packets at a fixed interarrival time. this method hides the information carried in the packet interarrival time. However, it presents a very high overhead in terms of latency.
- ***Variable Interarrival Time:*** this technique consists of sending the packets at random time intervals chosen randomly from a uniform distribution between two values.

We depict in Figure 2.2, 2 examples of mutation algorithms: Maximum Transmission Unit (MTU) padding and Constant Interarrival Time.

The authors in [112, 113, 114, 115, 110] consider traffic masking using mutation of packet sizes and interarrival times while realizing a tradeoff between performance and security of the system.

### 2.3.3 Morphing

Morphing techniques aim at confusing the classifier into classifying the target application traffic as another type. We illustrate the concept behind it in Figure 2.3. Wright et al. proposed a morphing technique that consists of transforming one class of traffic to look like another class by applying convex optimization techniques [85, 86]. They evaluate their method against two traffic classifiers for websites [116] and VoIP [41]. Compared to the mutation technique, morphing technique reduces the accuracy of the network traffic classifiers with much less overhead. Nevertheless, there is an issue with the practicality of this method given the needed computations and hence, it cannot be used in devices with restricted resources. Also, it increases latency due to the generation of random numbers for each input packet.

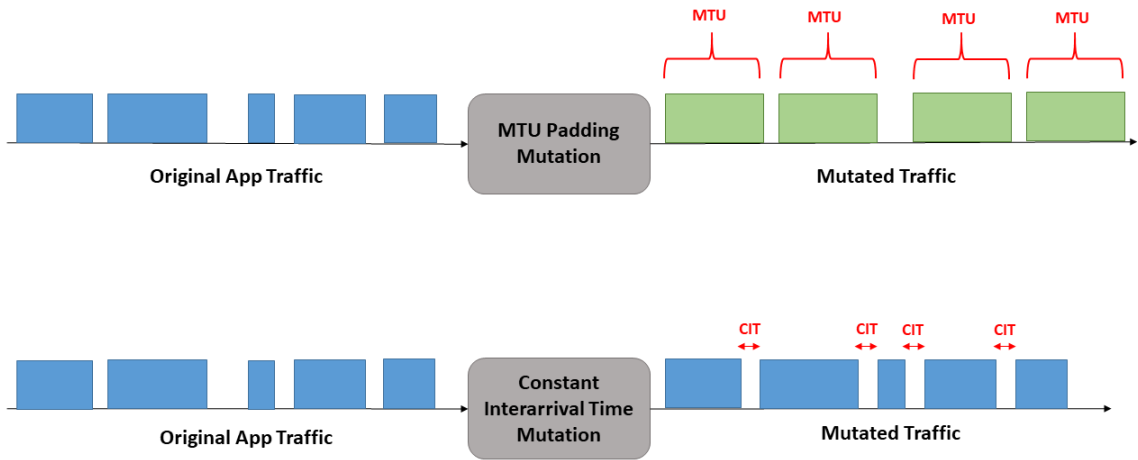


Figure 2.2: Mutation techniques

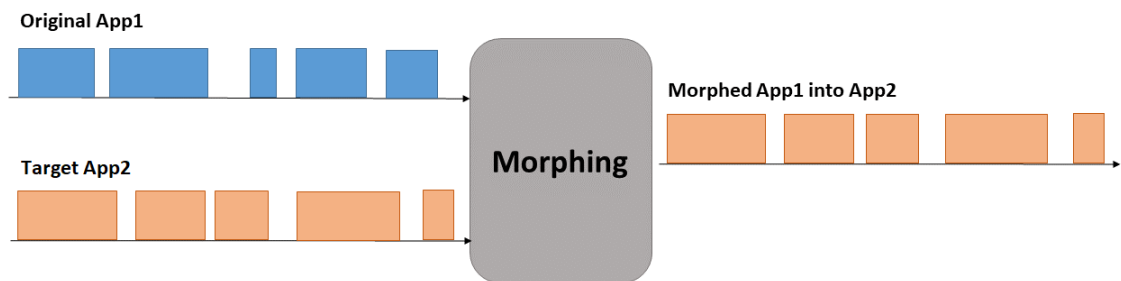


Figure 2.3: Morphing technique

### 2.3.4 Tunneling

Tunneling hides packet-related features (IP addresses, port numbers etc.) by the use of encryption and the creation of virtual networks. Virtual Private Net-

work (VPN) is a well known tunneling service that ensures hiding the connection metadata and consequently the users' privacy. VPN relies on a set of security protocols (e.g., IPSec, IKE, SSL, etc.) to build a tunnel between the VPN client and server, and then the server forwards the client encrypted packets to the designated destination. The source IP address that appears at the destination is the VPN server IP address. However, the work in [117] categorizes VPN traffic based on the application name and traffic type.

### 2.3.5 Other Techniques

In [118], Apthorpe et al. describe a stochastic traffic padding framework to protect smart homes against traffic analysis. It consists of shaping traffic to a fixed traffic pattern, in intermittent periods, to limit the information exposed about users' activities through traffic analysis. Their method obfuscates traffic originating from IoT devices. However, it could result in network latencies. In addition, this technique is not scalable to other types of networking traffic because it is primarily dealing with analysis of user traffic patterns.

[48] suggests using virtualized MAC addresses to obscure the adversary's analysis. Their model consists of creating multiple virtual MAC interfaces over a single wireless card. Then, the packets over these interfaces are dynamically scheduled, and the packet features are reshaped over each virtual interface. However, this solution is more appropriate for using a Wi-Fi connection on a computer.

Other techniques used in traditional networks could also be used for traffic obfuscation such as: jamming [119, 120, 121], identifier-free [122, 123, 124], and pseudonym [125, 126].

## 2.4 Website Traffic Anonymization

Most past studies for traffic anonymization against traffic analysis were specific to defend against website fingerprinting [87, 88, 89, 127, 90]. In fact, one of the first attempts to secure website users from privacy attacks was introduced by Wagner and Schneier [128]. They suggest to apply random length padding on all cipher modes of the SSL protocol. Buffered Fixed-Length Obfuscator [84] is another well-known defense that sends packets at fixed sizes and times, and uses dummy packets. However, it results in a high bandwidth overhead. More advanced techniques have been suggested in [92] where they use camouflage to change the patterns of the data traffic on purpose by loading several web pages instead of the requested one. In [129], a traffic morphing technique (Glove) was proposed. It consists of grouping websites, whose network flows are similar, into clusters. Later, Glove uses minimum dummy traffic to shape websites in a cluster to be the same. So, an attacker cannot classify a specific website, but only the cluster to which it fits. The authors in [130] present the WeFDE technique to

perform information leakage on a dataset. This technique estimates the bits of information that are revealed by particular features of a website traffic.

Recent studies [131] have shown that the applicability of those paradigms are sometimes limited to websites only. According to [132], website-based activities differ from in-app user activities and consequently, the solution for obfuscating the traffic could also be different. Hence, only few solutions of the previous works in obfuscation security, using machine-learning practices, can be used to thwart side channel information leakage specific to network applications.

## 2.5 Discussion: Key Findings and Limitations

It has been well-established that traffic obfuscation is a necessity to preserve network security. Defending against traffic analysis is a relatively young field of research but already has a wide diversity of approaches and systems. Different techniques have been proposed to preserve user privacy by thwarting traffic classification. Mutation and anonymization methods are currently dominant. Their effectiveness is acceptable, however, the large added overhead is not practical since it severely worsens the efficiency and performance of the original network protocols. Padding encrypted packet sizes to their maximum transmission unit (MTU) for example, could end up into more than doubling the amount of data sent [114]. The cost associated with their use is a drastic overhead in terms of the amounts of data being sent. This leads to performance degradation in terms of delay and bandwidth consumption [133].

On the other hand, despite the efforts to obfuscate side-channel information in recent years, it is still possible to conclude networking traffic flows even when implementing these method. In [84], Dyer et al. reviewed techniques known in the literature to defend against traffic analysis and revealed their key weaknesses.

Many works have been presented as promising to reduce the performance of a traffic classifier but their impact on the actual performance degradation has been ignored. To the best of our knowledge, there is no formal statement of the traffic anonymization problem. Most suggested solutions were specific to certain types of traffic such as mobile application, websites, IoT devices, etc. Also, no efficient optimal solution to minimize the overhead cost and implementation complexity to defeat classifiers has been suggested yet.

In brief, the literature lacks a robust yet accurate design for an obfuscation system. Overhead in terms of memory space, computational complexity, and processing time should also be considered as it is vital for real-time network services.



## 2.6 Network Traffic Modeling

The study of packet length distribution is a fundamental step for traffic modeling [134]. A large body of work in the literature studies network traffic properties. In [135], authors report packet size distribution for network flows at the Internet layer, transport layer, and application layer. According to [136], Pareto Second Kind distribution is best fit for network packet inter-arrival time distribution. In [137], the authors find that lognormal and GEV models are the optimal statistical models characterizing Internet traffic. According to [138], network traffic model for non-congested Internet backbone links can be described as a Poisson short-noise process. In [139], the authors demonstrate that models can be used to approximate distribution of network traffic. They suggest a probability density function model to fit packet lengths in computer networks. In addition, they confirm that Exponential, Lognormal, Pareto or Weibull distributions can be used for the same purpose. These clues lead to a conclusion that there are different models with different characteristics that can capture network traffic. Hence, there is not a single model that can be used. Standard goodness-of-fit tests such as Kolmogorov-Smirnov, Anderson-Darling, and Chi-Square [140] allow a mathematical proof for the optimal fit.

## 2.7 Conclusion

Applying Machine Learning in the network domain would enable creative network functions in addition to major threats to its security. Significant amount of previous work has investigated the topic of Traffic Analysis countermeasures. The proposed solutions were limited to specific uses and traffic types. Other network obfuscation proposed solutions did not consider the limited resources aspect in terms of processing and memory assets. Nor did they consider the interactive dynamic nature of network applications traffic. In this chapter, we reviewed traffic classification techniques, as well as the obfuscation methods designed to evade traffic classification. After examining the literature, the question becomes how to design a robust obfuscation framework that accounts for adversarial traffic analysis attacks, while considering network performance concerns. We focus on the accuracy, feasibility, and practicality of the defense model.

# Chapter 3

## Classification Attack

There have been much effort towards designing and deploying trustworthy and reliable network applications that ensure users' privacy and anonymity. However, recent publications have disclosed a variety of Traffic Analysis attacks, in the form of machine learning classifiers that attempt to trace end-user application identification through network traffic classification. In wireless LANs, eavesdropping is easy due to the shared medium and hence, encrypted traffic samples that users send over wireless links are practically exposed to sniffers monitoring the traffic. Even worse, the attacker can have direct access to the encrypted data of a user from the server. This adversary could be, for example, a powerful government organization that gained access to an ISP. Hence, the adversary has the flow features information like timing, size, direction, and count of packets in a specific encrypted network flow. The adversary could use these leaks to induce sensitive information from network contents. Our aim in this Chapter is to demonstrate through experimental evaluation that the privacy of network users is at risk. In the remainder of it, we explain models of network traffic classification based on machine learning, and we present a methodology for the identification of applications using traffic analysis. We perform this attack using side-channel information leakage in which we use the packet-level traffic analysis to infer the precise patterns in the packets no matter they are encrypted or not. At the end, we give the maximum success rate of the classification, and we describe the performance evaluation of the classifiers built.

### 3.1 Traffic Analysis

When sensitive data is transmitted between 2 devices withing a network, it is normally encrypted not to allow others to see it. This process conceals the information within the packet, leaving only ciphertext visible to a possible adversary. Supposing a strong encryption, no one should be able to understand the pro-

tected information as it goes over the Internet. But even if the data is effectively unseen, will the sender be able to preserve a full secrecy? Unfortunately, packet header information must be left unencrypted to ensure successful routing. Those packet sizes, inter-arrival times, and headers leave some information leakage to potential attackers. Figure 3.1 shows the contents of a single packet captured during an encrypted session using the well-known sniffing tool Wireshark. Even if the data is encrypted, some information is still freely obtainable to a possible adversary. Information that can be inferred from this packet are for instance:

- the source
- the destination
- the application-layer protocols in use (SSH, HTTPS, etc.)
- the amount of data enclosed
- the packet timings

```

+ Frame 78 (142 bytes on wire, 142 bytes captured)
+ Ethernet II, Src: 00:24:e8:b0:f2:0c (00:24:e8:b0:f2:0c), Dst: 00:16:01:c
+ Internet Protocol, Src: 192.168.11.5 (192.168.11.5), Dst: 130.127.200.12
+ Transmission Control Protocol, Src Port: 2580 (2580), Dst Port: 22 (22),
- SSH Protocol
  - SSH Version 2 (encryption:aes128-cbc mac:hmac-sha1 compression:none)
    Encrypted Packet: 0DEE7CDAEC39ADBAAA6B9AABD84F2A8420FAFF4285460E62...
    MAC: EA90B1C79DD982D06E2BB32ED9442D7B31240644
  
```

---

```

0000 00 16 01 c7 c2 30 00 24 e8 b0 f2 0c 08 00 45 00 .....0.$ .....E.
0010 00 80 c2 7d 40 00 80 06 21 c1 c0 a8 0b 05 82 7f ...}@... !.....
0020 c8 0c 0a 14 00 16 b6 a1 27 8c 72 37 ae 59 50 18 ..... 'r7.YP.
0030 fd 5b 35 4d 00 00 0d ee 7c da ec 39 ad ba aa 6b .[5M... |..9...k
0040 9a ab d8 4f 2a 84 20 fa ff 42 85 46 0e 62 00 50 ...0*... .B.F.b.P
0050 ec 0f fc 99 8e 96 b7 35 39 28 58 0b 6e 15 36 92 .....5 9(X.n.6.
0060 c9 9b 30 03 35 cf b7 b5 3a b2 af 7c 92 8b 04 84 ..0.5... :.|....
0070 e8 3a 4f 85 99 a0 6e ef 13 42 ea 90 b1 c7 9d d9 ..:O...n. .B.....
0080 82 d0 6e 2b b3 2e d9 44 2d 7b 31 24 06 44 ..n+...D -{1$.D
  
```

Figure 3.1: Packet captured from an encrypted SSH session

Each of these bits of information, mainly when united with similar information from additional packets in a flow, can be revealing about the data within a ciphered message.

## 3.2 Machine Learning based Classification

Before the classification attack is presented, we briefly introduce the machine learning techniques used. Machine learning is a growing branch of Artificial

Intelligence (AI) that emulate human intelligence. Diverse fields have been using machine learning in this new era of Big Data. In fact, it has been shown that it helped in boosting different network activities including:

- **Anomaly detection:** data security and service availability are critically important for operators and end users at the same time. Machine learning techniques can be used for anomaly diagnosis to detect abnormal traffic and to differentiate malicious traffic flows. That would isolate unwanted behavior in the network, prevent malware, and avoid intrusion to sensitive information.
- **Troubleshooting tasks:** the main goal is to find defective network devices, software misconfigurations, trace network faults, and points of packet losses.
- **Control and management of resources:** machine learning can be used in TCP/IP networks for traffic prediction, capacity planning, allocation, bandwidth resource management, diagnostic checking, and provisioning. That would boost maintenance and planning of networks.
- **Trend analysis:** the exact identification of user applications and the analysis of application popularity trends may give useful information, for network administrators to help traffic engineering, and for providers to show services based on user demand.

Machine learning consists of developing an automated system that processes large amount of data to extract information. It creates models to be used later for classification or regression. Some good references for Machine Learning and pattern recognition can be found in [141, 142, 143]. There are three types of machine learning algorithms: supervised, unsupervised, and semi-supervised. Supervised algorithms deals with labeled data, whereas unsupervised algorithms try to discover relations between the inputs without earlier information of the outputs. These relations can be similarities, statistical relationships, proximities, etc. In Supervised learning, training data are paired with their corresponding labels and classes. In Unsupervised learning, classes for each input data are not known. The algorithm finds its own way from the training input data. Supervised techniques are usually used to execute classification tasks, while the unsupervised ones are performed to cluster inputs. In semi-supervised learning, a part of the data is labeled and other parts are not. The labeled part is used to help the learning of the unlabeled part.

A network traffic classification attack based on statistical traffic analysis is a form of supervised machine learning [144]. Its implementation includes two datasets, one for training and the other for testing. Every instance in both datasets is represented by a set of features that have known labels or classes. Four main phases are included in the classification process and illustrated below in Figure 3.2 :

- Data preprocessing
- Features selection
- Learning algorithm
- Validation

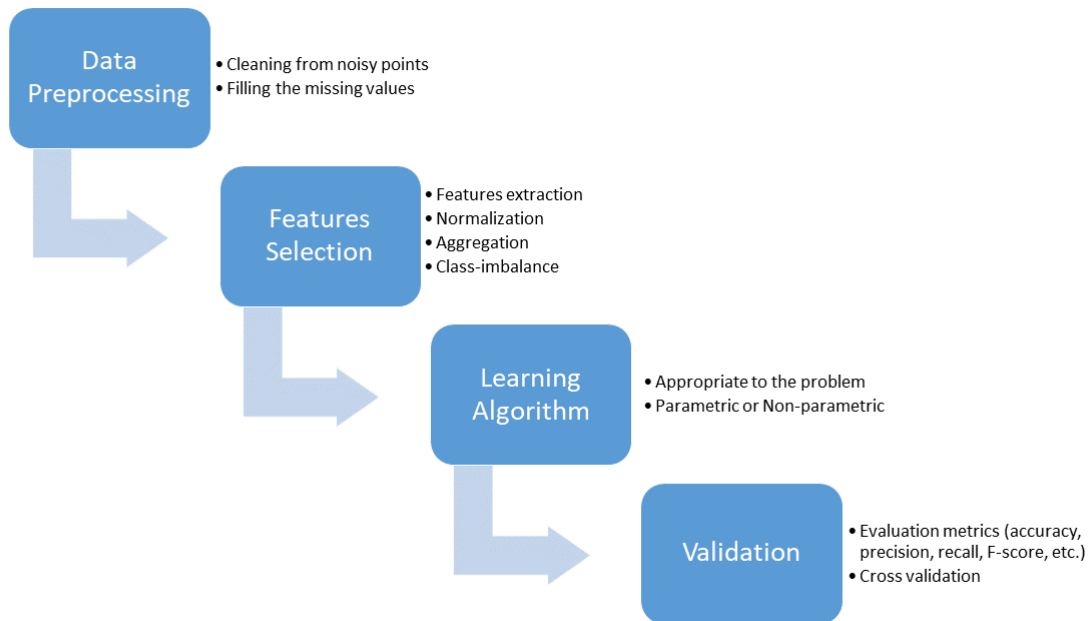


Figure 3.2: Internet traffic classification process

Data preprocessing is the process of cleaning the training dataset from noisy points, and filling the missing values.

Feature selection consists of extracting exact features from the flows of a dataset traffic. By definition, a flow is the traffic exchanged between two IPs with the same ports and protocol during a time period. Feature selection is one of the most important steps where features are measured or calculated. This step results in the computation of feature metrics that reflect specific properties of the collected data. At the end of the feature selection process, a structured table could be constructed where each row is a sample, and attributes are grouped in columns. The class or current status of each sample is a another additional column. At this stage, normalization or aggregation procedures can be applied to combine several features into a single one that would be more expressive to the problem. In addition, new attributes could be created using the original ones to better describe a certain process. These steps are referred to Feature Selection and Reduction.

They help in decreasing time consumption and the curse of dimensionality. They are based on Filter, Wrapper and Embedded approaches [145, 146]. Moreover, class-imbalance could be treated, at this stage, to avoid biasing in case one or more classes are in significant greater amount of samples than the others.

Then, there is the choice of an appropriate learning algorithm to meet the model objectives. This choice is directly related to the type of problem that we are trying to solve. The set of aggregated features is used to build a machine learning classifier, which is the core of the system. Classification methods can be categorized into two types: parametric and non-parametric. For the parametric category, the aim is to find a function  $f$ , such that  $Y = f(X)$ , where  $Y$  is the output (class label) and  $X$  is the features vector. The learning algorithm uses a set of training examples of the form  $(x_1, y_1), \dots, (x_m, y_m)$  for the expectation of a function  $f(x)$ . The  $x$  values are typically vectors of discrete values of the form  $\langle x_{i1}, x_{i2}, \dots, x_{in} \rangle$ . The  $y$  values are the projected outputs for the given  $x$  values, and are normally drawn from a discrete set of classes. Accordingly, the task of the parametric learning paradigm is the calculation of the function  $f(x)$  to create a classifier. On the other side, the goal of non-parametric techniques is to minimize the classification error without inferring the mapping between the input and output.

Supervised learning needs early information of the sample labels, which are necessary to validate the machine learning system. The typical methodology is to divide the dataset into training and testing datasets. The machine learning algorithms are constructed using the training set, while the testing set is used to evaluate the prediction capabilities of the machine learning model. During the training phase, the aim is to feed the classifier with collected data so that it would recognize differentiating patterns, whereas the testing phase would validate the algorithm and compute its performance metrics. Additionally, cross-validation could be used during the training phase to better evaluate the learning model. It consists of dividing the dataset in  $k$  subsets. One subset is used as the test set, and the rest for training the model. The same process is repeated for the  $k$  folds, and the overall performance would be the average of the evaluation score of each test set.

Finally, in order to validate the performance of machine learning classifiers, their classification capabilities could be measured in terms of the number of samples correctly and incorrectly assigned to classes. Common evaluation metrics are Accuracy, Precision, Recall, F-score, etc.

We will present in what follows some background information on well-known machine learning algorithms applied in the networking field and that we will use later on in our empirical evaluation.

### 3.2.1 Decision Trees

In 1986, Quinlan developed Decision trees models that can be used for both classification and regression [147]. Decision trees define rules from training data by answering a series of questions that lead to a class label (leaf of the tree) or a value when applied to any observation. Given a set of collected data features, to construct a decision tree, the feature of high importance is chosen to be at the root of this tree. The feature differentiation can be measured using different indexes such as entropy, Gini index, mutual information, etc. The total number of branches to reach a leaf defines the tree depth. Decision Trees methods can be used for numerical or categorical data [148]. They are very fast and perform well on large datasets. However, decision trees do not always determine an optimal choice at each node model. The best result chosen at each step is the global optimum but not necessarily the optimal decision. Moreover, decision trees are likely to over fit, especially when a tree is particularly deep. We could avoid this problem by setting a max depth at the cost of error due to bias.

### 3.2.2 Random Forest

Random Forest (RF) is a strong modeling type of Ensemble machine learning algorithms that combines multiple learner [149]. It is simply a collection of decision trees whose results are combined into one outcome. RF is one of the techniques that have high performance in traffic classification [150, 151]. Random Forests have the advantage to limit over-fitting without having error due to bias. Also, they decrease error due to variance by training many decision trees on diverse data and feature samples. Another way consists of using a random subset of features in each tree. If many trees are used in the forest, many features will be included. Therefore, error due to bias variance will be decreased.

### 3.2.3 Support Vector Machine (SVM)

Support vector machine is a well-known machine learning method that was developed by Cortes and Vapnik [152] and has been widely used for traffic classification [153, 154, 155, 156, 157]. The algorithm outputs a hyperplane that optimally separates the different classes. To create a hyperplane between two classes, SVM solves a maximization problem of the distance separating instances that belong to the different classes. The problem becomes more complex when there are multi-dimensional feature vectors with multiple classes. In this case, multi-SVM models are built using kernel-based techniques. Different Kernel functions can be specified for the decision function e.g. polynomial, RBF, etc. Consequently, the high computational complexity is a key challenge for SVM algorithms. Another limitation is the high training time needed when the training data size is large. However, SVM presents many advantages such as its applicability to different sort

of classification problems. It is a sparse technique meaning that it does not use the entire data, but it relies on only a subset of it (support vectors). Therefore, it is less memory demanding. In addition, SVM is efficient in high dimensional spaces.

### 3.2.4 K-nearest neighbor (k-NN)

K-nearest neighbor is a simple algorithm introduced by Fix and Hodges in 1951 [158] and can be used for both classification and regression. The principle behind k-NN is to find a predefined number of training samples (k-nearest neighbor learning), closest in distance to the new point, and predict the label from these. The distance can be any metric measure e.g. standard Euclidean distance. K-NN falls under the category of lazy learning, meaning that there is no training phase before classification. The entire training set is kept in memory to perform classifications. This would make the algorithm computationally expensive as it passes through all data points for each classification. Therefore, nearest neighbors tends to work best on smaller data sets that do not have many features. Despite its simplicity, k-NN is easy to interpret and has been a successful algorithm used especially in situations where the decision boundary is very irregular.

## 3.3 Threat Model

Before delving into the attack details, we argue the threat model for which we claim that traffic analysis attacks are effective. Our primary focus is an attacker who can induce a traffic flux in a targeted anonymity preserving channel and observe it leaking a victim user's online activity.

Figure 3.3 shows our threat model. We consider consumers using network-based applications where the program used, or the data related, or both of them reside on a network (often, but not always, the Internet). The adversary can sniff the exchanged data that is usually encrypted between the node and the server. Even worse, the adversary can have physical access to traces of encrypted traffic stored in the data center where data resides. We consider that the adversary cannot decrypt the packets, and aims to discover information about the network users and the used applications. To this end, our adversary applies statistical traffic analysis to find the precise application the user is running, although packet payloads are not readable. In this context, the attacker examines side-channel information (IP packet headers and metadata) from the encrypted mobile app traffic, builds a classifier for several network applications and then, matches the sample traffic captured to infer a user's exact app. This attack is passive and undetectable by users. Our goal is to thwart such information leakage.

Potential adversaries may be incentivized to discover a user behavior, his installed applications, the timing of his application's use. We divide our threat



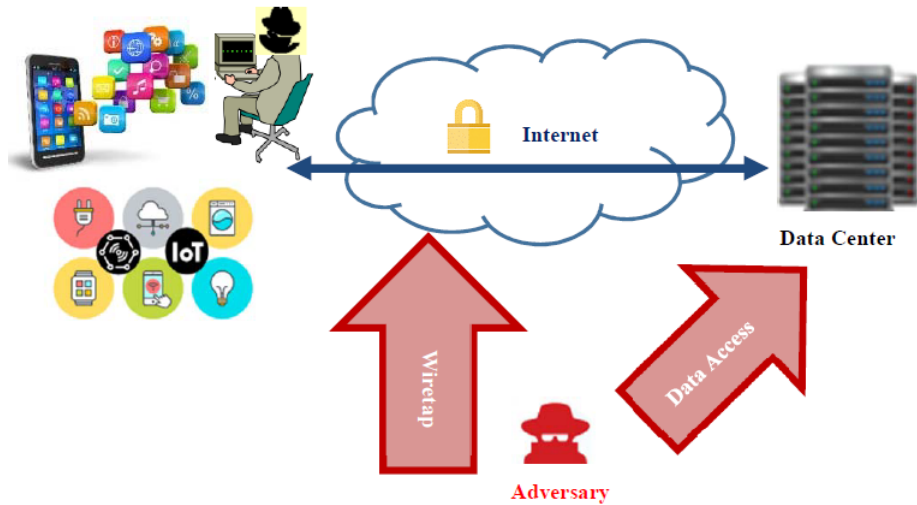


Figure 3.3: Threat Model

model into two different classes of attackers:

- Local attackers: these are attackers that can view and sniff traffic within the same local area network. They include compromised home routers, Wi-Fi eavesdropper by neighbors for example.
- External attackers: these are entities that can view the network traffic created after leaving the local area network. Example external attackers comprise government intelligence agencies, ISPs, and other on-path network spotters.

### 3.4 Classifier Adversary Model

We illustrate in Figure 3.4 an example of traffic classification attacks. We consider an adversary who monitors the Internet connection of a mobile user. The information flow generated by the apps is ciphered and authenticated. The adversary uses sniffer software (e.g. Wireshark) and does not have any knowledge about the software or encryption schemes implemented. Usually, such eavesdropping on network traffic is a passive operation and does not have any directly observable consequences. The classification system collects traffic traces of the user and identifies which mobile network application he is running.

The proposed attack can be divided into two phases: the training and the detection. The training phase consists of features extraction, and then of classifiers training. The detection phase consists of features extraction, and then of detection attack. During features extraction, inputs are raw traces of network traffic applications, and the output is feature vectors. The input and output for a

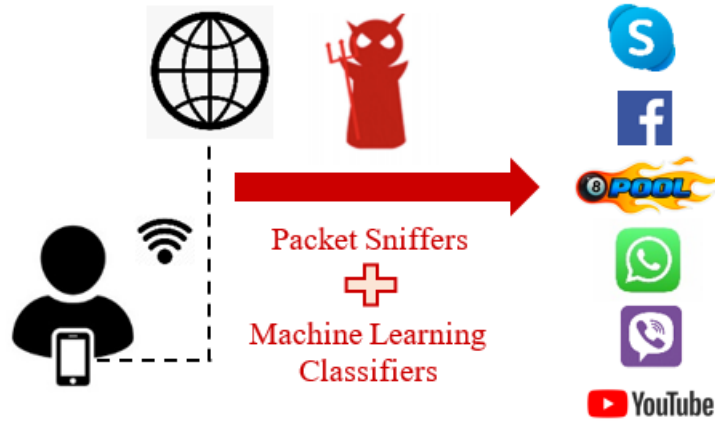


Figure 3.4: Mobile Traffic Classification Attack

classifier training are feature vectors and trained classifiers respectively. Whereas with respect to detection, the inputs are the victim’s raw sniffed packets, and feature vectors generated from a pool of raw network applications traces of interest. The output of this step is the detection result.

## 3.5 Classification Attack Empirical Evaluation

In this Section, the traffic classification attack model is detailed. This includes the dataset used, the classes definition, the features used for classification, and the network classifier models.

### 3.5.1 Dataset

In our app identification experiment, we use a dataset provided by [159]. The data was collected from different types of devices and applications, and consists of descriptions of app traffic flows of six popular network applications collected on Android smartphones and manually classified. *Skype* is considered for chatting and video calling, *Facebook* for social media browsing, *8 ball pool* for online gaming, *WhatsApp* for chatting, *Viber* for VoIP, and *YouTube* for social media browsing. Each packet in the dataset is defined by a set of attributes: the packet number, timestamp, packet length, IAT, direction (IP source/IP destination), and app label.

### 3.5.2 Feature selection

For each of the 6 applications, we extract traffic flows for a fixed time period of 1 second between two IPs with the same protocol and ports. We list the

applications and their differences in Table 3.1, and we label them from 1 to 6. The class names, the size of each class, the actions executed while capturing their traffic traces, their sample proportions, and the count of flows extracted are also described.

Table 3.1: Experiment dataset

#	App	Type	Actions	packets/class	% Packets	# Flows
1	Skype	Video Call	Video Call	197,115	0.350	520
2	Facebook	Interactive browsing	Browse, Comment, Like, Post	122,422	0.218	2,560
3	8 ball pool	Game	Play	121,663	0.217	2,064
4	WhatsApp	Messaging	Texting, Media sharing	3,315	0.006	435
5	Viber	VoIP	Voice Call	82,662	0.147	475
6	YouTube	Video streaming	Play videos	35,027	0.062	813

After extracting the flows, we extract the apps’ features based on packet lengths, IAT, and direction. According to Moore [160], every network traffic consists of 249 attribute features. Since some features contribute less to the real-time classification, and to avoid oversized number of features that would degrade our classification, we choose to implement 27 features as described in [159]. These features are selected based on Wrapper methods along with an analysis of dataset properties and relationships among features [159]. These features are selected to build 4 known supervised machine learning algorithms to construct a template model. We show in Table 3.2 the set of features used in our experiment and their denotation. We denote the incoming packets by PI and the outgoing packets by PO, the packet length by L, and the inter-arrival time by IAT. We use, for each classifier, different combinations of features to achieve the best results as detailed in Table 3.3. The classification aims at identifying the application name, and our goal is to find the model with the maximum accuracy.

### 3.5.3 Model Training and Classification Results

The feature extraction phase generates an imbalanced dataset where the classes are not equally represented. Therefore, we up-sample the minority classes to balance the data and get a total of 11,970 instances to build 4 machine learning algorithms. Then, we split the data into 75% and 25% for the training and testing sets, respectively. We use the ‘one versus one’ strategy for the multi-classification problem, and we evaluate our models using 5 folds’ cross validation. The accuracy of the different classifiers and their corresponding parameters are shown in Table 3.4. The results show that the Random Forest (RF) model achieves the highest accuracy. Table 3.5 shows the confusion matrices of the

Table 3.2: Feature Set for statistical classification

#	Feature	Description
1	PO count	Packets Out Count
2	PI count	Packets In Count
3	PI/PO Ratio	Packets Out/Packets In ratio
4	Bytes-out Count	Bytes Out Count
5	Bytes-in Count	Bytes In Count
6	Bytes out/in ratio	Bytes Out/Bytes In ratio
7	Avg. diff. PI IAT	Average difference of Inter-arrival time of incoming packets
8	Avg. diff. of PI L	Average difference of Lengths of incoming packets
9	Avg. diff. of PO IAT	Average difference of Inter-arrival time of outgoing packets
10	Avg. diff. of PO L	Average difference of Lengths of outgoing packets
11	Median of PI IAT	Median of Inter-arrival time of incoming packets
12	Median of PI L	Median of Lengths of incoming packets
13	Median of PO IAT	Median of Inter-arrival time of outgoing packets
14	Median L of PO	Median of Lengths of outgoing packets
15	Variance diff. PI IAT	Variance of difference of Inter-arrival time of incoming packets
16	Variance diff. of PI L	Variance of difference of Lengths of incoming packets
17	Variance diff. PO IAT	Variance of difference of Inter-arrival time of outgoing packets
18	Variance diff. of PO L	Variance of difference of Lengths of outgoing packets
19	Avg. of PI IAT	Average of Inter-arrival time of incoming packets
20	Avg. of PI L	Average of Length of incoming packets
21	Avg. of PO IAT	Average of Inter-arrival time of outgoing packets
22	Avg. of PO L	Average of Length of outgoing packets
23	Variance of PI IAT	Variance of Inter-arrival time of incoming packets
24	Variance of PI L	Variance of Lengths of incoming packets
25	Variance of PO IAT	Variance of Inter-arrival time of outgoing packets
26	Variance of PO L	Variance of Lengths of outgoing packets
27	IAT of PI bursts	Inter-arrival Time between incoming packets bursts

6 classes for Support Vector Machine (SVM), Bagged Trees (BT), K-Nearest Neighbor (KNN), and RF models. The included numbers are percentages. Based on our experimental results, we can draw the conclusion that the four algorithms realize high classification accuracy and have high true positives for each individual app.

### 3.6 Discussion

It seems intuitive that an adversary could potentially detect private sensitive data using statistical traffic analysis. The empirical evaluation presented in this chapter proved, using real world traces, that network security can be significantly compromised by passive attackers using machine learning. These techniques can be employed by attackers to mount their attacks without even being detected.

We implemented in this chapter techniques that leverage machine learning and traffic analysis to automatically fingerprint and identify smartphone apps.

Table 3.3: Features

#	Feature	SVM	BT	KNN	RF
1	PO count		x	x	x
2	PI count		x	x	x
3	PI/PO Ratio		x	x	x
4	Bytes-out Count		x	x	x
5	Bytes-in Count	x		x	x
6	Bytes out/in ratio	x	x	x	
7	Avg. diff. PI IAT	x		x	
8	Avg. diff. of PI L			x	
9	Avg. diff. of PO IAT		x	x	
10	Avg. diff. of PO L		x	x	
11	Median of PI IAT	x		x	
12	Median of PI L	x	x	x	x
13	Median of PO IAT		x	x	x
14	Median L of PO			x	
15	Variance diff. PI IAT		x	x	
16	Variance diff. of PI L	x	x	x	
17	Variance diff. PO IAT		x	x	
18	Variance diff. of PO L		x	x	
19	Avg. of PI IAT		x	x	x
20	Avg. of PI L	x		x	x
21	Avg. of PO IAT			x	
22	Avg. of PO L	x		x	
23	Variance of PI IAT		x	x	
24	Variance of PI L	x		x	
25	Variance of PO IAT			x	
26	Variance of PO L		x	x	x
27	IAT of PI bursts	x		x	

Table 3.4: Dataset accuracy

Model	Parameters	Accuracy
SVM	Quadratic Kernel	74.7 %
Bagged Trees	Min Leaf Size= 2	90.0 %
KNN	number of neighbors= 5 Distance Weight = Squared inverse	83.9 %
Random Forest	Min Leaf Size=2	91.1 %

We adapted for this reason 4 different machine learning algorithms (SVM, BT, KNN, and RF), and we performed traffic analysis attacks against users' privacy. We evaluated these methods using a real dataset collected over different types

Table 3.5: Confusion matrices for SVM, BT, KNN, and RF

Class	1		2		3		4		5		6	
1	90	94	4	1	1	<1	4	5	<1	0	0	<1
	>99	96	0	1	<1	0	0	3	0	0	0	0
2	14	1	65	84	1	2	8	6	1	<1	10	6
	6	3	76	82	<5	1	7	6	1	1	5	6
3	14	<1	2	2	77	87	4	6	1	0	10	4
	11	1	5	2	79	86	2	7	<1	0	2	4
4	25	<1	14	2	1	3	52	88	1	0	6	6
	15	0	6	0	0	2	79	92	<1	0	0	4
5	6	0	2	<1	0	0	3	<1	89	99	0	0
	<1	0	0	0	0	0	0	<1	>99	>99	0	0
6	8	<1	8	4	5	5	4	3	<1	0	74	88
	6	1	23	4	1	6	1	2	<1	1	69	86

SVM
  BT
  KNN
  RF

of devices, and from six popular network applications (*Skype*, *Facebook*, *8 ball pool*, *WhatsApp*, *Viber*, and *YouTube*). Then, we extracted 27 features from the dataset to build our 4 machine learning classifiers. We split the data into 75% and 25% for the training and testing sets, respectively. We used the ‘one versus one’ strategy for the multi-classification problem, and we evaluated our models using 5 folds’ cross validation. RF realized the highest accuracy of 91.1% followed consecutively by BT, KNN, and finally SVM.

It is worth noting that some classification approaches performed better than others in terms of accuracy, precision, recall, etc. Also, some apps themselves performed better than others when being classified. In that case, these apps reflect very distinct traffic flows that would characterize the specific app.

The presented frameworks are robust and scalable in the identification of smartphone apps from their network traffic. Also, the smartphone landscape offers exceptional challenges to the attack model, such as less available features, and the requirement for scalability and automation. Based on the results of the classification experiment presented, we confirmed that the privacy of the user is at risk due to the usage of apps on smartphones.

### 3.7 Conclusion

In this chapter, we showed that traffic analysis attacks can greatly compromise privacy of network applications. We presented a practical traffic analysis attack against mobile applications, that relies on using machine learning algorithms. Our experiments clearly validated that a malicious eavesdropper can identify the

users' apps, even if packets are encrypted, based only on examining the IP packet headers and metadata, and using machine-learning techniques. We presented classifier models created on training data that holds packet-level statistical information of a defined number of applications. By using this method, we classified specific apps with true positive rate of 91.1%. We believe that our experiments exposed a real weakness in supposedly protected anonymity schemes. The classification techniques described in this chapter will be used later on through this work to evaluate information leakage.

# Chapter 4

## Obfuscation for Better Secrecy

We focused in the previous chapter on our traffic analysis attack to confirm the identification of network applications within the anonymity set. In this Chapter, we detail our preliminary proposed obfuscation models that provide secrecy for an encrypted app’s network activities against traffic analysis. Our mutation techniques would alter the traffic statistical features, making it very challenging to know the original traffic type. Initially, we introduce *AdaptiveMutate* with 3 variations. The main idea behind this model is to mutate a packet feature from the source app to protect, to its similar corresponding feature from a target app that the source app will resemble. Next, we describe a system that regenerates statistical modeling of an app packet length, and we suggest an obfuscation model that mutates packet lengths of the incoming traffic to the regenerated ones. In the final part, we discuss our third framework that mutates the packet lengths of a source app to those lengths from the target app having similar bin probability. Finally, we conclude this chapter with a brief discussion regarding the results, highlighting the techniques that worked, and those that did not. We evaluate the effectiveness of the proposed schemes on real traffic dataset and we refer to the attack model described previously in Section 3.4 in all our obfuscation experiments.

### 4.1 *AdaptiveMutate*

As discussed previously, the key features of an encrypted traffic are the packet sizes and the interarrival delay of successive packets. Hence, an efficient obfuscation system mainly needs to reproduce these characteristics for an encrypted channel. Our design is focused on mutating both features. We aim to modify the packet size (padding) and IAT (shaping), in such a way to hide any information that serves for traffic detection or attack classification. First, we discuss mutating each feature separately; and then, we combine the mutation of both features.



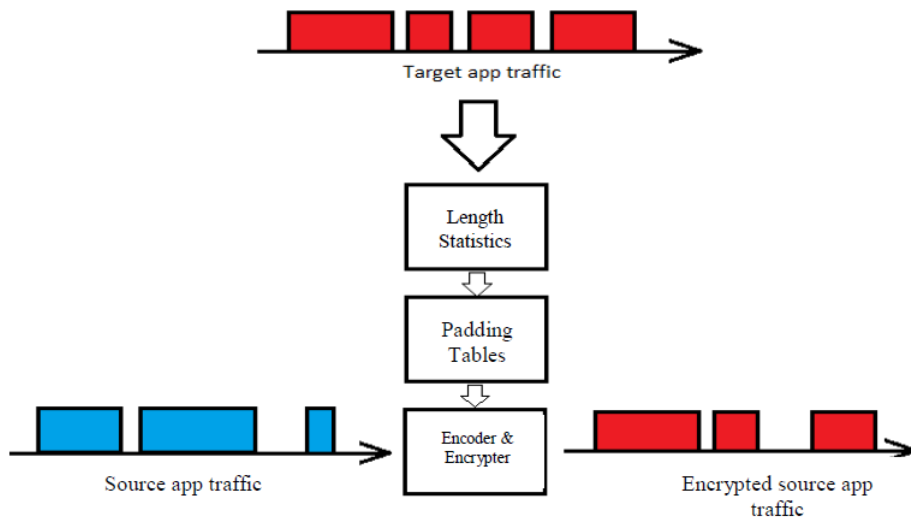


Figure 4.1: Block scheme of mutation operation

### 4.1.1 Packet Length Mutation

The mutation covers the entire packet flow transported between two endpoints. To confuse an app traffic, we change the flows generated by one app so that they look like the flows generated by another app. This entails resampling the packet lengths from the probability distributions of a target app traffic that are independent of the original packet lengths generated by the source app. In other words, our padding modifies the packet lengths in a way that the resultant lengths of the packets seem as if they are generated by the target app probability distribution function.

We assume a general setting illustrated in Figure 4.1. First, the user selects the source app to defend, as well as the target app that the source app should resemble. We get the probability distribution with respect to the packet lengths of a flow from the target app traffic. We prove in Subsection 4.1.4 that flows of the same app have similar length probability distribution. For each length in the target app, we save its corresponding probability in “length probability tables”.

The algorithm consists of modifying, at each run, the first  $n$  packets of an incoming flow from the source app. We store the fixed  $n$  packets from a flow to modify them in a buffer. Based on the length probability distribution of the target app calculated previously, we consider the smallest length as “1” in the target app. We calculate the number  $m$  of packets among  $n$  that needs to be modified to “1”. This number,  $m$ , is the probability of length “1” in the target app multiplied by  $n$ . Then, we choose the smallest  $m$  packets among  $n$  and shape them to “1”. Similarly, we mutate the remaining  $(n-m)$  packets to ensure that the smallest packets of the source app are shaped to the smallest packets in the target app. To mutate each incoming packet from the source app, we compare its size,

$s_1$ , with the size,  $t_1$ , that we need to mutate to. If  $s_1$  is less than  $t_1$ , we proceed by padding  $s_1$  with zeroes so that the resultant packet length is  $t_1$ . Practically, in real-time network applications, we set a flag  $F$  to indicate whether the packet is padded ( $F = 1$ ) or not ( $F = 0$ ). The packet also contains a field member which the receiving endpoint can use to determine the number of the added padding zeroes. Otherwise ( $s_1 > t_1$ ), we fragment the packet into two parts: the first part has the same size as  $t_1$ , and the second one is considered as a new incoming packet. This would guarantee that the length probability characteristics of the mutated source app traffic is kept the same as that of the target app, which achieves the morphing of lengths. From the analysis of the resultant packet stream, the estimation of the packet length statistics shows that the probability distribution of the resultant packet lengths is the same as that of the target app. It is worth noting that it is preferable to choose a target app in such a way so that most of its packet lengths are larger than those of the source app. Then, the algorithm would switch to padding more than fragmenting. This ensures that this scheme is transparent to the passive monitor since the traffic rate will be less affected. Our padding mechanism acts as a proxy between the mobile apps and the access point, so that it intercepts the data sent after the encryption is performed. Then, the modified data is sent by the network encryption. Packets are deciphered at the end and the padding is removed so that our system is transparent to the end-users. This algorithm works in real-time, and reduces the resultant padding overhead and effective classification by avoiding any leak of information about the source app packet lengths.

We have considered the app traffic mutation to thwart traffic classification by partially morphing the source app into a target app. We could consider the app traffic mutation, without morphing, to only prevent any leak of information about the source app packet lengths and hence to prevent effective classification. This would reduce the resultant padding overhead. Another variation of the algorithm is to modify the beginning of each flow from the source app, with the altered packets constituting the first 10% of each flow. The difference between these two versions is the number of packets mutated. As the number of mutated packets decreases, the thwarting algorithm becomes less effective, but the traffic overhead decreases significantly. This partial obfuscation allows to lessen the amount of overhead and delay at the cost of leaking some information about the network flow content. Any one of these models can be selected based on the preference of the user for more security or less overhead.

#### 4.1.2 Inter-arrival time mutation

Even though the primary goal of our work is to make the output of mutation converge in distribution to that of the target app, it considers only the lengths of packets of the encrypted traffic in the previous subsection. We neglect one key element of the encrypted traffic from our design scope, namely, the Inter-Arrival

Time (IAT) between consecutive packets. The same approach elaborated in Subsection 4.1.1 is applied for the mutation, but the inter-packet delay becomes now the design focus. We aim to achieve the same goal of confusing an app traffic by changing the IAT between packets. Our scheme starts from a set of mobile network traffic packets that represent the typical flow of a target app. We find the probability distribution with respect to the IAT of the target app traffic. For each IAT in the target app, we save its corresponding probability in the “IAT probability tables”. The algorithm consists of modifying at each run the first  $n$  IAT of an incoming flow from the source app. We hold the first  $n$  packets from a flow to modify them in a buffer for a sufficient amount of time. Based on the inter-packet delay probability distribution of the target app, we consider the smallest inter-packet delay as “ $l$ ” in the target app. We calculate the number  $m$  of the time intervals among  $n$  that needs to be modified to “ $l$ ”. This number,  $m$ , is the probability of IAT of “ $l$ ” in the target app multiplied by  $n$ . Then, we choose the smallest  $m$  IAT among  $n$  and shape them to “ $l$ ”. Similarly, we mutate the remaining  $(n-m)$  time intervals to ensure that the minimum IAT of source app is shaped to the minimum IAT in target app. At every run, we hold  $n$  packets for a sufficient time equal to the maximum IAT,  $IAT_{max}$ , multiplied by  $n$ . Each incoming packet from the source app is sent with the corresponding calculated inter-packet delay. This guarantees that the inter-packet delay probability characteristics of the mutated source app traffic is kept the same as that of the target app, which achieves morphing of inter-packet delays. This scheme is transparent to the receiver since the information will not be padded.

### 4.1.3 Lengths and IAT mutation

This scheme extends our algorithm to fully represent the characteristics of the encrypted networking traffic by combining both lengths and IAT mutation. Our goal is to make the target traffic less distinguishable from the source traffic by leading the classifiers to misidentify the traffic as a class of our choosing. This achieves the morphing of a source class to a target class. The same approaches of mutating length and IAT, discussed in Subsections 4.1.1 and 4.1.2 are applied consecutively.

### 4.1.4 Practical considerations

We aim to identify, for each app, the most dissimilar app corresponding to it to maximize the confusion between apps after mutation. The cosine similarity measure is a classical measure to represent the relationship between two sets, and it is widely used as a measure of similarity. It is defined as the inner product of two vectors divided by the product of their lengths. Its geometric meaning is the cosine of the angle between the two vectors. The cosine similarity measure

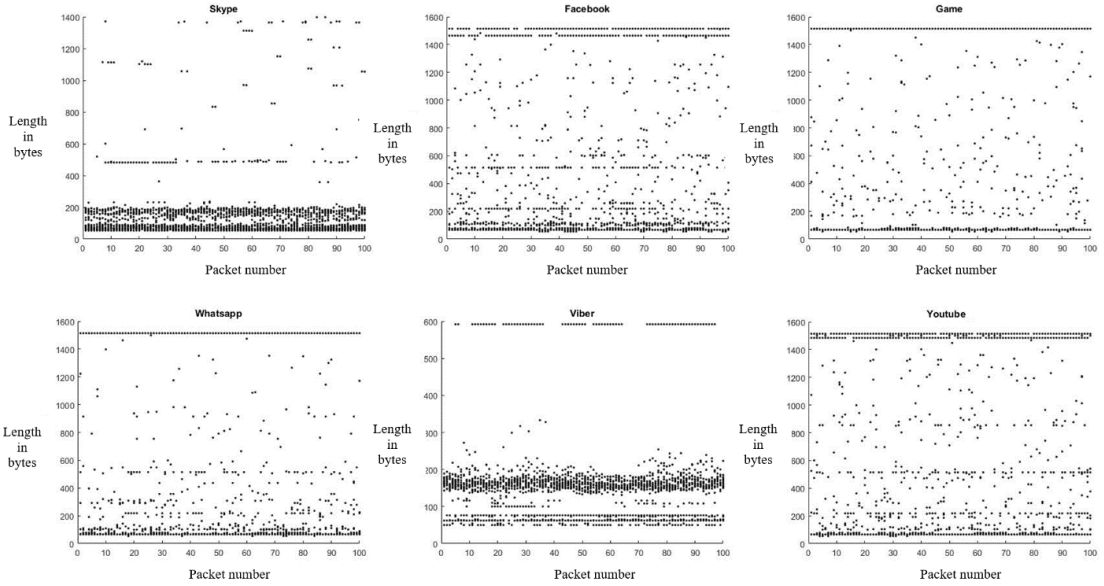


Figure 4.2: Lengths in bytes of the first 100 packets of 20 different flows in app traffic

between 2 sets A and B is defined as:

$$\text{Cos}(A, B) = \frac{A \cdot B}{|A||B|} = \frac{\sum_{i=1}^n A_i \cdot B_i}{\left(\sqrt{\sum_{(i=1)}^n A_i^2} \sqrt{\sum_{(i=1)}^n B_i^2}\right)} \quad (4.1)$$

where  $A_i$  and  $B_i$  are components of vector A and B, respectively.

The resulting cosine similarity ranges from 1, indicating the same, to -1, meaning the opposite; while the value of 0 indicates decorrelation, and in-between values represent intermediate similarity or dissimilarity.

First, we demonstrate that all flows in a specific app are similar and follow the same distribution. Figure 4.2 and Figure 4.3 show, respectively, the lengths in bytes and IAT in seconds of the first 100 packets in each flow for 20 different flows in each app. Most of the graphs depict how closely the flows distribution are similar, both for packet lengths and for IAT. In fact, dense samples indicate that lengths and IAT are taking precise overlap values. Hence, each flow in each app is similar to the other flows in the same app as mentioned in [161, 162, 163]. To calculate the cosine similarity between apps, we pick a random flow from each app. Lengths' graphs of *WhatsApp*, *Game*, and *YouTube* show relatively higher scatter than other apps. This could lead to a higher obfuscation, no matter which flow is chosen, since the range of lengths to mutate is bigger and can therefore confuse the classifier further.

Table 4.1 represents cosine similarity measures between the lengths' vectors of flows from two different apps. Similarly, Table 4.2 cosine similarity measures

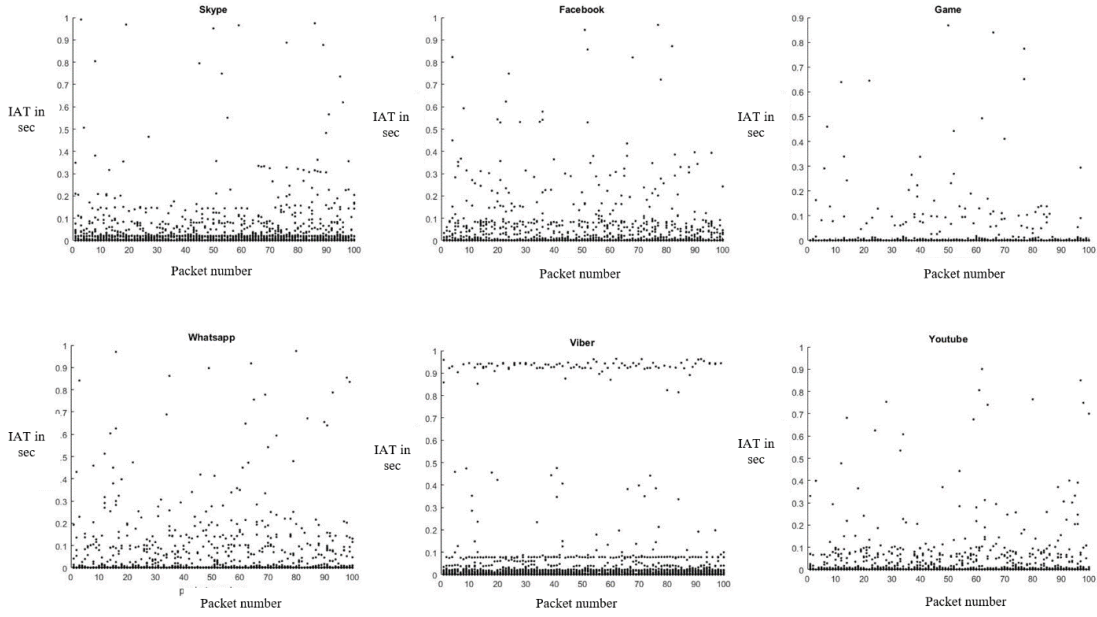


Figure 4.3: IAT in seconds of the first 100 packets of 20 different flows in app traffic

Table 4.1: Cosine similarity of length vectors

Source app \ Target app	Skype	Facebook	Game	WhatsApp	Viber	YouTube
Skype	1	0.4597	0.4484	<b>0.3846</b>	0.6185	0.5153
Facebook	0.4597	1	0.4540	<b>0.3374</b>	0.5869	0.4987
Game	0.4484	0.4540	1	<b>0.4176</b>	0.667	0.5247
WhatsApp	0.3846	<b>0.3374</b>	0.4176	1	0.5509	0.3915
Viber	0.6185	0.5869	0.667	<b>0.5509</b>	1	0.6478
YouTube	0.5153	0.4987	0.5247	<b>0.3915</b>	0.6478	1

Table 4.2: Cosine similarity of inter-arrival time vectors

Source app \ Target app	Skype	Facebook	Game	WhatsApp	Viber	YouTube
Skype	1	0.00012	0.00007	0.00007	<b>0.00004</b>	0.6740
Facebook	0.00012	1	0.0714	<b>0.00016</b>	0.0040	0.00084
Game	0.00007	0.0714	1	<b>0.000001</b>	0.004	0.00008
WhatsApp	0.00007	0.00016	<b>0.000001</b>	1	0.00026	0.00037
Viber	<b>0.000042</b>	0.0040	0.004	0.00026	1	0.00019
YouTube	0.6740	0.00084	<b>0.00008</b>	0.00037	0.00019	1

between the vector of IAT of flows from two apps. Cosine values that are most dissimilar, with respect to the designated feature, are highlighted in boldface. To maximize its efficiency, our algorithm can dynamically adjust the mutation process. Given the source app, it chooses the corresponding most dissimilar app

suitable to mutate in terms of lengths or IAT.

### 4.1.5 Obfuscation experiment and results

In this part, we empirically evaluate the efficiency of our thwarting technique, *AdaptiveMutate*, against the four traffic classifiers discussed in the previous Chapter. We present three experiments corresponding to three different obfuscation models supported by *AdaptiveMutate*, where we first apply the mutation of packet lengths, then the mutation of IAT, and finally the mutation of both features at the same time. Our evaluation focuses on the ability of our tool to reduce the accuracy of the classifiers.

In what follows, we conduct a comprehensive simulation where we mutate each of the 6 apps to the 5 other apps using different modes of mutation. In this first experiment, where we mutate lengths, we consider 2 versions of the algorithm. For each mutation of a source app to a target app, we start from a set of mobile network traffic packets that represent the typical flow of a target app. We calculate their lengths probability distribution. For each  $n = 1000$  packets of an incoming flow from the source app, we run the algorithm. Based on the length probability distribution table, our algorithm selects successively the smallest lengths among  $n$  that need to be mutated to the corresponding smallest ones in the target app. We compare the new packet size with the old one, and accordingly, we proceed by either padding zeroes or fragmenting. In the second version of this algorithm, we only mutate the first 100 packets of every flow. The results of the 2 versions of length mutation are shown in Table 4.3 and Table 4.4. In these tables, the cells corresponding to an app mutated to itself denotes that no mutation has happened.

Table 4.3: *AdaptiveMutate* results using first version of packet length mutation

Target app \ Source app	Skype		Facebook		Game		WhatsApp		Viber		YouTube	
Skype			30.5 %	14.2 %	18.6 %	15.7 %	17.2 %	14.1 %	30.2 %	17.1 %	19.9 %	15.4 %
Facebook	26.6 %	15.8 %	28.2 %	16.9 %	22.5 %	17.6 %	24.6 %	17.4 %	21.8 %	20.3 %	22.8 %	16.8 %
Game	23.9 %	17.9 %	18.9 %	15.6 %	24.1 %	17.7 %	23.9 %	17.4 %	25.6 %	18.2 %	24.3 %	17.9 %
WhatsApp	18.9 %	15.6 %	19.0 %	15.4 %			18.8 %	14.3 %	31.1 %	17.9 %	19.8 %	15.7 %
Viber	23.5 %	17.2 %	24.9 %	17.5 %			23.9 %	17.1 %	28.1 %	20.5 %	22.7 %	16.4 %
YouTube	17.3 %	14.2 %	17.2 %	13.6 %	18.7 %	14.4 %			20.1 %	15.6 %	17.1 %	14.1 %
Skype	24.2 %	17.1 %	23.7 %	17.5 %	23.7 %	17.2 %			22.8 %	16.9 %	24.4 %	17.6 %
Facebook	30.1 %	17.3 %	29.4 %	16.1 %	31.0 %	17.9 %	20.2 %	15.5 %			30.9 %	17.7 %
Game	21.9 %	20.4 %	25.4 %	18.1 %	28.2 %	20.7 %	22.9 %	16.8 %			28.0 %	19.8 %
WhatsApp	19.8 %	15.3 %	19.9 %	16.1 %	19.8 %	15.7 %	17.1 %	14.1 %	31.1 %	17.6 %		
Viber	22.9 %	16.7 %	24.5 %	17.8 %	22.7 %	16.4 %	24.4 %	17.6 %	27.8 %	19.9 %		

■ SVM ■ BT ■ KNN ■ RF

In our second experiment, we apply *AdaptiveMutate* using IAT mutation to mutate each one of the 6 apps to another app. For each mutation, we calculate the IAT probability distribution of a flow from the target app. At each run, we buffer 1000 packets for a time equal to 1000 times the maximum IAT among

Table 4.4: *AdaptiveMutate* results using second version of packet length mutation

Source app \ Target app	Skype		Facebook		Game		WhatsApp		Viber		YouTube	
Skype			31.4 %	36.2 %	31 %	37.6 %	30.8 %	35.4 %	32.2 %	37.5 %	30.5%	37.8 %
			43.2 %	31.8 %	45.8 %	35.1 %	43.3 %	36.8 %	45.9 %	39.2 %	42.3 %	33.9 %
Facebook	31.3%	36.3 %			31.1 %	36.6 %	30.6 %	34.9 %	31.9 %	37.9 %	31.7 %	36.8%
	43.2 %	31.7 %			45.9 %	35.0 %	43.2 %	32.5 %	42.3 %	34.1 %	42.1 %	32.6 %
Game	31.1 %	37.7%	31.0 %	36.7 %			30.8 %	35.8 %	34.2 %	38.1 %	31 %	37.6 %
	45.7 %	35.0 %	45.9%	35.1 %			46.5 %	32.9 %	43.1%	35.4%	42.4%	34.0%
WhatsApp	30.6 %	35.3 %	30.7 %	34.8 %	30.7 %	35.9 %			31.8 %	38.0 %	32.1 %	38.1 %
	43.0 %	36.5 %	43.1 %	32.4 %	46.6 %	32.7 %			42.5%	34.0%	42.4%	34.0%
Viber	32.1 %	37.3%	31.8 %	37.8 %	34.1 %	38.3 %	31.8 %	38.0 %			34.3 %	38.2%
	45.8 %	39.3%	42.4%	34.2 %	43.2 %	35.3 %	42.5%	34.0%			43.0%	35.6%
YouTube	30.6 %	37.7 %	31.9 %	36.7 %	31.1 %	37.7%	32.2 %	38.0 %	34.2 %	38.2%		
	42.2 %	34.0 %	42.2%	32.5 %	42.2 %	34.1 %	42.5%	34.1%	43.1%	35.7%		

SVM
 BT
 KNN
 RF

the 1000 IAT. Based on the IAT probability distribution table, our algorithm selects successively the minimum IATs among  $n$  that need to be mutated to the corresponding ones in the target app. Then, each packet is sent according to the calculated IAT. The results are illustrated in Table 4.5.

Table 4.5: *AdaptiveMutate* results using mutation of IATs

Source app \ Target app	Skype		Facebook		Game		WhatsApp		Viber		YouTube	
Skype			58.1 %	65.8 %	52.4 %	65.1 %	51.8 %	63.5 %	51.7 %	66.2 %	58.1%	69.9 %
			51.2 %	60.4 %	54.8 %	62.3 %	54.3 %	62.5 %	54.2 %	62.8%	55.7 %	65.3 %
Facebook	58.0%	65.9 %			53.6 %	65.3 %	51.9 %	66.2 %	51.9 %	68.1 %	52.0 %	66.9%
	51.3 %	60.2 %			55.0 %	62.3 %	54.4 %	63.0 %	54.3 %	62.7 %	54.8 %	62.4 %
Game	53.3 %	65.1%	52.5 %	65.0 %			51.6%	66.1 %	51.8 %	66.3 %	52.2 %	66.0 %
	55.2 %	62.2 %	54.9%	62.5 %			54.2 %	54.1 %	54.2%	62.9%	54.6%	62.3%
WhatsApp	51.9 %	63.5 %	51.9 %	66.3 %	51.8 %	66.0 %			53.6 %	67.1 %	53.2 %	66.2 %
	54.4 %	62.7 %	54.6 %	63.1 %	54.3 %	54.0 %			55.8%	62.5%	54.2%	63.2%
Viber	51.8%	66.3%	51.9 %	68.0 %	51.9 %	66.5 %	53.7 %	67.0 %			51.9 %	65.9%
	54.1 %	62.9%	54.0%	62.6 %	54.3 %	62.8 %	55.5%	62.7%			54.3%	61.3%
YouTube	58.3 %	69.8 %	52.1 %	66.7 %	52.1 %	66.1%	53.1 %	66.1 %	51.7 %	65.8%		
	55.6%	65.2 %	54.9%	62.5 %	54.5 %	62.5 %	54.3%	63.4%	54.1%	61.5%		

SVM
 BT
 KNN
 RF

In our third experiment, we apply *AdaptiveMutate* using both lengths and IAT mutation to mutate each app among the 6 apps to another one. Using our 4 model classifiers, we show in Table 4.6 the prediction accuracy for the modified traffic. Also, we calculate the overhead resulting from padding in Table 4.7. In Table 4.8, we calculate the performance time delay parameter resulting from mutating the IAT. This parameter is defined as the ratio of the difference between the sum of resulting IAT after and before mutation, and the sum of IAT before mutation.

In Table 4.3, the simulation results show that *AdaptiveMutate*, using the first version of length mutation algorithm from *Skype* to *Game*, for example, decreases SVM classifier's accuracy from 74.7% to 18.6%, Bagged Trees classifier's accuracy from 90% to 15.7%, KNN classifier's accuracy from 83.9% to 22.5%, and Random Forest classifier's accuracy from 91.1% to 17.6%. *AdaptiveMutate*, using

Table 4.6: *AdaptiveMutate* results using mutation of lengths and IATs of packets

Source app \ Target app	Skype		Facebook		Game		WhatsApp		Viber		YouTube	
Skype			18.2%	6.5%	17.8%	6.3%	17.5%	6%	18.9%	7.1%	18.8%	7%
Facebook	18.3%	6.4%			17.8%	6.5%	17.2%	6.1%	18.7%	7.1%	17.8%	6.5%
Game	17.9%	6.1%	17.9%	6.4%					17.7%	6.4%	19.1%	7.5%
WhatsApp	15.7%	7.9%	15.7%	7.1%	15.7%	7.2%	15.5%	7.0%	16.2%	7.9%	15.8%	7.3%
Viber	15.4%	7.0%	15.7%	7.1%			15.6%	7.1%	16.5%	8.2%	16.2%	7.9%
YouTube	17.4%	6.1%	17.1%	6.2%	17.8%	6.5%			18.9%	7.1%	17.6%	6.1%
	15.7%	7.2%	15.6%	7.1%	15.5%	7.2%			16.3%	7.8%	15.7%	7.2%
	18.8%	7.0%	18.5%	7.3%	19.0%	7.6%	18.8%	7.0%			18.9%	7.3%
	16.1%	7.5%	16.6%	7.8%	16.6%	8.3%	16.4%	7.9%			16.5%	8.1%
	18.9%	7.1%	17.6%	6.3%	18.7%	7.1%	17.4%	6.2%	18.8%	7.2%		
	16.3%	7.7%	15.5%	7.1%	16.0%	7.6%	15.6%	7.3%	16.5%	8.0%		

■ SVM ■ BT ■ KNN ■ RF

Table 4.7: Overhead resulting from padding

Source app \ Target app	Skype	Facebook	Game	WhatsApp	Viber	YouTube
Skype	0%	18.4%	11.2%	9.5%	3.2%	39.7%
Facebook	4.8%	0%	4.1%	12.6%	4.3%	24.1%
Game	4.6%	3.5%	0%	1.0%	2.1%	34.2%
WhatsApp	0.2%	0.3%	5.7%	0%	0.2%	21.2%
Viber	53.4%	59.7%	55.6%	53.2%	0%	67.6%
YouTube	0.2%	0.3%	0.2%	0.006%	0.002%	0%

Table 4.8: Performance Time Delay Resulting from Mutating the IATs

Source app \ Target app	Skype	Facebook	Game	WhatsApp	Viber	YouTube
Skype	0%	80%	201%	257%	39%	37%
Facebook	4%	0%	84.2%	406%	75%	72%
Game	153%	112%	0%	552%	97%	92
WhatsApp	166%	461%	571%	0%	172%	176
Viber	85%	95%	99%	164%	0%	3%
YouTube	82	91%	99%	177%	3%	0%

Table 4.9: Confusion of *Skype* Traffic Mutated to *Game* by RF Classifier

Classifier	1	2	3	4	5	6
Mutation	10.8%	0.6%	80.3%	0.5%	6.4%	1.4%

the second version of the length mutation algorithm from *Skype* to *Game*, can decrease SVM classifier’s accuracy from 74.7% to 31%, Bagged Trees classifier’s accuracy from 90% to 37.6%, KNN classifier’s accuracy from 83.9% to 45.8%, and Random Forest classifier’s accuracy from 91.1% to 35.1% with only 4.29% average overhead. It could be used when less overhead is required than higher evasion defense. The cost is less, with results of acceptable attack evasion.



Simulation results, presented in Table 4.5, show that *AdaptiveMutate* with IAT mutation is not efficient in reducing the classifier’s accuracy. This could be related to the fact that the classifiers we build in our experiments have a weak relevance with features built around IAT. Other classifiers that rely on IAT as a strong feature can be more vulnerable to IAT mutation. However, when IAT mutation is combined with lengths mutation, the best results are achieved as shown in Table 4.6. For example, mutating *Skype* to *Game* decreases SVM classifier’s accuracy from 74.7% to 17.8%, Bagged Trees classifier’s accuracy from 90% to 6.3%, KNN classifier’s accuracy from 83.9% to 15.5%, and Random Forest classifier’s accuracy from 91.1% to 7.2% with only 11.2% average padding overhead and 201% of time delay. Full confidentiality can be reached with a cost of increasing traffic volume by a factor of 1.11, which is acceptable, especially when considering that full privacy could be essential for sensitive apps. In this case, the time needed for *Skype* to operate is triple the usual time, which could not be acceptable for some apps in which delay is not tolerated. Table 4.8 could be used to choose a suitable target app to mutate to with less performance degradation, especially when considering that using *AdaptiveMutate* ensures slightly different results when the target app is changed. This proves the robustness of our algorithms. For instance, mutating *Skype* to *Viber* is more suitable in this case. It reduces SVM classifier’s accuracy from 74.7% to 18.9%, Bagged Trees classifier’s accuracy from 90% to 7.1%, KNN classifier’s accuracy from 83.9% to 16.4%, and Random Forest classifier’s accuracy from 91.1% to 7.9% with only 3.2% average padding overhead and 39% of time delay. According to Table 4.9, 80.3% of the *Skype* traffic is confused as *Game* traffic by the RF classifier. This demonstrates the efficiency of our algorithm to convert the *Skype* traffic class into a *Game* class.

To maximize *AdaptiveMutate*’s efficiency, a user can choose the corresponding most dissimilar app suitable to mutate to in terms of lengths or IAT. By comparing the results in Table 4.3, Table 4.4, Table 4.5, and Table 4.6 with the cosine similarities of lengths and IAT in Table 4.1 and Table 4.2, respectively, we can notice that mapping a source app to the most dissimilar app achieves the highest efficiency of *AdaptiveMutate*. Furthermore, our algorithms are robust in a way that they can achieve relatively similar results for every mapping of the source app to the target app. Also, our countermeasures optimally balance efficiency and privacy. They are suitable for mobile devices with limited plans, since they do not introduce much overhead in terms of data and bandwidth. Unlike other approaches, the methods presented in this section are very appropriate for mobile app flows, where an excessive overhead is not acceptable as it affects the users’ experience. Moreover, they reduce the required processing and computing, and as such, they are feasible for constrained networking equipment. Another advantage of these methods is that they are operated in a dynamic online way with minimal latency.

### 4.1.6 Validation on MTU quantization

To assess the efficiency of our scheme, we compare it to another common approach for mitigating traffic analysis threats, “MTU Quantization” [84]. As mentioned previously, ciphered mobile traffic flows leak information through the range of packet lengths. In fact, packet length is among the most significant characteristics for mobile app traffic classification [86]. One obvious way to terminate such side information is by fixing all packets at maximum transmit unit (MTU). We evaluate our evasion method by examining the effectiveness of *AdaptiveMutate* and comparing it to MTU quantization. We run the same experiment on MTU quantization scheme to predict the classes of *Skype* mutated app traffic for the sake of example. Then, we predict using our 4 classifiers classes of the mutated traffic using MTU quantization.

The MTU quantization scheme decreases SVM classifier’s accuracy from 76.7% to 32.9%, Bagged Trees classifier’s accuracy from 90% to 21.3%, KNN’s classifier from 83.9% to 26.2%, and Random Forest classifier’s accuracy from 91.1% to 22.3%. *AdaptiveMutate* achieves better results in terms of reducing the classifiers’ accuracy. Also, MTU quantization scheme results in an overhead of 216.6%; which is inefficient. Padding packets to their maximum sizes degrades the performance of the network due to excessive padding that induces big amount of data sent. This approach is not practical, especially in case of mobile traffic because it involves such a large overhead.

### 4.1.7 Validation on *AppScanner*

For further evaluation, we conduct an experiment to test *AdaptiveMutate* on *AppScanner*, a well-known recent mobile apps classification approach presented in [17, 41]. First, we implement the *AppScanner* classification model: for each of the 6 applications of our dataset, we extract the flows that are defined as the sequence of packets within a burst of 1 s with the same destination IP and port addresses. We derive 40 features for each flow as described in [17], and we implement Random Forest with 150 estimators. Our traffic flow extraction results in 6867 flows that are divided into 75% for training and 25% for testing. We use the ‘one versus one’ strategy for the multi-classification problem and we evaluate it using 5 folds’ cross validation. *AppScanner* method achieves an overall accuracy of 94.8% in our test set of 6 apps.

To test the impact of thwarting, we choose *Skype* as the source app and *Game* as the target app. We modify *Skype*’s packet lengths using our three different algorithms as described before. Then, we predict using *AppScanner* classifier classes of the mutated traffic. The simulation results show that *AppScanner*’s accuracy drops from 94.8% to 18.19% when using *AdaptiveMutate* with the first version of lengths mutation, to 38.4% when using *AdaptiveMutate* with the second version of lengths mutation, and to 17.9% using *AdaptiveMutate* with both



Figure 4.4: *AdaptiveMutate* scheme

lengths and IAT mutations. This proves the competence of *AdaptiveMutate*'s algorithms to thwart *AppScanner* classification.

#### 4.1.8 Adaptive real-time approach

It offers traffic mutation service using three options: lengths mutation, interarrival time mutation, lengths & inter-arrival time mutation. In this section, we elaborate the adaptive capability of *AdaptiveMutate* that presents a viable approach to counter the unnecessary computation overhead. Based on the specific app analysis, *AdaptiveMutate* would autonomously determine the feature to mutate. Our adaptive selection structure is smart enough to adaptively select an optimal model to be used for mutation. For example, according to our experiments, to obscure the *YouTube* traffic, *Viber* app is the most suitable target app. Using *AdaptiveMutate*, with lengths and IAT mutations, decreases SVM classifier's accuracy from 74.7% to 18.8%, Bagged Trees classifier's accuracy from 90% to 7.2%, KNN classifier's accuracy from 83.9% to 16.5%, and Random Forest classifier's accuracy from 91.1% to 8% with only 0.002% average padding overhead and 3% of time delay.

Additionally, based on the classifier to defend against and the features used to build it, *AdaptiveMutate* would choose the ultimate feature to mutate. One can never guess which classifier an attacker uses to threaten a smartphone's privacy. Using *AdaptiveMutate* is like establishing an immune system to prevent from information leak due to the use of apps. For example, if the user wants to protect his privacy against classifiers built around length features, *AdaptiveMutate* would switch to mutating only the lengths. Otherwise, when the user needs to thwart against classifiers based on the IAT feature, the mutation of IAT is considered. Else, when the user needs to maximize his protection against classifiers that could rely on both IAT and lengths of packets, both features are considered in the mutation.

## 4.2 Anonymization Through Probabilistic Distribution

In this section, we develop a simpler and more scalable system to tackle the problem of network app traffic anonymization without the need of another app's model traffic. This could happen using probabilistic distribution of packet sizes. We propose first a scheme that regenerates statistical modeling of app packet lengths. Then, we present a privacy preserving technique that mutates packet lengths of the incoming traffic to the regenerated ones.

### 4.2.1 System Overview

Modeling app traffic plays an important role in network design and planning. In addition, these models can be used to create synthetic traffic. Authors in [164] show that any Internet traffic has self-similarity property. This implies that different parts of Internet traffic are either exactly the same or similar. Consequently, we use captured sessions of mobile app traffic communication to conduct our experiment. Initially, we identify a proper model that fits packet lengths of different apps traffic and distributions that do not fit correctly. It is worth noting that there is no single model that can describe efficiently traffic in all types of networks [136]. In fact, there are various number of probability distribution models that capture correct features and characteristics of traffic. We will prove the optimal model mathematically using *Kolmogorov-Smirnov standard goodness-of-fit test*, which represents the maximum absolute difference between distribution and experimental curves. Lesser *KS* indicates the better is the fit. Once the proper distribution fit is determined, packet lengths of an application traffic are regenerated accordingly.

Our goal is to protect against statistical traffic analysis by confusing an app traffic. In our first model, we mutate flows generated by the source app to defend so that their distribution becomes similar to regenerated packet lengths of another target app. For each incoming packet from the source app, we compare its size  $s_1$  to the size  $t_1$  of the regenerated length in the target app. If  $s_1$  is less than  $t_1$ , we proceed by padding  $s_1$  with zeroes such that the resultant packet length is the same as  $t_1$ . Otherwise, we fragment the packet into 2 fragments: first fragment has the same size as  $t_1$ , second fragment is considered as a new incoming packet where its size is compared to the next generated packet length of the target app.

We have considered app traffic mutation to thwart traffic classification by mutating a source app into a different target app. We could consider app traffic mutation of an app to itself without the need of a target app by only changing the distribution of its lengths. This would make our model more scalable. It rejects the need for a second app, especially if the type of the app that we are trying to evade is unknown. Therefore, in our second model, we only need to

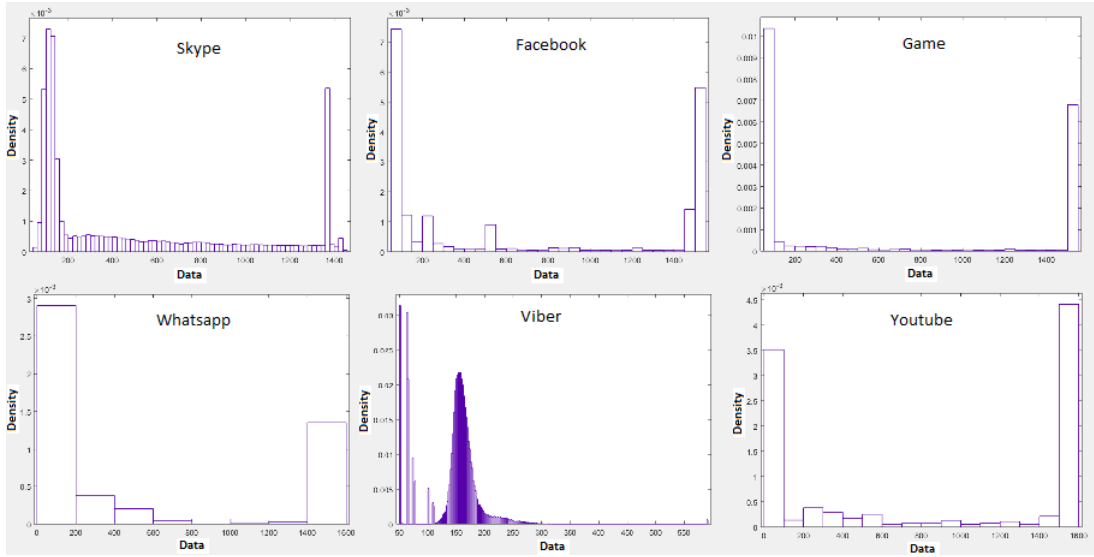


Figure 4.5: App traffic packet sizes distribution

study the distribution of the app’s packet lengths and change it to a different one. To realize this, we identify a distribution model that does not fit packet lengths of the app traffic in consideration, and we regenerate packet lengths from it. Lengths of regenerated packets are considered as ‘target’ app traffic lengths. We apply the mutation method explained previously to mutate a source packet length to the target packet length.

## 4.2.2 Experiment Evaluation

In our experiment, we use network traffic traces of the 6 apps of the dataset described in Subsection 3.5.1.

### Mobile Apps Probability Distribution Profiling

First, we examine the distribution of packet lengths across all instances in each app traffic set, and we estimate its probability distribution. Graphs in Figure 4.5 show the statistical analysis results of app traffic packet sizes distribution for the 6 apps of our dataset.

Next, we examine the best-fit function for each app packet sizes distribution. We model different distributions for the 6 different apps traffic. We consider Normal, Poisson, Rician, and Weibull as they are used in computer network traffic modeling. Normal is described by location and scale parameters. Poisson uses mean parameter. Rician is characterized by non-centrality and scale parameters. Weibull uses shape parameter and scale parameter.

We model these distributions for packet lengths of *Skype* and *Viber* in Fig-

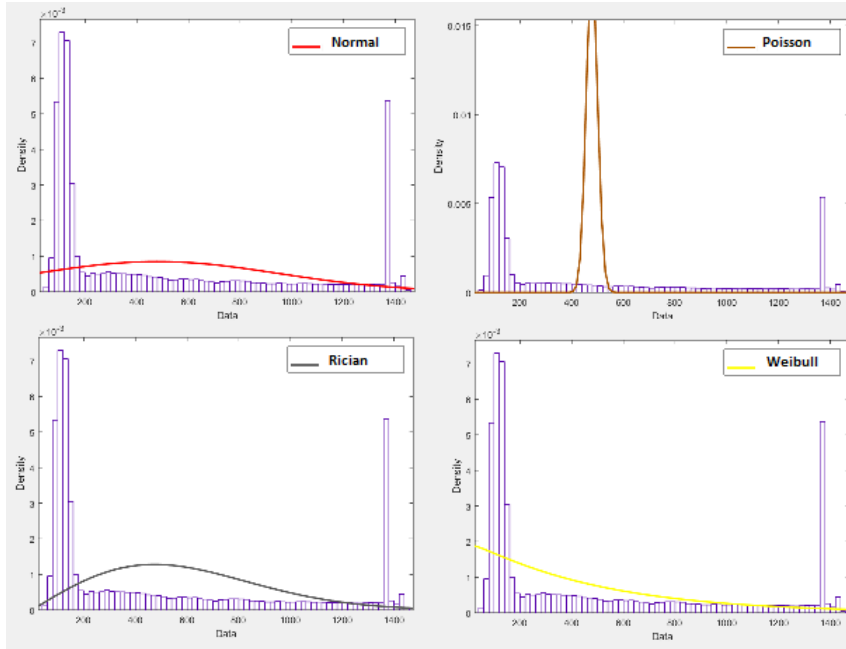


Figure 4.6: Distribution fits with respect to *Skype* packet lengths

ure 4.6 and Figure 4.7, respectively. We also specify the parameters used in Table 4.10. Param 1 and Param 2 represent the first and second parameter of distribution, respectively. Each class among Normal, Poisson, Rician, and Weibull hardly fit the data distribution perfectly. However, by visual examination of the fitted curves displayed in Figure 4.6, Normal is perceived to be the best-fit distribution over *Skype* traffic. *Viber* is observed to be closer to Poisson distribution in Figure 4.7. By simply plotting the 4 distributions fits for the remaining 4 apps, we notice that neither distribution matches.

Next, we evaluate, using goodness-of-fit function, which distribution-fit accurately describes traffic in our traces. At this purpose, we use *Kolmogorov-Smirnov* test to prove that a Normal process accurately describes traffic in *Skype*, and that a Poisson process accurately identifies *Viber* traffic. In fact, Normal showed the lowest *KS* parameter in case of *Skype*, and Poisson has the lowest *KS* parameter in case of *Viber* as presented in Table 4.10.

### Obfuscation Experiments and Results

To evaluate our evasion defense algorithms, we use the app classification experiment discussed in Chapter 3. Next, we describe two experiments corresponding to two different obfuscation techniques. The difference between the 2 models is the target app used in mutation.

In our first experiment, we calculate the mean and standard deviation of real *Skype* packet lengths to regenerate them from normal distribution. We also

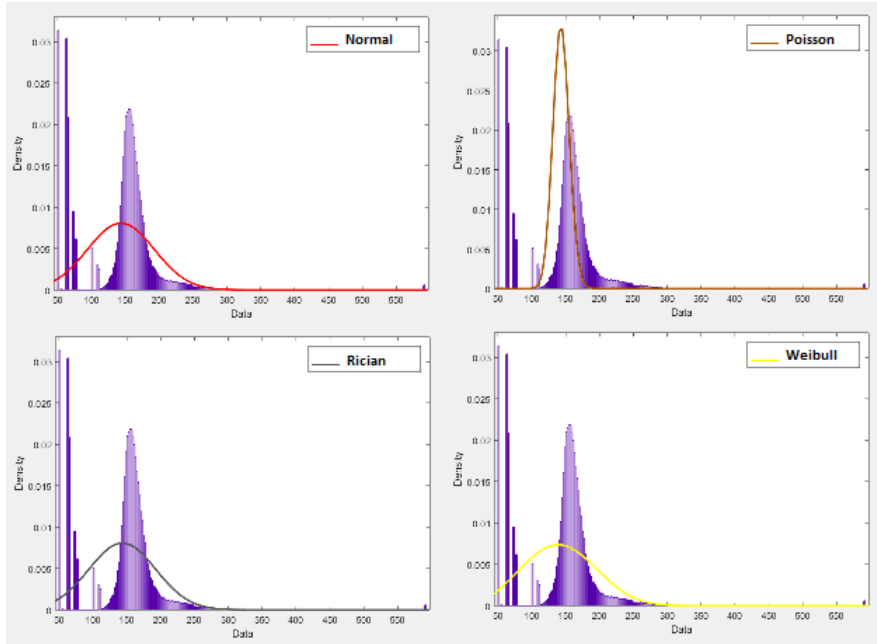


Figure 4.7: Distribution fits with respect to *Viber* packet lengths

Table 4.10: Packet lengths Distribution Fit Parameters and Goodness-of-fit

App	Distribution	Param 1	Param 2	KS
Skype	Normal	Location= 478.3	Scale= 469.9	0.217
	Poisson	Mean= 478.3	-	0.944
	Rician	Noncentrality= 27.8	Scale=473.8	0.375
	Weibull	Shape=483.4	Scale=1.0	0.329
Viber	Normal	Location= 143.3	Scale= 49.3	0.485
	Poisson	Mean= 143.3	-	0.116
	Rician	Noncentrality= 132.5	Scale= 52.0	0.526
	Weibull	Shape= 159.2	Scale= 2.9	0.583

calculate the mean of *Viber* packet lengths to regenerate them from Poisson distribution. The regeneration of *Skype* packet lengths from Normal distribution comes at the cost of 1.6% overhead only in comparison with the original *Skype* traffic. Also, regenerating *Viber* packet sizes from Poisson distributions costs 0.259% only when we compare it to the original *Viber* traffic.

Packets of our dataset are transferred one after the other and several flow sessions of these packets occur at the same time. We sort packets for each flow session between two communicating entities. Applying our first obfuscation technique, we mutate, for each flow session, the source app traffic to regenerate packet lengths from Normal distribution of *Skype*. We pad with zeroes each incoming packet from the source app to a generated packet length in case it is smaller.

Otherwise, we proceed by fragmenting it.

We test the 6 apps consecutively as source apps to mutate. Using our 4 model classifiers, we predict the classes of the modified traffic. Results of prediction for Quadratic SVM, Bagged Tree, Fine KNN, and Random Forest are shown in Table 4.11, Table 4.12, Table 4.13, Table 4.14 respectively for the six different apps. We present the resulting overhead from mutating each source app packet lengths to Normal distribution of *Skype* packet sizes in Table 4.15.

Simulation results show for example that mutating *Game* packet lengths to Normal distribution of *Skype* packet sizes can decrease SVM classifier’s accuracy from 76.7% to 0.4% as shown in Table 4.11, Bagged Trees classifier’s accuracy from 90% to 0.7% as shown in Table 4.12, KNN classifier’s accuracy from 83.9% to 2.76% as shown in Table 4.13, and Random Forest classifier’s accuracy from 91.1% to 0.9% as shown in Table 4.14 with only 12.51% average overhead as presented in Table 4.15. Hence, drastic drop of the accuracy is achieved with a minimal cost of traffic volume increase, by a factor 1.12. Also, 80.96% of the *Game* traffic was confused by Bagged Tree as *Skype* traffic as shown in Table 4.12. This proves the efficiency of our algorithm to morph a *Game* traffic class into *Skype* class.

Table 4.11: Confusion of Source app mutated to Normal distribution of *Skype* packet sizes by Quadratic SVM

Predicted as \ Source app	Skype	Facebook	Game	WhatsApp	Viber	YouTube
<b>Skype</b>	35.47%	43.59%	0.47%	3.09%	7.5%	9.88%
<b>Facebook</b>	49.42%	39.49%	0.15%	1.84%	2.86%	6.25%
<b>Game</b>	43.22	34.25%	0.4%	4.37%	6.67%	11.09%
<b>Whatsapp</b>	32.84	28%	0.42%	1.89%	11.37%	25.47%
<b>Viber</b>	42.07%	40.71%	0.74%	3.32%	3.57%	9.59%
<b>Youtube</b>	35.47%	43.59%	0.47%	3.09%	7.5%	9.88%

Table 4.12: Confusion of Source app mutated to Normal distribution of *Skype* packet sizes by Bagged Tree

Predicted as \ Source app	Skype	Facebook	Game	WhatsApp	Viber	YouTube
<b>Skype</b>	55.71%	30.77%	0%	6.35%	0.19%	6.92%
<b>Facebook</b>	16.84%	68.32%	0.35%	6.37%	1.33%	6.8%
<b>Game</b>	80.96%	10.56%	0.7%	3.18%	0.48%	4.12%
<b>Whatsapp</b>	47.13%	24.6%	0%	15.17%	0.69%	12.41%
<b>Viber</b>	32.84%	24%	0%	2.32%	10.53%	30.32%
<b>Youtube</b>	62.36%	15.25%	0.74%	4.31%	1.23%	16.11%

Similarly, we mutate in each session the source app traffic to regenerated packet lengths from Poisson distribution of *Viber*. Using our 4 model classifiers,



Table 4.13: Confusion of Source app mutated to Normal distribution of *Skype* packet sizes by Fine KNN

Source app \ Predicted as	Skype	Facebook	Game	WhatsApp	Viber	YouTube
<b>Skype</b>	47.89%	22.23%	1.8%	8.09%	3.75%	16.25%
<b>Facebook</b>	55.72%	23.45%	2.13%	3.05%	2.03%	13.61%
<b>Game</b>	40.46%	23.68%	2.76%	14.25%	2.53%	16.32%
<b>Whatsapp</b>	35.16%	25.26%	1.05%	13.68%	9.68%	15.16%
<b>Viber</b>	51.41%	20.3%	3.32%	5.9%	1.35%	17.71%
<b>Youtube</b>	47.89%	22.23%	1.8%	8.09%	3.75%	16.25%

Table 4.14: Confusion of Source app mutated to Normal distribution of *Skype* packet sizes by Random Forest

Source app \ Predicted as	Skype	Facebook	Game	WhatsApp	Viber	YouTube
<b>Skype</b>	57.23%	28.21%	0%	5.73%	0.15%	8.64%
<b>Facebook</b>	22.24%	58.91%	0.71%	5.98%	2.13%	9.8%
<b>Game</b>	76.12%	9.23%	0.9%	5.90%	0.58%	7.19%
<b>Whatsapp</b>	49.2%	28.69%	0%	17.78%	0.71%	3.41%
<b>Viber</b>	34.51%	36%	0%	1.12%	10.11%	18.24%
<b>Youtube</b>	59.9%	17.79%	0.81%	5.12%	1.76%	13.51%

Table 4.15: Overhead resulting from mutating to Normal distribution of *Skype* packet sizes

Target \ Source app	Skype	Facebook	Game	WhatsApp	Viber	YouTube
Normal distribution of Skype	1.6%	5.28%	12.51%	35.26%	63.34%	13.37%

we predict the classes of the modified traffic. Results of prediction for Quadratic SVM, Bagged Tree, Fine KNN, and Random Forest are shown in Table 4.16, Table 4.17, Table 4.18, and Table 4.19, respectively for the six different apps. We present the resulting overhead from mutating each source app packet lengths to Poisson distribution of *Viber* packet sizes in Table 4.20. Simulation results show for example that mutating *Game* packet lengths to Poisson distribution of *Skype* packet sizes can decrease SVM classifier’s accuracy from 76.7% to 2.66% as shown in Table 4.16, Bagged Trees classifier’s accuracy from 90% to 0.48% as shown in Table 4.17, KNN classifier’s accuracy from 83.9% to 48.89% as shown in Table 4.18, and Random Forest classifier’s accuracy from 91.1% to 1.43% as shown in Table 4.19 with only 14.33% average overhead as presented in Table 4.20. Hence, drastic drop of the accuracy is achieved with a minimal cost of traffic volume increase, by a factor 1.14.

In our second experiment, we evaluate another obfuscation technique where we change the distribution of packet lengths of an app into a dissimilar one. We

Table 4.16: Confusion of Source app mutated to Poisson distribution of *Viber* packet sizes by Quadratic SVM

Source app \ Predicted as	Skype	Facebook	Game	WhatsApp	Viber	YouTube
<b>Skype</b>	30.00%	42.31%	0%	1.35%	24.81%	1.54%
<b>Facebook</b>	36.54%	38.65%	0%	1.54%	21.73%	1.54%
<b>Game</b>	9.11%	64.24%	2.66%	0.15%	23.3%	0.53%
<b>Whatsapp</b>	22.99%	66.67%	0.46%	2.3%	7.13%	0.46%
<b>Viber</b>	6.53%	9.05%	0%	0.21%	84.21%	0%
<b>Youtube</b>	6.89%	64.21%	0.25%	0.62%	27.18%	0.86%

Table 4.17: Confusion of Source app mutated to Poisson distribution of *Viber* packet sizes by Bagged Tree

Source app \ Predicted as	Skype	Facebook	Game	WhatsApp	Viber	YouTube
<b>Skype</b>	52.69%	22.61%	0%	5.38%	18.85%	0.38%
<b>Facebook</b>	17.81%	68.2%	0%	7.15%	6.8%	0%
<b>Game</b>	7.56%	83.72%	0.48%	7.27%	0.97%	0%
<b>Whatsapp</b>	24.6%	53.79%	0%	18.39%	3.22%	0%
<b>Viber</b>	1.68%	6.58%	0%	3.16%	88.58%	0%
<b>Youtube</b>	10.95%	71.96%	0%	12.67%	4.18%	0.25%

Table 4.18: Confusion of Source app mutated to Poisson distribution of *Viber* packet sizes by Fine KNN

Source app \ Predicted as	Skype	Facebook	Game	WhatsApp	Viber	YouTube
<b>Skype</b>	45.00%	31.15%	1.92%	2.12%	19.42%	0.38%
<b>Facebook</b>	21.13%	49.49%	11.56%	5.04%	11.95%	0.82%
<b>Game</b>	20.45%	20.54%	48.89%	2.76%	4.75%	2.62%
<b>Whatsapp</b>	25.98%	40.46%	7.82%	20.69%	3.45%	1.61%
<b>Viber</b>	1.47%	7.79%	0.21%	2.32%	88.21%	0%
<b>Youtube</b>	13.65%	53.38%	10.95%	8.73%	3.94%	13.65%

Table 4.19: Confusion of Source app mutated to Poisson distribution of *Viber* packet sizes by Random Forest

Source app \ Predicted as	Skype	Facebook	Game	WhatsApp	Viber	YouTube
<b>Skype</b>	53.48%	24.31%	0%	4.21%	17.54%	0.38%
<b>Facebook</b>	19.84%	65.3%	0%	6.54%	6.41%	1.01%
<b>Game</b>	5.09%	84.92%	1.43%	8.05%	0.71%	0%
<b>Whatsapp</b>	21.6%	55.39%	0%	16.99%	4.51%	0%
<b>Viber</b>	1.58%	7.32%	0%	4.88%	85.87%	0%
<b>Youtube</b>	9.06%	72.57%	0%	13.28%	5.09%	0.14%

Table 4.20: Overhead resulting from mutating to Poisson distribution of *Viber* packet sizes

Source app \ Target	Skype	Facebook	Game	WhatsApp	Viber	YouTube
Normal distribution of Viber	19.87%	7.72%	14.33%	9.87%	0.0259%	3.81%

discussed earlier in Section 4.2.2 that apps other than *Skype* or *Viber* do not fit correctly into Normal or Poisson distributions. Therefore, we mutate packet lengths of *Facebook*, *Game*, *WhatsApp*, and *YouTube* to regenerated Normal or Poisson distributions of their packet sizes. We concentrate on Normal distribution and Poisson distribution for simple modeling. Prediction results of classes of the modified app traffic of *Facebook*, *Game*, *WhatsApp*, and *YouTube* using our 4 classifiers are presented in Table 4.21, Table 4.22, Table 4.23, and Table 4.24, respectively.

This obfuscation technique results in less but acceptable attack evasion outcomes compared to our first one presented earlier. For example, when predicting *Game* class after mutating its packet sizes to Normal distribution, SVM classifier’s accuracy decreases from 76.7% to 0.68%, Bagged Trees classifier’s accuracy from 90% to 0.19%, KNN classifier’s accuracy from 83.9% to 4.55%, and Random Forest classifier’s accuracy from 91.1% to 0.28% with only 7.73% average overhead. Also, when predicting *Game* class after mutating its packet sizes to Poisson distribution, SVM classifier’s accuracy decreases from 76.7% to 2.23%, Bagged Trees classifier’s accuracy from 90% to 1.7%, KNN classifier’s accuracy from 83.9% to 12.16%, and Random Forest classifier’s accuracy from 91.1% to 1.9% with only 29.1% average overhead.

Table 4.21: Prediction of mutated *Facebook* packet lengths to itself using Normal or Poisson distributions

Algorithm	SVM	BT	KNN	RF	Overhead
<b>Original</b> Facebook accuracy	76.7%	90%	88.9%	91.1%	-
Facebook <b>Mutated</b> to Facebook with lengths generated from Normal distribution	46.56%	68.48%	46.91%	65.2%	7.33%
Facebook <b>Mutated</b> to Facebook with lengths generated from Poisson distribution	52.93%	59.65%	50.27%	58.3%	29.90%

Table 4.22: Prediction of mutated *Game* packet lengths to itself using Normal or Poisson distributions

Algorithm	SVM	BT	KNN	RF	Overhead
<b>Original</b> Game accuracy	76.7%	90%	88.9%	91.1%	-
Game <b>Mutated</b> to Game with lengths generated from Normal distribution	0.68%	0.19%	4.55%	0.28%	7.73%
Game <b>Mutated</b> to Game with lengths generated from Poisson distribution	2.23%	1.7%	12.16%	1.9%	29.10%

Table 4.23: Prediction of mutated *WhatsApp* packet lengths to itself using Normal or Poisson distributions

Algorithm	SVM	BT	KNN	RF	Overhead
<b>Original</b> WhatsApp accuracy	76.7%	90%	88.9%	91.1%	-
WhatsApp <b>Mutated</b> to WhatsApp with lengths generated from Normal distribution	11.72%	15.63%	18.62%	16.11%	32.62%
WhatsApp <b>Mutated</b> to WhatsApp with lengths generated from Poisson distribution	3.68%	15.86%	30.34%	0.97%	21.75%

Table 4.24: Prediction of mutated *YouTube* packet lengths to itself using Normal or Poisson distributions

Algorithm	SVM	BT	KNN	RF	Overhead
<b>Original</b> YouTube accuracy	76.7%	90%	88.9%	91.1%	-
YouTube <b>Mutated</b> to YouTube with lengths generated from Normal distribution	37.88%	13.78%	9.59%	12.86%	6.77%
YouTube <b>Mutated</b> to YouTube with lengths generated from Poisson distribution	1.11%	4.18%	5.04%	4.89%	4.52%

## 4.3 Mutation for Similar Probability Distribution

### 4.3.1 Model Overview

In this model, we mutate each packet in the flows generated by the source app to protect such that their distribution becomes similar to packet lengths of another target app. This requires resampling packet lengths from probability distribution of a target app traffic that is different from the original packet lengths generated by the source app. In other words, we chop and pad network protocols packets so that the modified flow resembles a pre-defined target distribution.

Figure 4.8 illustrates our traffic mutation technique. Given a pair of applications A and B, where A is the source app to protect, and B is the target app that the source app should look like, we choose a bin size  $w$  for packet lengths. We derive the probability distribution  $P_A$  and  $P_B$  with respect to bins of packet lengths of A and B, respectively. For each incoming packet  $p$  of A of size  $L_p$

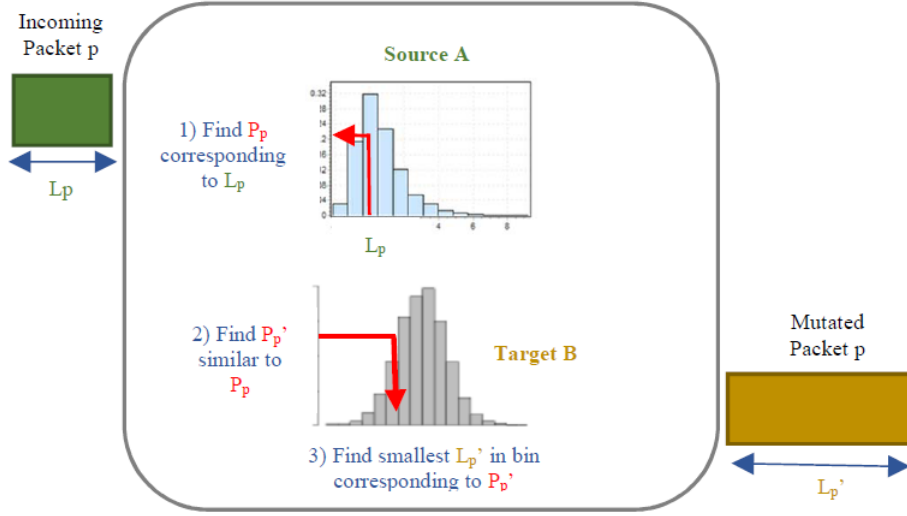


Figure 4.8: Mutation Algorithm

having a probability  $P_p$ , we find the most similar probability  $P_p'$  in target app B. Then, and to reduce overhead, we find the smallest size  $L_p'$  among the sizes in the bin of probability  $P_p'$  in B.

To find the most similar  $P_p'$  to  $P_p$ , we consider the vector  $P_B$  of probabilities for the different bins of packet lengths of target app. We find  $P_p'$  among the values in vector  $P_B$  satisfying (4.2)

$$\min |P_p - P_p'| \quad (4.2)$$

Then, we compare  $L_p'$  and  $L_p$ . If  $L_p' > L_p$ , we pad the packet from A with zeroes such that the resultant packet length is  $L_p'$  and we send the padded packet. Otherwise, the algorithm splits the data into multiple packets. We send  $L_i'$  bytes of the original packet and continue sampling from PB until all bytes of A are sent. Packet lengths that are output onto the network seem to be drawn from the target application distribution.

Using our algorithm, we actually mutate the smallest probabilities of packet lengths of A to the smallest probabilities of packet lengths of B and the largest ones from A to the largest in B. As long as the source process creates a necessarily big number of packets from a source distribution A, the output will converge in distribution to that of the target B realizing morphing of A into B. Hence, the classifier confuses an incoming packet from A of a length specific to B and classifies it in target app category. This method realizes morphing, but it can introduce large overhead in terms of padding depending on the choice of the target app for each source app.

One can argue that in spite of applying this technique, a user's behavioral patterns still leak. To obfuscate his behavior, a user may choose the target app

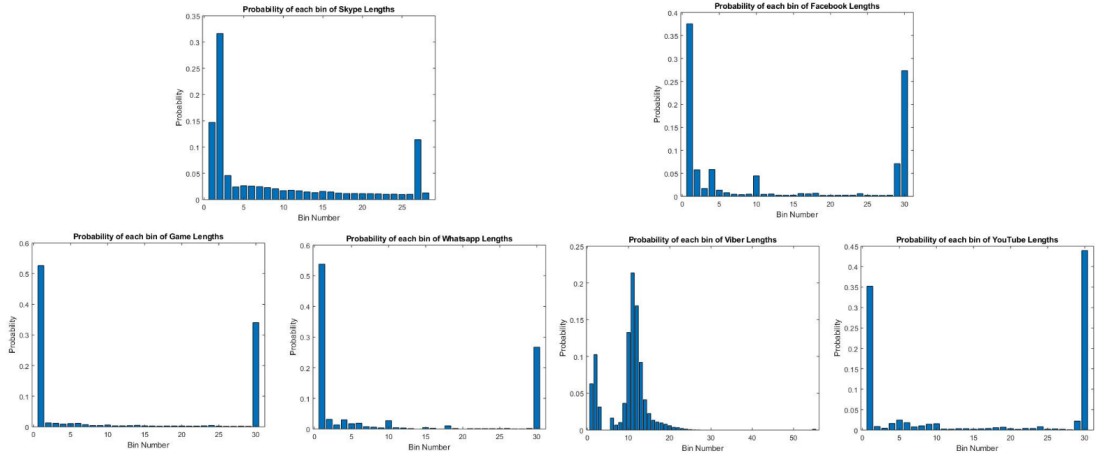


Figure 4.9: App Traffic Packet Sizes Distribution

that he does not usually use. Many apps can be appropriate in this case. The user can choose the target app that is the most dissimilar to his normal behavior for better obfuscation. Also, it would be beneficial to change the target app at each run to prevent the attacker from knowing the mapping of source app to target app.

Also, it is possible to generate matrices mapping each bin of a source packet to a size in another target app offline before the algorithm is used, and so the entire process becomes more practical inducing minimal latency, especially for dynamic applications.

### 4.3.2 Experimental Evaluation

In the following, we present an experiment as proof of concept of the above explained obfuscation algorithm. Our first step in this context is studying the packet size probability profiling for the used apps. Then, we present a comprehensive evaluation of our method using the different combinations of app traffic in our dataset. We rely in our evaluation on the mobile traffic classification attack experiment detailed in Chapter 3.

#### Mobile Apps Packet Size Probability

First, we divide the traces in each app traffic set into bins of size  $w$  of 50 bytes. We study the probability of each bin of packet lengths across all instances. Graphs in Figure 4.9 show the statistical analysis results of packet sizes distribution for the 6 apps of our dataset.

## Obfuscation Experiment and Results

We choose the source app to confuse and the target app to resemble. We save the probability of each packet length bin for the source and target apps. Then, for each incoming packet from source app, we find the probability of its corresponding bin size. We find the most similar length probability in target app and its corresponding bin. We pad or chop the incoming packet length to the smallest length inside the designated target bin.

Using our 4 model classifiers, we predict the classes of the mutated traffic. We present, in Table 4.25, Table 4.26, Table 4.27, Table 4.28, Table 4.29, Table 4.30 comprehensive simulation results of mutation using our algorithm of each application in our dataset to the 5 other different apps, respectively. Mutating *Skype* and *Facebook* was not as effective as mutating the other apps. For the sake of simplicity, we will explain in what follows the inefficiency of *Skype* mutation, and the efficiency of *Game* mutation. The other apps' results can be explained similarly.

In case of *Game* mutated to *Skype*, first bins of *Game* are mutated to bin numbered 2 of *Skype*, and last bins of *Game* are also mutated to bin numbered 2 of *Skype* because it is the most similar one. However, bin numbered 2 is representative for *Skype* because it has the highest probability of 0.32. Hence, the overall probability distribution of mutated *Game* was changed and the classifier identifying *Game* would confuse it as *Skype*. The same explanation clarifies the effectiveness of *Game* mutated to *Facebook*, and *Game* mutated to *Viber*.

When mutating *Game* to *WhatsApp*, the first bins of *Game* are mutated to the first bins of *WhatsApp* and the last bins of *Game* to the last bins of *WhatsApp*. So, the probability distribution of mutated *Game* would fit into probability distribution of *WhatsApp* confusing the classifier to classify the mutated *Game* traffic as *WhatsApp*.

When applying *Game* to *YouTube* using our algorithm, the first bins of *Game* are mutated to the last bins of *YouTube* and the last bins of *Game* to the first bins of *YouTube*. Therefore, the probability distribution of mutated source traffic is completely modified into that of the target app. Consequently, the classifier would classify the mutated *Game* traffic as *YouTube*.

The simulation results presented in Table 4.27, for example, show that mutating *Game* to *Viber* using our algorithm can decrease SVM classifier's accuracy from 76.7% to 0.48 %, Bagged Trees classifier's accuracy from 90% to 0.19%, KNN classifier's accuracy from 83.9% to 2.18%, and Random Forest classifier's accuracy from 91.1% to 0.22% with only 11.86% average overhead. According to Table 4.31, 69.23% of the *Game* traffic was confused as *Viber* traffic. This demonstrates the efficiency of our algorithm to Morph one class of *Game* traffic into the *Viber* class.

Table 4.25: Results of *Skype* Mutation

Algorithm	SVM	BT	KNN	RF	Overhead
Original % accuracy	76.7	90	88.9	91.1	-
% Accuracy of <b>Skype</b> mutated to <b>Facebook</b>	47.31	53.08	40.00	51.2	8.24%
% Accuracy of <b>Skype</b> mutated to <b>Game</b>	47.88	54.62	42.88	53.87	15.7%
% Accuracy of <b>Skype</b> mutated to <b>WhatsApp</b>	47.88	54.52	42.88	54.1	20.48%
% Accuracy of <b>Skype</b> mutated to <b>Viber</b>	49.62	58.46	45	57.98	2.64%
% Accuracy of <b>Skype</b> mutated to <b>YouTube</b>	47.31	53.08	40	52.55	49.85%

Table 4.26: Results of *Facebook* Mutation

Algorithm	SVM	BT	KNN	RF	Overhead
Original % accuracy	76.7	90	88.9	91.1	-
% Accuracy of <b>Facebook</b> mutated to <b>Skype</b>	44.3	58.95	47.81	57.4	85.21%
% Accuracy of <b>Facebook</b> mutated to <b>Game</b>	45.59	59.1	47.89	58.23	32.02%
% Accuracy of <b>Facebook</b> mutated to <b>WhatsApp</b>	44.57	58.09	47.31	57.5	27.7%
% Accuracy of <b>Facebook</b> mutated to <b>Viber</b>	45.43	58.01	49.96	56.2	10.51%
% Accuracy of <b>Facebook</b> mutated to <b>YouTube</b>	44.45	58.48	47.5	58.9	19.2%

Table 4.27: Results of *Game* Mutation

Algorithm	SVM	BT	KNN	RF	Overhead
Original % accuracy	76.7	90	88.9	91.1	-
% Accuracy of <b>Game</b> mutated to <b>Skype</b>	0.39	0.34	2.23	0.56	80%
% Accuracy of <b>Game</b> mutated to <b>Facebook</b>	0.19	0.24	2.13	0.44	21.37%
% Accuracy of <b>Game</b> mutated to <b>WhatsApp</b>	0.24	0.29	2.33	0.24	71%
% Accuracy of <b>Game</b> mutated to <b>Viber</b>	0.48	0.19	2.18	0.22	11.86%
% Accuracy of <b>Game</b> mutated to <b>YouTube</b>	0.53	0.15	1.74	0.18	36.33%

Table 4.28: Results of *WhatsApp* Mutation

Algorithm	SVM	BT	KNN	RF	Overhead
Original % accuracy	76.7	90	88.9	91.1	-
% Accuracy of <b>WhatsApp</b> mutated to <b>Skype</b>	5.52	14.02	11.3	13.4	70.76%
% Accuracy of <b>WhatsApp</b> mutated to <b>Facebook</b>	7.13	14.48	14.25	13.26	25.11%
% Accuracy of <b>WhatsApp</b> mutated to <b>Game</b>	5.06	14.02	10.34	14.01	21.73%
% Accuracy of <b>WhatsApp</b> mutated to <b>Viber</b>	3.22	14.94	11.95	13.29	7.76%
% Accuracy of <b>WhatsApp</b> mutated to <b>YouTube</b>	5.29	15.4	13.33	14.32	39.89%

## 4.4 Discussions

We explored so far the feasibility and effectiveness of several strategies to defend against traffic analysis attacks. With the reported experiments, we have investigated the security and performance of the different obfuscation systems. In all cases, corresponding to these mechanisms, a supposed adversary failed to correctly distinguish the victim apps on different identification rates.

Our first proposed system relied on modifying the packet lengths, and/or



Table 4.29: Results of *Viber* Mutation

Algorithm	SVM	BT	KNN	RF	Overhead
Original % accuracy	76.7	90	88.9	91.1	-
% Accuracy of <b>Viber</b> mutated to <b>Skype</b>	5.26	33.26	9.68	32.42	94.92%
% Accuracy of <b>Viber</b> mutated to <b>Facebook</b>	7.79	33.68	6.95	32.22	70.43%
% Accuracy of <b>Viber</b> mutated to <b>Game</b>	9.68	34.73	11.79	34.3	49.88%
% Accuracy of <b>Viber</b> mutated to <b>WhatsApp</b>	8	34.74	8.84	33.29	76.85%
% Accuracy of <b>Viber</b> mutated to <b>YouTube</b>	9.89	32.84	10.53	33.1	89.81%

Table 4.30: Results of *YouTube* Mutation

Algorithm	SVM	BT	KNN	RF	Overhead
Original % accuracy	76.7	90	88.9	91.1	-
% Accuracy of <b>YouTube</b> mutated to <b>Skype</b>	9.86	8.61	17.22	10.1	27.48%
% Accuracy of <b>YouTube</b> mutated to <b>Facebook</b>	9.95	8.61	17.22	9.93	85.34%
% Accuracy of <b>YouTube</b> mutated to <b>Game</b>	9.76	8.61	17.22	10.4	5.69%
% Accuracy of <b>YouTube</b> mutated to <b>WhatsApp</b>	9.84	11.81	19.19	9.72	8.3%
% Accuracy of <b>YouTube</b> mutated to <b>Viber</b>	9.96	11.32	20.54	9.82	1.68%

Table 4.31: Confusion of *Game* mutated into *Viber* using Our Algorithm

	Skype	Facebook	Game	WhatsApp	Viber	YouTube
Game (%)	13.87	12.94	0.22	1.73	69.23	2.01

the IAT information of the source app so that the lengths or IAT of the output packets seem as if they are coming from the target app probability mass function. We mapped the smallest values of packet lengths or IAT of the original app to the corresponding smallest ones in the target app with the aim of minimizing the overhead. Our study proved that the impact of a such thwarting system has significant effects on the general detection accuracy of the models in question, where the percentage dropped from 91.1% to 7% with 11.2% overhead.

Our second countermeasure relied on statistical modeling and mutating incoming packet lengths to regenerating ones from a predefined model. The main advantage of this system is that it doesn't require the need of another real app traffic especially if the type of the app that we are trying to evade is unknown. To realize this, we elaborated a method to identify a distribution model that fits packet lengths of the app traffic in consideration. Then, we mutated flows generated by the source app to defend so that their distribution becomes similar to regenerated packet lengths of another target app. Using this method, the targeted misclassification was reduced from 91.1% to 0.9% with 12.1% overhead.

Third countermeasure consisted of mutating source app packet lengths to target app packet lengths with similar bin probability. The results showed the effectiveness of the proposed method where classification accuracy dropped from 91.1% to 0.22% using the first algorithm with 11.86% of overhead.

The 3 thwarting systems considered so far attempted to obfuscate leaked

features of the traffic via padding, fragmenting, traffic shaping, etc. The test results showed the effectiveness and robustness of the different proposed models to protect vulnerable traffic. Despite the very promising results achieved by our different obfuscation systems, it is clear that such these approaches exhibits some intrinsic limitations. For instance, countermeasures that present a big overhead in terms of bandwidth, time, and computational cost are not practical since such overhead directly result in added monetary charge, poorer user experience, and bigger battery consumption.

After several successful defense models, the question becomes how to optimize the solution to realize balance between required quality of service and efficiency. This will be considered in the next chapter. Also, the challenge becomes harder to find an analytical explanation of the results of the different obfuscation systems and to compare between them.

## 4.5 Conclusion

In this chapter, we developed different anonymity techniques to thwart classification and avoid detection in the sake of user privacy. The numerical investigations presented give merit to our simple obfuscation approaches. In fact, the results show the effectiveness of the proposed method to cause an attacker to fail his classification. While there is an amount of overhead with respect to padding, as expected since no optimization is tried, however our techniques eliminate information that could be misused by the classification adversary. Also, most of the methods presented so far conserve implementation simplicity, and acceptable overhead.

After protecting insecure traffic, the challenge becomes harder when obfuscation systems need to abide to better performance requirements, have small computational needs, and confine minimal incurred overhead. Obviously, that would reinforce the necessity for optimal adequate countermeasures.

# Chapter 5

## Obfuscation Optimization

While we investigated earlier a handful of obfuscation algorithms resulting in significant outcomes, still there has not been a formal definition of our security problem leading to an ideal robust defense model against Traffic Analysis attacks. No mathematical solution has been assessed yet to optimally balance both efficiency and performance for a solid obfuscation algorithm. In this chapter, we state an optimization problem to find the full obfuscation algorithm that minimizes the average overhead within the set of ideal algorithms. We aim at finding the sufficient padding or fragmentation that guarantees good secrecy while optimizing the overhead.

Information leakage of the flow content is provided typically from traffic features like packet lengths, inter-packet times, traffic direction, etc. To achieve maximum protection, we formulate mathematically an optimization problem that yields a constructive solution for an algorithm achieving maximum obfuscation, while minimizing the overhead and subject to traffic constraints. We modify each packet in the flows, created by the source app to defend, such that their distribution becomes similar to packet sizes of a different target app. To this end, we consider one source app to protect and a number of  $N$  target apps to mutate to.

### 5.1 Problem Statement

At a high-level, our optimization problem is defined as follows. Given a source application, our goal is to mutate its traffic such that it resembles and is confused as some target application (i.e., a different application) with respect to a given set of features. The user chooses first the source application that he would like to defend, as well as a set of target applications that he would like to make the source application look like. Of course, we need to ensure that the implemented mutation is the most effective, for some measure of efficiency, such as the number of overhead bytes added. The user applies the optimization technique that dictates how each packet size from the source application should be mutated such

that the resulting distribution resembles the target with a minimum of overhead. Naturally, after the mapping of the packet sizes is performed offline, the mutation algorithm captures the data to be conducted across the network before headers are calculated or encryption is applied. The mutated traffic is then sent to the network stack, encrypted, and transmitted across the network as usual. Our objective is to maximize network traffic anonymization (i.e. minimize the accuracy of a classifier detecting certain applications) and to minimize the overhead resulting from padding. This would be realized by means of:

1. selecting the optimum target app,
2. and determining the optimum packet size to mutate to.

As such, the mutation algorithm can be formulated as an optimization problem to mutate a source application into a target application.

An application  $i$  can be defined by  $[m_i, L_i, P_i]$ :

- The number of possible packet size ranges of width  $w$  is denoted by  $m_i$ .
- The vector  $L_i = [L_i^1, L_i^2, \dots, L_i^{m_i}]$  of smallest lengths in each bin of size  $w$ . For example,  $L_i^{m_i}$  is the smallest length in the last bin of application  $i$  and having a total number  $m_i$  of bins. The average of  $L_i$  is denoted by  $\text{Avg}(L_i)$ .
- The probability distribution characteristic denoted by  $P_i = [P_i^1, P_i^2, \dots, P_i^{m_i}]$  represents the vector of length probabilities for bins of size  $w$ . For example,  $P_i^{m_i}$  is the probability of length  $L_i^{m_i}$  for application  $i$ .

An incoming packet from the source application can be defined by its size  $l$  and the probability  $p$  of the bin to which  $l$  belongs. We define a binary variable for each application that indicates whether the app is selected as target app or not;  $x_i$  denotes the variable for the  $i^{\text{th}}$  app. Given a set of  $N$  applications, the vector  $X$  (of length  $N$ ) represents  $[x_1, x_2, \dots, x_N]$ . Also, we delineate a binary variable for each bin of an application  $i$  that indicates whether the packet bin is selected to be mutated to or not. It is worth noting that the smallest size in a bin is used for mutation.  $y_i^j$  denotes the variable for the  $j^{\text{th}}$  bin of application  $i$ . Given a set of  $m_i$  ranges, the vector  $y_i$  (of length  $m_i$ ) designates  $[y_i^1, y_i^2, \dots, y_i^{m_i}]$ . In order to minimize the accuracy of the classifier, we need to find the probability of a packet length in a bin of a target app such that it is similar to  $p$ . At the same time, to minimize the overhead, the selected packet length must be similar to  $l$ . In case the difference between  $p$  and the designated probability is too small (less than 0.1 in our case), the difference between the selected length and  $l$  needs to be checked to be less than the average difference. This would guarantee that the overhead does not reach a large value. The obfuscation optimization problem

for mutating an incoming packet from a source app is as follows: Find the vector  $X$  (of length  $N$ ), and the vector  $y_i$  (of length  $m_i$ ), and minimize:

$$\text{Min} \sum_{i=1}^N x_i \left( \sqrt{\sum_{j=1}^{m_i} y_i^j (|P_i^j - p|^2 \times |L_i^j - l|^2)} \right)$$

Subject to:

1.

$$P_i^j \in [0, 1]; i = 1, \dots, N$$

2.

$$|L_i^j - l| < |Avg(L_i) - l| \text{ for } |P_i^j - p| < 0.1$$

3.

$$\sum_{i=1}^N x_i = 1; x_i = 0 \text{ or } 1; i = 1, \dots, N$$

4.

$$\forall i \in [1, N], \sum_{j=1}^{m_i} y_i^j = 1; y_i^j = 0 \text{ or } 1; j = 1, \dots, m_i$$

In the following, we will present 2 solutions for the optimization problem:

- An analytical solution.
- A practical solution *Opriv* provided by an optimization modeling software.

## 5.2 Analytical Solution

The ultimate mutation solution for our optimization problem is to identify for each length probability from source app the most similar ones from target apps. Then, we mutate the length from source app to the smallest corresponding length of these similar probabilities from target apps.

Figure 5.1 depicts our solution. We consider a source application  $S$  to protect and a set of target applications to mutate to, numbered from 1 to  $N$ . For each incoming packet of  $S$  of size  $l$  having a probability  $p$ , we find the bins of packets within the  $N$  applications with similar size probabilities  $[p_1, p_2, \dots, p_N]$  using Equation (4.2). We mark these bins in red in Figure 5.1. Then, we find their corresponding smallest lengths  $[l_1, l_2, \dots, l_N]$  in the bin they belong to. We choose to mutate  $l$  to the smallest length among  $[l_1, l_2, \dots, l_N]$ .

The classifier would notice specific lengths from the  $N$  target apps and then, it would classify the incoming packets in their corresponding target app rather

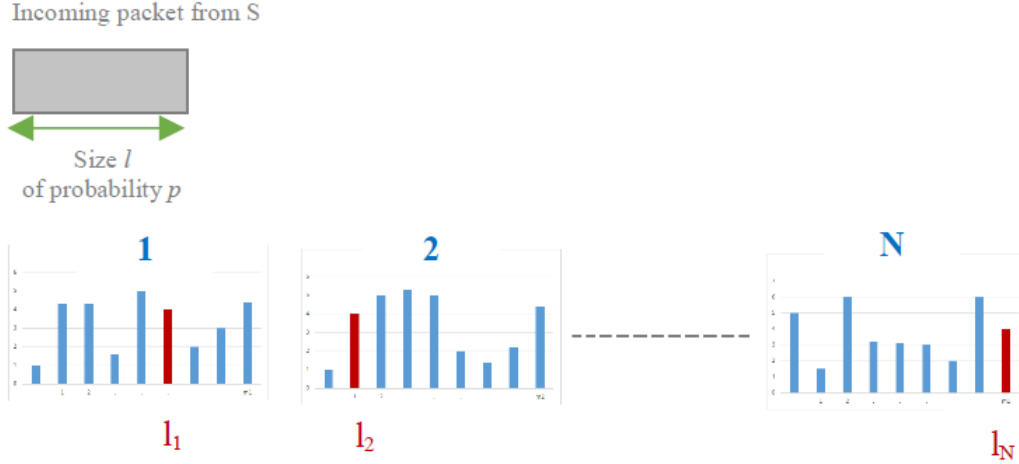


Figure 5.1: Camouflage Scheme

than random classification. The use of optimization ensures minimum padding overhead when mutating the source packets.

Additionally, the generation of the matrices mapping each bin of a source packet to a size in another target app can be done offline, and therefore the whole method becomes more practical resulting in minor latency, mainly for the dynamic type of applications.

By applying this algorithm, the user’s behavior is obfuscated by design because each single source app is mutated to several target apps at once. This would cause a confusion for the classifier, and the attacker would not be able to discriminate efficiently a single app. Hence, applying our analytical optimal solution would increase the unpredictability of the source as well as the target app, and ultimately the user’s behavior.

### 5.3 *Opriv*

In this part, we present *Opriv*, a practical solution to the optimization problem. The model has constraints including a quadratic term and therefore it is a Mixed Integer Quadratically Constrained Program (MIQCP) problem. As a practical result, we deploy and solve the problem deciding on Mixed-Integer Quadratic Problem MIQP linearization using the *IBM ILOG CPLEX Optimization Studio Version: 12.9.0.0* software. CPLEX was chosen for its robustness in solving large-scale, real-world problems. The coding in CPLEX is depicted in Figure 5.2 and Figure 5.3 . The result for one iteration is vector  $X$  to identify the target app for mutation, and the corresponding  $Y$  to specify the packet bin to mutate to in target app.

The proposed method, implemented using IBM’s CPLEX optimization pack-

```

//Parameters
int N=...;//number of apps
int M=...;//number of bins in each app
float p=...;//probability of size of incoming packet
from source app
int l=...;//size of incoming packet from source app
float AvgSize=...;//Average size of incoming packet
sizes
range app=1..N;
range bin=1..M;
float prob[app][bin]=...;
float L[app][bin]=...;

//Variables
dvar float+ x[app];
dvar float+ y[app][bin];

//Model
minimize sum(i in app, j in bin)
x[i]*(y[i][j]*sqrt((prob[i][j]-p)^2*(L[i][j]-1)^2));

subject to{
forall (i in app)
  cons01:
  x[i]==0 || x[i]==1;

forall (i in app)
  cons02:
  sum(i in app) x[i]==1;

forall (i in app, j in bin)
  cons03:
  y[i][j]==0 || y[i][j]==1;

forall (i in app)
  cons04:
  sum(j in bin) y[i][j]==1;

forall (i in app, j in bin)
  cons05:
  if (abs(prob[i][j]-p)<=0.1) (L[i][j]-1)<=(AvgSize-1);}

```

Figure 5.2: OPL 12.9.0.0 Model

```

N=5;
M=30;
p=0.1;
l=100;
AvgSize=100.0;

SheetConnection my_sheet("data.xlsx");

prob from SheetRead(my_sheet,"Probability");
L from SheetRead(my_sheet,"size");

x to SheetWrite (my_sheet,"X");
y to SheetWrite (my_sheet,"Y");

```

Figure 5.3: OPL 12.9.0.0 Data

age (ILOG), required 1–4 minutes to solve with a dual-core Intel i7 processor. To solve several instances of the same problem, we define our model in Java and call *Cplex Java API* on Eclipse to solve it. We record in Table 5.1 the average execution time needed to execute each instance of optimization for the different applications.

Table 5.1: Average Execution Time for each instance of optimization for the different applications

Application	Average Execution Time in seconds
Skype	0.16
Facebook	0.36
Game	0.2
WhatsApp	0.14
Viber	0.36
YouTube	0.19

## 5.4 Experimental Evaluation

In what follows, we empirically evaluate the efficiency of our solutions. We find the probability distribution with respect to packet lengths of the source app and those of the target apps traffic as depicted in Chapter 4 in Figure 4.9. To evaluate both solutions, we use our 4 model classifiers described in Chapter 3, and we predict the classes of the mutated traffic.



### 5.4.1 Analytical Solution Implementation

For each incoming packet from a source app, we implement the analytical solution of the optimization problem to select the optimal target app and packet size to mutate to. Table 5.2 to Table 5.7 present, for each bin size of a source app from our dataset described in Chapter 3, the corresponding optimum target app and optimum packet size to mutate to, using our obfuscation model.

Table 5.8 presents mutation results using our analytical solution for each app in our dataset. For example, mutating Game using analytical solution algorithm can decrease SVM classifier’s accuracy from 76.7% to 0.24%, Bagged Trees classifier’s accuracy from 90% to 0.001%, KNN classifier’s accuracy from 83.9% to 1.89%, and Random Forest classifier’s accuracy from 91.1% to 1.76% with only 0.73% average overhead.

We also assess our results in terms of the f1-score. The highest possible value of f1-score is 1, indicating perfect precision and recall, and the lowest possible value is 0, if either the precision or the recall is zero. Table 5.9 presents f1-score results using our analytical solution of each app in our dataset.

### 5.4.2 *Opriv* Evaluation

For each incoming packet from the source app, we implement *Opriv* to select the optimal target app and packet size to mutate to. First, we select the unique values of lengths and their corresponding probabilities of a certain app in our dataset, as input to our code in IBM ILOG CPLEX. The output results are the optimal packet size to mutate to. We generate tables mapping each length source packet to its corresponding optimal target size offline, and so the entire process becomes more practical inducing minimal latency, especially for dynamic applications. *Skype*, *Facebook* and *Game* have around 1400 unique values of sizes. While, *WhatsApp* and *Viber* have around 200 unique values. As for *YouTube*, the number is 1000. For the sake of simplicity, we show only a sample of seven mapping values in Table 5.10 to Table 5.15. Then for each incoming packet, *Opriv* matches its packet length and probability using our offline tables, and infers the corresponding optimum packet size that we need to mutate to. We pad or chop the incoming packet length to the corresponding optimum length.

We present, in Table 5.16, comprehensive simulation results of mutation using *Opriv* of each application in our dataset. As expected using optimization, *Opriv* realizes a good balance between efficiency and overhead especially for *Game*, *WhatsApp*, *Viber*, and *YouTube*. We will explain in Chapter 6 the reason why *Skype* and *Facebook* obfuscation were less efficient.

Table 5.2: Optimization Results of *Skype* Source App using Analytical Solution

Source Bin Number	Optimum Target App	Optimum Length to mutate to
1	Game	104
2	YouTube	54
3	Game	104
4	Game	104
5	Game	104
6	Game	104
7	Game	104
8	Game	104
9	Game	104
10	Viber	100
11	Viber	100
12	Viber	100
13	Game	104
14	Game	104
15	Viber	100
16	Game	104
17	Game	104
18	Game	154
19	Game	154
20	Viber	210
21	Viber	210
22	Viber	210
23	Viber	120
24	Viber	120
25	Game	204
26	Viber	220
27	Viber	62
28	Game	104

### 5.4.3 Discussion

We presented in this chapter, realistic and precise obfuscation approaches to implement in real world practice. Our analytical approach is systematic and achieves maximum obfuscation with minimum padding overhead. *OPriv* is differential from other past studies by the way it is coded in an optimization software so that it could be immediate and practical. In addition, the mapping of values between original and mutated is realized length to length and not bin to bin. This would imply better precision of *OPriv* than that of previous works. Also, *OPriv* is suitable for all types of network devices and does not need much processing requirements.

The results presented show the effectiveness and robustness of the proposed obfuscation models in balancing performance and security. They ensure privacy protection, and reduce data and bandwidth overhead at the same time. Our models are well intended to both secure traffic and preserve the computational requirements. They are appropriate for network devices with limited plans es-

Table 5.3: Optimization Results of *Facebook* Source App using Analytical Solution

Source Bin Number	Optimum Target App	Optimum Length to mutate to
1	YouTube	54
2	Viber	50
3	Viber	100
4	Viber	50
5	Game	104
6	Viber	230
7	YouTube	154
8	Viber	250
9	YouTube	154
10	Game	104
11	YouTube	154
12	Viber	240
13	Viber	260
14	Viber	270
15	Viber	260
16	Viber	240
17	Viber	240
18	Viber	110
19	Viber	270
20	Viber	270
21	Viber	270
22	Viber	260
23	Viber	270
24	Viber	240
25	Viber	270
26	Viber	270
27	Viber	270
28	Viber	260
29	Viber	50
30	YouTube	54

pecially that they reduce overhead in terms of data and bandwidth. Another advantage of these methods is that matrices mapping the mutation needed to modify incoming length into optimal length can be realized offline. Accordingly, our schemes can run in a dynamic online manner and the entire process induces minimal latency. The mapping of packet lengths from a source app to the corresponding optimal lengths in target app using *OPriv* presented in tables Table 5.10 to Table 5.15, and the obfuscation results in Table 5.16 are very compatible with the same mapping in Table 5.2 to Table 5.7 and obfuscation results in Table 5.8 of using the analytical model. This would prove the solidarity of our analytical model without the need to use of an optimization software. However, *OPriv* could be more hands-on in terms of easiness to use an available commercial software.

Table 5.4: Optimization results of *Game Source App* using Analytical Solution

Source Bin Number	Optimum Target App	Optimum Length to mutate to
1	Facebook	54
2	WhatsApp	154
3	Viber	210
4	YouTube	104
5	Viber	120
6	Viber	210
7	Viber	110
8	YouTube	154
9	YouTube	154
10	Viber	240
11	Viber	260
12	Viber	260
13	Viber	250
14	YouTube	154
15	Viber	260
16	Viber	260
17	Viber	270
18	Viber	260
19	Viber	260
20	Viber	260
21	Viber	270
22	Viber	270
23	Viber	260
24	YouTube	154
25	Viber	270
26	Viber	270
27	Viber	280
28	Viber	270
29	Viber	280
30	Facebook	54

## 5.5 Conclusion

Common techniques for defending against traffic analysis attacks, do not sufficiently conceal user activities with reasonable data overhead. In this chapter, we developed optimized models to prevent the adversary’s ability to precisely distinguish genuine user activities. The optimum masking solutions shall shape the packet flow so that full privacy at minimal overhead cost is guaranteed. For this reason, we formulated network obfuscation of the traffic analysis attack as a mathematical optimization problem. Then, we proposed an analytical solution to solve the problem of selecting the optimal target app as well as the optimal target packet length to mutate to. We also provided a practical solution *Opriv* designed and deployed to yield for a constructive solution achieving maximum obfuscation and minimum padding overhead. We demonstrated the effectiveness and robustness of the proposed models both in theory and empirically by per-

Table 5.5: Optimization Results of *WhatsApp* Source App using Analytical Solution

Source Bin Number	Optimum Target App	Optimum Length to mutate to
1	Facebook	54
2	Viber	73
3	Game	104
4	Viber	73
5	Viber	100
6	Viber	100
7	Viber	230
8	Viber	240
9	Viber	250
10	Viber	73
11	Viber	250
12	Viber	260
13	Viber	280
14	YouTube	154
15	Viber	270
16	Viber	300
17	Viber	120
18	Viber	270
19	Viber	592
20	Viber	592
21	Viber	592
22	Viber	592
23	Viber	592
24	Viber	270
25	Viber	290
26	Viber	290
27	Viber	270
28	YouTube	54

forming our defense models on real traffic traces. These 2 optimal approaches outperform the other state-of-the-art defense solutions against traffic analysis.

Table 5.6: Optimization Results of *Viber* Source App using Analytical Solution

Source Bin Number	Optimum Target App	Optimum Length to mutate to
1	Game	104
2	Game	104
3	Game	104
4	WhatsApp	884
5	Game	104
6	WhatsApp	408
7	Game	254
8	Game	104
9	Skype	53
10	Skype	53
11	Skype	53
12	Game	104
13	Game	104
14	Game	104
15	Game	104
16	Game	254
17	Game	204
18	Facebook	304
19	WhatsApp	408
20	Facebook	404
21	Game	554
22	WhatsApp	809
23	WhatsApp	1060
24	Skype	1253
25	WhatsApp	884
26	WhatsApp	884
27	WhatsApp	884
28	WhatsApp	884
29	WhatsApp	1060

Table 5.7: Optimization Results of *YouTube* Source App using Analytical Solution

Source Bin Number	Optimum Target App	Optimum Length to mutate to
1	Facebook	54
2	Game	204
3	Viber	250
4	Viber	100
5	Game	104
6	Viber	100
7	Viber	230
8	Viber	120
9	Game	104
10	Viber	100
11	Viber	270
12	Viber	270
13	Viber	260
14	Viber	260
15	Viber	270
16	Viber	260
17	Viber	250
18	Viber	240
19	Viber	110
20	Viber	250
21	Viber	270
22	Viber	250
23	Viber	250
24	Game	204
25	Viber	270
26	Viber	260
27	Viber	270
28	Viber	290
29	Game	104
30	Facebook	54

Table 5.8: Results of Apps Mutation using Analytical Solution

Algorithm	SVM	BT	KNN	RF	Overhead
Original % Accuracy	76.7	90	83.9	91.1	-
% Accuracy of Skype Mutated	41.93	53.48	42.68	41.24	4.28%
% Accuracy of Facebook Mutated	46.31	70.21	48.57	45.92	7.91%
% Accuracy of Game Mutated	0.24	0.001	1.89	1.76	0.73%
% Accuracy of WhatsApp Mutated	4.72	13.53	12.62	11.91	0.106%
% Accuracy of Viber Mutated	13.12	32.05	12.61	13.19	0.57%
% Accuracy of YouTube Mutated	15.26	6.09	16.77	14.99	0.93%

Table 5.9: f1-score of Apps Mutation using Analytical Solution

<b>Algorithm</b>	<b>SVM</b>	<b>BT</b>	<b>KNN</b>	<b>RF</b>
Skype Mutated	0.42	0.56	0.41	0.45
Facebook Mutated	0.41	0.68	0.52	0.44
Game Mutated	0.001	0	0.001	0.012
WhatsApp Mutated	0.03	0.01	0.02	0.02
Viber Mutated	0.02	0.03	0.01	0.01
YouTube Mutated	0.01	0.05	0.01	0.04

Table 5.10: Sample of length mapping of *Skype* App using *OPriv*

Original Length	Original Length Probability	Optimum Length to mutate to
85	0.0086	54
103	0.0072	154
120	0.008	154
134	0.00623	154
145	0.004064	104
760	0.00031	804
1440	0.000431	1354

Table 5.11: Sample of length mapping of *Facebook* App using *OPriv*

Original Length	Original Length Probability	Optimum Length to mutate to
66	0.33539	54
74	0.00487	104
122	0.00015	154
187	0.00030	205
384	0.000335	320
898	0.00014	854
1514	0.27281	1504

Table 5.12: Sample of length mapping of *Game* App using *OPriv*

Original Length	Original Length Probability	Optimum Length to mutate to
74	0.01562	104
78	0.01908	154
102	0.00757	355
544	0.00286	704
363	0.00113	554
732	0.00105	804
1210	0.00186	1307



Table 5.13: Sample of length mapping of *WhatsApp* App using *OPriv*

Original Length	Original Length Probability	Optimum Length to mutate to
66	0.43874	104
74	0.01267	254
192	0.00694	154
292	0.00815	304
308	0.00724	355
914	0.00663	1057
1514	0.26705	1354

Table 5.14: Sample of length mapping of *Viber* App using *OPriv*

Original Length	Original Length Probability	Optimum Length to mutate to
50	0.0628	154
63	0.04657	104
65	0.04173	304
150	0.02118	205
155	0.02219	254
160	0.02028	154
592	0.00119	654

Table 5.15: Sample of length mapping of *YouTube* App using *OPriv*

Original Length	Original Length Probability	Optimum Length to mutate to
66	0.33381	62
78	0.01202	104
500	0.00562	592
514	0.00668	240
548	0.00616	300
1484	0.01667	804
1514	0.43893	1354

Table 5.16: Results of apps mutation using *OPriv*

Algorithm	SVM	BT	KNN	RF	Overhead
<b>Original</b>	76.7%	90%	83.9%	91.1%	-
<b>Skype</b>	40.90%	51.21%	43.68%	40.92%	3.45 %
<b>Facebook</b>	42.2%	47.77%	47.54%	41.12%	4.41%
<b>Game</b>	0.85%	1.45%	1.83%	1.42%	0.17%
<b>WhatsApp</b>	11.25%	12.93%	10.8 %	10.83%	0.93%
<b>Viber</b>	13.2%	12.58%	11.74%	11.78%	0.84%
<b>YouTube</b>	8.97%	2.58%	8.49%	9.1%	1.58%

# Chapter 6

## Metrics to compare and enhance obfuscation efficiency

Based on the results in previous chapters, we provided so far different obfuscation systems realizing our goal to defend against Traffic Analysis attacks. Our empirical obfuscation solutions were efficient in reducing the accuracy of classifiers. However, the validation accuracy does not always imply best testing results. A main issue for the robustness of those techniques is to check how much effort do they require, and how much overhead do they induce. In this context, our aim in this Chapter is to have a measure that reflects the model performance as a function of the amount of overhead and secrecy realized. On the other hand, there is an urge to explain the empirical results of obfuscation models by exploring metrics that reflect their efficiency. What is needed is a criteria for selecting tunable parameters to achieve best results of the obfuscation model. The concept of measure function would help in explaining and predicting theoretically the obfuscation results for a specific defense system.

### 6.1 Comparison Criteria and Methodology

To compare between two obfuscation methods for protecting against statistical traffic analysis and to estimate the success of the proposed obfuscation scheme, different aspects have to be considered:

- First, the impact on the classifier's performance of the considered obfuscation method. This includes the performance characteristics of a network flow that could be affected such as data and bandwidth overhead, delay, throughput degradation, etc.
- Second, model robustness is key when comparing different obfuscation methods. The model robustness is related to its capability of protecting different data types.

- Third, the computational overhead comprehending the complexity created by the obfuscated program in terms of resource requirements such as compilation time, memory, and processing. The obfuscation process needs to be lightweight, especially when applied in real-time processing mode.
- Fourth, the practicality and dynamicity. This measures how much handfult and easy to use is the algorithm.

### 6.1.1 Defense Evaluation

We propose in this subsection a measure to evaluate obfuscation techniques against traffic analysis attacks. This measure would describe how good a certain obfuscation model is and would compare between different defense systems. We name it *Obfuscation Effectiveness* and we define it using 2 metrics: the *attacker confidence*, and the *data overhead*.

The *attacker confidence* is defined as the estimated ratio of correct app inferences to attempted app inferences, by an attacker, with no previous knowledge if an obfuscation technique is already deployed. Lesser attacker confidence indicates that the technique is more efficient at defending user privacy. With no protection, attack confidence could reach 1.

*Data overhead* is the ratio of network data sent with and without a certain obfuscation technique. For instance, a bandwidth overhead of 1.5 indicates that applying the technique ends up with 1.5 times as much bytes traffic sent on unprotected network traffic. This ratio is generally greater than 1. Of course, a smaller data overhead is desirable because additional traffic leads to more network congestion.

Eventually, the *Obfuscation Effectiveness* is the product of *attacker confidence* and *data overhead*. Some obfuscation models have a fixed product, but others are adjustable to users' preferences for privacy versus data expenses.

We present in table Table 6.1, *Obfuscation Effectiveness* of Random Forest classifier for some obfuscation algorithms suggested through our work. The smaller is *Obfuscation Effectiveness*, the more the defense model has a good performance in terms of secrecy realized and resulting data overhead.

## 6.2 Information Theory

Information entropy is a mathematical concept that measures how much information there is in an event [165]. It was widely implemented in security systems to detect traffic anomalies [166, 167], or for systems of DDoS defense [168]. Most of these measures have never found any application in machine learning and network obfuscation systems. We aim to provide a better understanding of the implementation of entropy-based methods in obfuscation systems against traffic analysis attacks.

Table 6.1: Obfuscation effectiveness

Obfuscation Model	Version	Obfuscation Effectiveness
AdaptiveMutate	Packet length mutation (2nd version)	0.369
AdaptiveMutate	Inter-arrival time mutation	0.071
AdaptiveMutate	Lengths and IAT mutation	0.084
Anonymization through probabilistic distribution	First alg. (Mutation to Normal distribution of <i>Skype</i> )	0.087
Anonymization through probabilistic distribution	First alg. (Mutation to Poisson distribution of <i>Viber</i> )	0.137
Anonymization through probabilistic distribution	Second algorithm	0.16
Mutation To similar probability distribution	-	0.002
Optimal Analytical Solution	-	0.019
Opriv	-	0.015

For instance, Shannon entropy and Kullback-Leibler divergence (also known as information divergence or relative entropy) are the two most important quantities in information theory and its applications. In the attempt to explain the efficiency of our obfuscation systems, we present in what follows metrics of information to test their performance namely: degree of traffic masking effectiveness, Kullback-Leibler divergence, traffic divergence.

### 6.2.1 Degree of traffic masking effectiveness

Given a random variable  $X$  which takes values of the finite set of  $M$  values  $x_1, x_2, \dots, x_M$  and  $p_i := P(X = x_i)$  is the probability of occurrence of  $x_i$ ; hence, the Shannon entropy is:

$$H = - \sum_{i=1}^M p_i \log(p_i)$$

The conditional entropy of a variable  $Y$  given  $X$  of  $M$  values is defined by:

$$H(Y/X) = \sum_{i=1}^M p(X = i) H(Y/X = i)$$

Consider a traffic network and its set of  $p$  features  $X = X_1, X_2, \dots, X_p$ , the degree of traffic masking effectiveness using a certain obfuscation algorithm that mutates any of the  $p$  features, can be studied in terms of conditional entropy. Most traffic analysis attacks use information derived from the sequences of packet

lengths, and/or from interarrival times as input. This motivates our definition of degree of traffic masking effectiveness to measure how much the adversary can learn about a certain network application given the observed packet lengths or gap times leaked by the traffic.

Let  $\Phi$  be the set of packets that refer to a specific application, and let  $\Psi$  the side information on  $\Phi$  gained by the adversary by observing the encrypted traffic. The degree of masking effectiveness is measured by the conditional entropy  $H(\Phi/\Psi)$  to denote how much information an adversary can learn about the identification of a certain application given the observed traffic features. Accordingly, the smaller is  $H(\Phi/\Psi)$ , the more likely  $\Phi$  is a function of  $\Psi$ . This means that mutating or modifying any pattern of  $\Psi$  would sincerely affect the identification of  $\Phi$ . Moreover, a high  $H(\Phi/\Psi)$  indicates that  $\Phi$  is less dependent on  $\Psi$ . Hence, no matter how much the obfuscation algorithm mutates  $\Psi$ , that would not significantly affect the fingerprinting of  $\Phi$ .

As a proof of concept, we study the degree of masking effectiveness  $H(\Phi/\Psi)$  for the apps in our dataset. We evaluate  $H(\Phi/\Psi)$  for variables of side information being packet lengths, and packet interarrival time IAT. We present the results in Table 6.2.

Table 6.2: Degree of traffic masking effectiveness

$\Phi \backslash \Psi$	Lengths	IAT
Skype	0.3776	0.9336
Facebook	0.4374	0.7545
Game	0.1168	0.7530
WhatsApp	0.0451	0.3511
Viber	0.1862	0.6020
YouTube	0.2635	0.4362

According to Table 6.2, the values of  $H(\Phi/\Psi)$  with  $\Psi$ =packet lengths, are relatively low indicating a high dependency of our apps on their packet lengths. In this case, any security algorithm that would mutate the lengths effectively, would obfuscate the adversary knowledge of the exact application using traffic analysis. The impact of mutation of lengths of *Game*, *WhatsApp*, *Viber*, *YouTube* would be foreseen and tangible as  $H(\Phi/\Psi)$  for those apps are relatively minor in comparison to those of *Skype* and *Facebook*. This could explain results in Table 5.16, where *OPriv* appeared to be less efficient in cases of *Skype* and *Facebook* in comparison to the other apps.

The values of  $H(\Phi/\Psi)$  with  $\Psi$ =packet IAT, are relatively high. This means that in our case of traffic traces, the knowledge of the exact type of application does not mainly depend on IATs. Any obfuscation algorithm that modifies the IAT would be less effective in obscuring the application identity.

This is proved by simulation of our previously proposed method in Chapter 4, *AdaptiveMutate*, in which we mutate packet lengths, or inter-arrival time information separately. Simulation results showed a significant efficiency of *AdaptiveMutate* when applied using lengths mutation. However, IAT mutation was not efficient enough in reducing the classifier’s accuracy.

## 6.2.2 Kullback-Leibler divergence

We presented in Sections 4.1 and 4.2, algorithms that mutate a source app to a target app in the aim to defend against network application fingerprinting against traffic analysis. We used a trial-and-error process where we mutated comprehensively using our algorithm each application to all the other apps in order to decide on the best target app to mutate to. We will present instead in this part the Kullback-Leibler divergence, a mathematical defined metric to decide on which target app is best to mutate to.

Given two complete discrete probability distributions  $P=(p_1, p_2, \dots, p_n)$ , and  $Q=(q_1, q_2, \dots, q_n)$  where  $p_i, q_i \in [0,1]$  and  $\sum_{i=1}^n p_i = \sum_{i=1}^n q_i = 1$ . The Kullback-Leibler divergence is a measure of divergence between P and Q. According to [169], the Kullback-Leibler  $D(P || Q)$  is obtained using the formula:

$$D(P || Q) = \sum_{i=1}^n (p_i \log(\frac{p_i}{q_i}))$$

The Kullback-Leibler divergence can be used as a decision function to decide on which network application in a dataset to mutate to. In our context of network obfuscation assessment, we denote by P the probability distribution of a feature in the source app, and by Q that of the destination app to mutate to. The smaller  $D(P || Q)$  is, the closer the distributions P and Q are, and vice versa. A better obfuscation efficiency would be realized in cases where Q is chosen in a way such that  $D(P || Q)$  is large. Since more the source and destination app feature probability distributions are different, the more there is changeable features of the source app, and consequently would confuse it as the destination app by the classifier. The changeable features make the accurate identification of network application more difficult. Whereas, in case P and Q are similar, the mutation of P would not change much its distribution making the confusion less significant.

Table 6.3 presents KL divergence for lengths probabilities of the different apps and explains results of the obfuscation algorithms presented in Sections 4.1 and 4.2 where mutation to target apps with higher KL values of length probabilities lead to better obfuscation results.

According to Table 6.4, KL divergence for IAT probabilities estimation of the different apps are close to zero and therefore are very similar. Hence, there is no changeable features in case of IAT mutation making the confusion of a classifier less significant. This is proved in results of *AdaptiveMutate* when we try to mutate IATs.

Table 6.3: Kullback-Leibler divergence of packet lengths probabilities

P \ Q	Skype	Facebook	Game	WhatsApp	Viber	YouTube
Skype	0	1.7775	2.5947	2.5188	2.2243	2.6258
Facebook	1.9620	0	4.2658	3.1426	4.2831	3.2296
Game	2.3262	4.1728	0	3.0878	2.3356	2.1204
WhatsApp	2.9386	3.1240	3.1057	0	3.8656	4.2221
Viber	2.6613	2.7377	3.6844	3.1471	0	2.3251
YouTube	2.6019	3.1697	2.1257	4.2374	2.8794	0

Table 6.4: Kullback-Leibler divergence of IAT probabilities

P \ Q	Skype	Facebook	Game	WhatsApp	Viber	YouTube
Skype	0	0.0006	0.0002	0.0226	0.0655	0.0017
Facebook	0.0009	0	0.0018	0.0194	0.0617	0.0006
Game	0.0002	0.0009	0	0.0231	0.0665	0.0019
WhatsApp	0.0727	0.0512	0.0797	0	0.0500	0.0324
Viber	0.2863	0.2154	0.3529	0.073	0	0.1933
YouTube	0.0033	0.0011	0.0039	0.0158	0.0592	0

### 6.2.3 Traffic divergence

Traffic divergence  $TD(P || Q)$  between the original and mutated traffic could be expressed in terms of Kullback-Leibler divergence. P denotes in this case the probability distribution of a certain side information before mutation, and Q that after mutation. Hence,  $TD(P || Q)$  measures to which extent a certain obfuscation algorithm can modify the probability distribution of a side information of an app traffic. TD reflects on the efficiency of the obfuscation algorithm, and on the amount of overhead resulting from it. A high TD indicates a high change of the probability distribution of the specified side information feature. If this feature brings small degree of traffic masking (detailed in 6.2.1 of this section), that would result in an efficient protection against traffic analysis. However, a high traffic divergence could be reflected in a higher overhead. TD could be used to compare between obfuscation algorithms. It gives an insight of how much the algorithm causes modification and overhead. The finest obfuscation algorithm is the one that mutates a side information with least degree of traffic masking and results in least TD at the same time. Table 6.5 presents traffic divergence between original traffic of our app dataset, and the mutated traffic using *OPriv*.

Table 6.5: Length Traffic Divergence TD between original and mutated traffic using *OPriv*

App	Length TD
Skype	4.9948
Facebook	6.5112
Game	3.2843
WhatsApp	4.2030
Viber	3.8317
YouTube	4.8685

### 6.3 Conclusion

In this Chapter, we were able to come up with a metric that characterizes a defense model as a function of the amount of overhead and protection realized. Also, we provided novel metrics based on entropy to evaluate, for any obfuscation system, the right choice of features to mutate, as well as the right choice of target applications to mutate to. Our metrics based on concepts of information theory are namely: degree of traffic masking effectiveness, Kullback-Leibler divergence, traffic divergence. The information metrics presented in this research proved for the first time the obfuscation results of our work and of other previous works. This would give a significant understanding of obfuscation results for any thwarting technique.



# Chapter 7

## Conclusion

An increasing amount of research is centered nowadays on network traffic classification, since it can be convenient for the enforcement of management policies, traffic filtering, and QoS support mechanisms. Mainly, techniques of classification based on machine learning and statistical traffic analysis have been validated. These methods extract some features of the flows (e.g. packet lengths, inter-arrival times, direction) and exploit it to conclude the class of application or service running the traffic, among some potential alternatives. This type of privacy violation is also threatening other supposedly end-to-end secure networks such as web pages identification, and voice communication detection.

Security is a challenging concern and a key requirement for network designers. With network applications, security implications are well pronounced since the impact of attacks is drastic. This is mainly because those applications are central to our life including a wide variety of them, from simple location awareness to very critical healthcare uses. In fact, current implementations of network applications still reveal traffic features out of encrypted traffic. These can be exploited by machine learning techniques to obtain leakage on the information carried compromising their anonymity and secrecy. Published studies to date have not come up with an ideal solution to prevent such attacks. In this thesis, we addressed the privacy protection of network communication against side information based inferential attacks. We aimed at investigating optimal methods to secure a network traffic under the scrutiny of an adversary who uses statistical traffic analysis. Our goal was to find an optimum masking that shapes the packet flow, and sends it over the insecure network.

To do so, we started by defining the security model highlighting the threat, and we evaluated empirically a method for the identification of mobile apps using traffic analysis. Our results provided tangible information to speculate the accuracy of a powerful adversary, that is equipped to take over on a victim's encrypted traffic, and to de-anonymize the identify of the actual source app. Then, we presented practical implementable methods for obfuscation. In this context, we considered techniques such as mutation and morphing. Mutation

aims at modifying the traffic characteristics in the aim of confusing the classifier by padding and shaping the traffic. On the other hand, morphing aims at imitating the traffic characteristics to make a flow look like another flow.

We presented 3 models as preliminary obfuscation techniques to mitigate the problem. First, *AdaptiveMutate*, a generalized method that has 3 variations and included mutating packet lengths, and/or inter-arrival time information. Next, we proposed a confusion system based on the probabilistic distribution of packet sizes and does not require another app's model traffic. We regenerated an app packet sizes from its best-fitted distribution model. Then, we mutated packet lengths of a source app that we are trying to protect to regenerated packet sizes of another target app. Also, we undertook this problem by defining a model that mutates the packet lengths of an app to protect to those lengths from another app having similar bin probability.

However, the proposed methods do not prevent adversarial attacks totally and would create sometimes large overhead. And, thus an optimal obfuscation system should account for such problems yielding a trade-off between privacy and overhead/complexity of the masking algorithm. Also, it would offer a theoretical proof on the amount of overhead necessary to achieve perfect secrecy. For this reason, we defined a mathematical model for network obfuscation. We formulated analytically the problem of selecting the target app and the length from the target app to mutate to to optimize the anonymization accuracy while minimizing the overhead. Then, we proposed two methods to solve it: an analytical solution, and *Opriv*. These techniques outperform the other state-of-the-art defense solutions against traffic analysis and ensure minor information leak.

Last but not least, we suggested quantitative metrics to compare the characteristics and feasibility of an obfuscation strategy. In this aim, we relied on information theory and entropy to propose new measures for our obfuscation system's resilience to traffic analysis attacks.

As a future work, we plan to devise a traffic morphing technique that confuses a classifier of one class of traffic by making it look like another class. A mathematical formulation of the morphing problem could be provided to obtain an optimal solution. Moreover, we will focus on doing more measurements to validate our defense models against traffic analysis attacks. For the sake of comprehensiveness and to obtain a better understanding of the defense mechanisms effectiveness, we perhaps need to obtain more results from experiments and to test our proposed techniques on different applications and types of traffic. Also, we could test our defense mechanisms against different classifier models. Besides, we plan to integrate and implement our designs in a general network architecture. We firmly believe that our proposed solutions will have a huge impact on the privacy of network communications.

# Bibliography

- [1] C. Callegari, A. Coluccia, A. D’Alconzo, W. Ellens, S. Giordano, M. Mandjes, M. Pagano, T. Pepe, F. Ricciato, and P. Zuraniewski, “A methodological overview on anomaly detection,” in *Data traffic monitoring and analysis*, pp. 148–183, Springer, 2013.
- [2] M. Aloqaily, S. Otoum, I. Al Ridhawi, and Y. Jararweh, “An intrusion detection system for connected vehicles in smart cities,” *Ad Hoc Networks*, vol. 90, p. 101842, 2019.
- [3] A. L. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” *IEEE Communications surveys & tutorials*, vol. 18, no. 2, pp. 1153–1176, 2015.
- [4] E. A. Shams and A. Rizaner, “A novel support vector machine based intrusion detection system for mobile ad hoc networks,” *Wireless Networks*, vol. 24, no. 5, pp. 1821–1829, 2018.
- [5] F. Pasqualetti, F. Dörfler, and F. Bullo, “Attack detection and identification in cyber-physical systems,” *IEEE transactions on automatic control*, vol. 58, no. 11, pp. 2715–2729, 2013.
- [6] L. Scime and J. Beuth, “Anomaly detection and classification in a laser powder bed additive manufacturing process using a trained computer vision algorithm,” *Additive Manufacturing*, vol. 19, pp. 114–126, 2018.
- [7] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, “State-of-the-art deep learning: Evolving machine intelligence toward tomorrow’s intelligent network traffic control systems,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2432–2455, 2017.
- [8] Y. He, F. R. Yu, N. Zhao, V. C. Leung, and H. Yin, “Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach,” *IEEE Communications Magazine*, vol. 55, no. 12, pp. 31–37, 2017.

- [9] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, “Applications of deep reinforcement learning in communications and networking: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.
- [10] R. Dingledine, N. Mathewson, and P. Syverson, “Tor: The second-generation onion router,” tech. rep., Naval Research Lab Washington DC, 2004.
- [11] <http://anon.inf.tu-dresden.de/>.
- [12] A. K. Das, P. H. Pathak, C.-N. Chuah, and P. Mohapatra, “Contextual localization through network traffic analysis,” in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pp. 925–933, IEEE, 2014.
- [13] D. Herrmann, R. Wendolsky, and H. Federrath, “Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier,” in *Proceedings of the 2009 ACM workshop on Cloud computing security*, pp. 31–42, 2009.
- [14] C. V. Wright, L. Ballard, F. Monrose, and G. M. Masson, “Language identification of encrypted voip traffic: Alejandra y roberto or alice and bob?,” in *USENIX Security Symposium*, vol. 3, pp. 43–54, 2007.
- [15] A. M. White, A. R. Matthews, K. Z. Snow, and F. Monrose, “Phonotactic reconstruction of encrypted voip conversations: Hookt on fon-iks,” in *2011 IEEE Symposium on Security and Privacy*, pp. 3–18, IEEE, 2011.
- [16] Q. Wang, A. Yahyavi, B. Kemme, and W. He, “I know what you did on your smartphone: Inferring app usage over encrypted data traffic,” in *2015 IEEE Conference on Communications and Network Security (CNS)*, pp. 433–441, IEEE, 2015.
- [17] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, “Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic,” in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 439–454, IEEE, 2016.
- [18] B. Yamansavascular, M. A. Guvensan, A. G. Yavuz, and M. E. Karsligil, “Application identification via network traffic classification,” in *2017 International Conference on Computing, Networking and Communications (ICNC)*, pp. 843–848, IEEE, 2017.
- [19] S. Yu, “Big privacy: Challenges and opportunities of privacy study in the age of big data,” *IEEE access*, vol. 4, pp. 2751–2763, 2016.

- [20] A. D’Alconzo, I. Drago, A. Morichetta, M. Mellia, and P. Casas, “A survey on big data for network traffic monitoring and analysis,” *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 800–813, 2019.
- [21] S. Dong and R. Li, “Traffic identification method based on multiple probabilistic neural network model,” *Neural Computing and Applications*, vol. 31, no. 2, pp. 473–487, 2019.
- [22] S. Leroux, S. Bohez, P.-J. Maenhaut, N. Meheus, P. Simoens, and B. Dhoedt, “Fingerprinting encrypted network traffic types using machine learning,” in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–5, IEEE, 2018.
- [23] A. Aksoy, S. Louis, and M. H. Gunes, “Operating system fingerprinting via automated network traffic analysis,” in *2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2502–2509, IEEE, 2017.
- [24] Y. Xu, T. Wang, Q. Li, Q. Gong, Y. Chen, and Y. Jiang, “A multi-tab website fingerprinting attack,” in *Proceedings of the 34th Annual Computer Security Applications Conference*, pp. 327–341, 2018.
- [25] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen, “Automated website fingerprinting through deep learning,” *arXiv preprint arXiv:1708.06376*, 2017.
- [26] A. Kumar and T. J. Lim, “Early detection of mirai-like iot bots in large-scale networks through sub-sampled packet traffic analysis,” in *Future of Information and Communication Conference*, pp. 847–867, Springer, 2019.
- [27] J. Yu, F. Liu, W. Zhou, and H. Yu, “Hadoop-based network traffic anomaly detection in backbone,” in *2014 IEEE 3rd International Conference on Cloud Computing and Intelligence Systems*, pp. 140–145, IEEE, 2014.
- [28] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, “Analyzing android encrypted network traffic to identify user actions,” *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 1, pp. 114–125, 2015.
- [29] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, “Can’t you hear me knocking: Identification of user actions on android apps via traffic analysis,” in *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, pp. 297–304, 2015.
- [30] A. Bakhshandeh and Z. Eskandari, “An efficient user identification approach based on netflow analysis,” in *2018 15th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC)*, pp. 1–5, IEEE, 2018.

- [31] T. T. Nguyen and G. Armitage, “A survey of techniques for internet traffic classification using machine learning,” *IEEE communications surveys & tutorials*, vol. 10, no. 4, pp. 56–76, 2008.
- [32] <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml/>.
- [33] A. Tongaonkar, R. Keralapura, and A. Nucci, “Challenges in network application identification.,” in *LEET*, 2012.
- [34] N. Al Khater and R. E. Overill, “Network traffic classification techniques and challenges,” in *2015 Tenth International Conference on Digital Information Management (ICDIM)*, pp. 43–48, IEEE, 2015.
- [35] A. W. Moore and K. Papagiannaki, “Toward the accurate identification of network applications,” in *International Workshop on Passive and Active Network Measurement*, pp. 41–54, Springer, 2005.
- [36] Q. Xu, T. Andrews, Y. Liao, S. Miskovic, Z. M. Mao, M. Baldi, and A. Nucci, “Flowr: a self-learning system for classifying mobile application traffic,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 1, pp. 569–570, 2014.
- [37] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, “Acas: automated construction of application signatures,” in *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, pp. 197–202, 2005.
- [38] D. Qin, J. Yang, J. Wang, and B. Zhang, “Ip traffic classification based on machine learning,” in *2011 IEEE 13th International Conference on Communication Technology*, pp. 882–886, IEEE, 2011.
- [39] J. Dromard, P. Owezarski, V. Mozo, A. Ordozgoiti, and B. Vakaruk, *Deliverable Algorithms Description: Traffic pattern evolution and unsupervised network anomaly detection ONTIC D4. 2*. PhD thesis, CNRS-LAAS; Universidad politécnica de Madrid, 2016.
- [40] B. Ma, H. Zhang, Y. Guo, Z. Liu, and Y. Zeng, “A summary of traffic identification method depended on machine learning,” in *2018 International Conference on Sensor Networks and Signal Processing (SNSP)*, pp. 469–474, IEEE, 2018.
- [41] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, “Robust smartphone app identification via encrypted network traffic analysis,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 63–78, 2017.

- [42] J. S. Atkinson, J. E. Mitchell, M. Rio, and G. Matich, “Your wifi is leaking: What do your mobile apps gossip about you?,” *Future Generation Computer Systems*, vol. 80, pp. 546–557, 2018.
- [43] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, “A systematic approach to developing and evaluating website fingerprinting defenses,” in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 227–238, 2014.
- [44] G. Ateniese, B. Hitaj, L. V. Mancini, N. V. Verde, and A. Villani, “No place to hide that bytes won’t reveal: Sniffing location-based encrypted traffic to track a user’s position,” in *International Conference on Network and System Security*, pp. 46–59, Springer, 2015.
- [45] K. Park and H. Kim, “Encryption is not enough: Inferring user activities on kakaotalk with traffic analysis,” in *International Workshop on Information Security Applications*, pp. 254–265, Springer, 2015.
- [46] S. Chen, R. Wang, X. Wang, and K. Zhang, “Side-channel leaks in web applications: A reality today, a challenge tomorrow,” in *2010 IEEE Symposium on Security and Privacy*, pp. 191–206, IEEE, 2010.
- [47] S. Dai, A. Tongaonkar, X. Wang, A. Nucci, and D. Song, “Networkprofiler: Towards automatic fingerprinting of android apps,” in *2013 Proceedings IEEE INFOCOM*, pp. 809–817, IEEE, 2013.
- [48] F. Zhang, W. He, Y. Chen, Z. Li, X. Wang, S. Chen, and X. Liu, “Thwarting wi-fi side-channel analysis through traffic demultiplexing,” *IEEE transactions on wireless communications*, vol. 13, no. 1, pp. 86–98, 2013.
- [49] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, “Traffic classification through simple statistical fingerprinting,” *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 1, pp. 5–16, 2007.
- [50] N. Williams, S. Zander, and G. Armitage, “A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification,” *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 5, pp. 5–16, 2006.
- [51] Y. Hu, D.-M. Chiu, and J. C. Lui, “Application identification based on network behavioral profiles,” in *2008 16th International Workshop on Quality of Service*, pp. 219–228, IEEE, 2008.
- [52] J. Höchst, L. Baumgärtner, M. Hollick, and B. Freisleben, “Unsupervised traffic flow classification using a neural autoencoder,” in *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, pp. 523–526, IEEE, 2017.

- [53] K. Singh, S. Agrawal, and B. Sohi, “A near real-time ip traffic classification using machine learning,” *International Journal of Intelligent Systems and Applications*, vol. 5, no. 3, p. 83, 2013.
- [54] N. Namdev, S. Agrawal, and S. Silkari, “Recent advancement in machine learning based internet traffic classification,” *Procedia Computer Science*, vol. 60, pp. 784–791, 2015.
- [55] F. Al-Obaidy, S. Momtahn, M. F. Hossain, and F. Mohammadi, “Encrypted traffic classification based ml for identifying different social media applications,” in *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, pp. 1–5, IEEE, 2019.
- [56] S. Rezaei and X. Liu, “Deep learning for encrypted traffic classification: An overview,” *IEEE communications magazine*, vol. 57, no. 5, pp. 76–81, 2019.
- [57] H. Zhou, Y. Wang, and M. Ye, “A method of cnn traffic classification based on sppnet,” in *2018 14th International Conference on Computational Intelligence and Security (CIS)*, pp. 390–394, IEEE, 2018.
- [58] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, “Deep packet: A novel approach for encrypted traffic classification using deep learning,” *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.
- [59] Z. Wang, “The applications of deep learning on traffic identification,” *BlackHat USA*, vol. 24, no. 11, pp. 1–10, 2015.
- [60] G. Levchuk, “Function and activity classification in network traffic data: existing methods, their weaknesses, and a path forward,” in *Machine Intelligence and Bio-inspired Computation: Theory and Applications X*, vol. 9850, p. 985004, International Society for Optics and Photonics, 2016.
- [61] L. Akoglu, H. Tong, and D. Koutra, “Graph based anomaly detection and description: a survey,” *Data mining and knowledge discovery*, vol. 29, no. 3, pp. 626–688, 2015.
- [62] J. Jusko and M. Rehak, “Identifying peer-to-peer communities in the network by connection graph analysis,” *International Journal of Network Management*, vol. 24, no. 4, pp. 235–252, 2014.
- [63] M. Iliofotou, H.-c. Kim, M. Faloutsos, M. Mitzenmacher, P. Pappu, and G. Varghese, “Graption: A graph-based p2p traffic classification framework for the internet backbone,” *Computer Networks*, vol. 55, no. 8, pp. 1909–1920, 2011.



- [64] Y. Jin, E. Sharafuddin, and Z.-L. Zhang, “Unveiling core network-wide communication patterns through application traffic activity graph decomposition,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 1, pp. 49–60, 2009.
- [65] K. Xu, F. Wang, and L. Gu, “Behavior analysis of internet traffic via bipartite graphs and one-mode projections,” *IEEE/ACM Transactions on Networking*, vol. 22, no. 3, pp. 931–942, 2013.
- [66] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, “BlinC: multilevel traffic classification in the dark,” in *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 229–240, 2005.
- [67] J. Cao, A. Chen, I. Widjaja, and N. Zhou, “Online identification of applications using statistical behavior analysis,” in *IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference*, pp. 1–6, IEEE, 2008.
- [68] W. Lu and L. Xue, “A heuristic-based co-clustering algorithm for the internet traffic classification,” in *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, pp. 49–54, IEEE, 2014.
- [69] Z. Nascimento, D. Sadok, S. Fernandes, and J. Kelner, “Multi-objective optimization of a hybrid model for network traffic classification by combining machine learning techniques,” in *2014 International Joint Conference on Neural Networks (IJCNN)*, pp. 2116–2122, IEEE, 2014.
- [70] H. Dong, G.-L. Sun, and D.-D. Li, “A hybrid method for network traffic classification,” in *Proceedings of 2013 2nd International Conference on Measurement, Information and Control*, vol. 1, pp. 653–656, IEEE, 2013.
- [71] C. V. Wright, L. Ballard, S. E. Coull, F. Monroe, and G. M. Masson, “Uncovering spoken phrases in encrypted voice over ip conversations,” *ACM Transactions on Information and System Security (TISSEC)*, vol. 13, no. 4, pp. 1–30, 2010.
- [72] S. Feghhi and D. J. Leith, “A web traffic analysis attack using only timing information,” *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 8, pp. 1747–1759, 2016.
- [73] R. Spreitzer, S. Griesmayr, T. Korak, and S. Mangard, “Exploiting data-usage statistics for website fingerprinting attacks on android,” in *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pp. 49–60, 2016.

- [74] T. Stöber, M. Frank, J. Schmitt, and I. Martinovic, “Who do you sync you are? smartphone fingerprinting via application behaviour,” in *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*, pp. 7–12, 2013.
- [75] Y. Fu, H. Xiong, X. Lu, J. Yang, and C. Chen, “Service usage classification with encrypted internet traffic in mobile messaging apps,” *IEEE Transactions on Mobile Computing*, vol. 15, no. 11, pp. 2851–2864, 2016.
- [76] Z. Chen, B. Yu, Y. Zhang, J. Zhang, and J. Xu, “Automatic mobile application traffic identification by convolutional neural networks,” in *2016 IEEE Trustcom/BigDataSE/ISPA*, pp. 301–307, IEEE, 2016.
- [77] H. Cheng and R. Avnur, “Traffic analysis of ssl encrypted web browsing,” *URL citeseer.ist.psu.edu/656522.html*, 1998.
- [78] G. D. Bissias, M. Liberatore, D. Jensen, and B. N. Levine, “Privacy vulnerabilities in encrypted http streams,” in *International Workshop on Privacy Enhancing Technologies*, pp. 1–11, Springer, 2005.
- [79] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, “Website fingerprinting at internet scale.,” in *NDSS*, 2016.
- [80] A. Iacovazzi and Y. Elovici, “Network flow watermarking: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 512–530, 2016.
- [81] L. Dixon, T. Ristenpart, and T. Shrimpton, “Network traffic obfuscation and automated internet censorship,” *IEEE Security & Privacy*, vol. 14, no. 6, pp. 43–53, 2016.
- [82] T. A. Ghaleb, “Techniques and countermeasures of website/wireless traffic analysis and fingerprinting,” *Cluster Computing*, vol. 19, no. 1, pp. 427–438, 2016.
- [83] X. Fu, B. Graham, R. Bettati, and W. Zhao, “On countermeasures to traffic analysis attacks,” in *IEEE Systems, Man and Cybernetics Society Information Assurance Workshop, 2003.*, pp. 188–195, IEEE, 2003.
- [84] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, “Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail,” in *2012 IEEE symposium on security and privacy*, pp. 332–346, IEEE, 2012.
- [85] B. Qu, Z. Zhang, L. Guo, X. Zhu, L. Guo, and D. Meng, “An empirical study of morphing on network traffic classification,” in *7th International Conference on Communications and Networking in China*, pp. 227–232, IEEE, 2012.

- [86] C. V. Wright, S. E. Coull, and F. Monrose, “Traffic morphing: An efficient defense against statistical traffic analysis,” in *NDSS*, vol. 9, Citeseer, 2009.
- [87] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, “Toward an efficient website fingerprinting defense,” in *European Symposium on Research in Computer Security*, pp. 27–46, Springer, 2016.
- [88] M. Imani, M. S. Rahman, and M. Wright, “Adversarial traces for website fingerprinting defense,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2225–2227, 2018.
- [89] D. Lu, S. Bhat, A. Kwon, and S. Devadas, “Dynaflow: An efficient website fingerprinting defense based on dynamically-adjusting flows,” in *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*, pp. 109–113, 2018.
- [90] P. Sirinam, M. Imani, M. Juarez, and M. Wright, “Deep fingerprinting: Undermining website fingerprinting defenses with deep learning,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1928–1943, 2018.
- [91] M. Liberatore and B. N. Levine, “Inferring the source of encrypted http connections,” in *Proceedings of the 13th ACM conference on Computer and communications security*, pp. 255–263, 2006.
- [92] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, “Website fingerprinting in onion routing based anonymization networks,” in *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*, pp. 103–114, 2011.
- [93] H. Liu, Z. Wang, and F. Miao, “Concurrent multipath traffic impersonating for enhancing communication privacy,” *International Journal of Communication Systems*, vol. 27, no. 11, pp. 2985–2996, 2014.
- [94] Y. Hu, X. Li, J. Liu, H. Ding, Y. Gong, and Y. Fang, “Mitigating traffic analysis attack in smartphones with edge network assistance,” in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2018.
- [95] Y. Gokcen, V. A. Foroushani, and A. N. Z. Heywood, “Can we identify nat behavior by analyzing traffic flows?,” in *2014 IEEE Security and Privacy Workshops*, pp. 132–139, IEEE, 2014.
- [96] P. Syverson, R. Dingledine, and N. Mathewson, “Tor: The secondgeneration onion router,” in *Usenix Security*, pp. 303–320, 2004.

- [97] Z. Yuan and C. Wang, “An improved network traffic classification algorithm based on hadoop decision tree,” in *2016 IEEE International Conference of Online Analysis and Computing Science (ICOACS)*, pp. 53–56, IEEE, 2016.
- [98] E. Hodo, X. Bellekens, E. Iorkyase, A. Hamilton, C. Tachtatzis, and R. Atkinson, “Machine learning approach for detection of nontor traffic,” in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, pp. 1–6, 2017.
- [99] A. Montieri, D. Ciunzo, G. Bovenzi, V. Persico, and A. Pescapé, “A dive into the dark web: Hierarchical traffic classification of anonymity tools,” *IEEE Transactions on Network Science and Engineering*, 2019.
- [100] M. AlSabah and I. Goldberg, “Performance and security improvements for tor: A survey,” *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, pp. 1–36, 2016.
- [101] S. Matic, C. Troncoso, and J. Caballero, “Dissecting tor bridges: a security evaluation of their private and public infrastructures,” in *Network and Distributed Systems Security Symposium*, pp. 1–15, The Internet Society, 2017.
- [102] V. Shmatikov and M.-H. Wang, “Timing analysis in low-latency mix networks: Attacks and defenses,” in *European Symposium on Research in Computer Security*, pp. 18–33, Springer, 2006.
- [103] X. Fu, B. Graham, R. Bettati, W. Zhao, and D. Xuan, “Analytical and empirical analysis of countermeasures to traffic analysis attacks,” in *2003 International Conference on Parallel Processing, 2003. Proceedings.*, pp. 483–492, IEEE, 2003.
- [104] W. Wang, M. Motani, and V. Srinivasan, “Dependent link padding algorithms for low latency anonymity systems,” in *Proceedings of the 15th ACM conference on Computer and communications security*, pp. 323–332, 2008.
- [105] B. N. Levine, M. K. Reiter, C. Wang, and M. Wright, “Timing attacks in low-latency mix systems,” in *International Conference on Financial Cryptography*, pp. 251–265, Springer, 2004.
- [106] T.-H. Cheng, Y.-D. Lin, Y.-C. Lai, and P.-C. Lin, “Evasion techniques: Sneaking through your intrusion detection/prevention systems,” *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 1011–1020, 2011.
- [107] A. Diyanat, A. Khonsari, and S. P. Shariatpanahi, “A dummy-based approach for preserving source rate privacy,” *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1321–1332, 2016.

- [108] P. Winter, T. Pulls, and J. Fuss, “Scramblesuit: A polymorphic network protocol to circumvent censorship,” in *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, pp. 213–224, 2013.
- [109] W. Mazurczyk, S. Wendzel, S. Zander, A. Houmansadr, and K. Szczypiorski, *Information hiding in communication networks: fundamentals, mechanisms, applications, and countermeasures*. John Wiley & Sons, 2016.
- [110] A. Iacovazzi and A. Baiocchi, “Internet traffic privacy enhancement with masking: Optimization and tradeoffs,” *IEEE transactions on parallel and distributed systems*, vol. 25, no. 2, pp. 353–362, 2013.
- [111] X. Fu, B. Graham, R. Bettati, and W. Zhao, “On effectiveness of link padding for statistical traffic analysis attacks,” in *23rd International Conference on Distributed Computing Systems, 2003. Proceedings.*, pp. 340–347, IEEE, 2003.
- [112] A. Iacovazzi and A. Baiocchi, “Protecting traffic privacy for massive aggregated traffic,” *Computer Networks*, vol. 77, pp. 1–17, 2015.
- [113] A. Iacovazzi and A. Baiocchi, “Padding and fragmentation for masking packet length statistics,” in *International Workshop on Traffic Monitoring and Analysis*, pp. 85–88, Springer, 2012.
- [114] A. Iacovazzi and A. Baiocchi, “Investigating the trade-off between overhead and delay for full packet traffic privacy,” in *2013 IEEE International Conference on Communications Workshops (ICC)*, pp. 1345–1350, IEEE, 2013.
- [115] A. Iacovazzi and A. Baiocchi, “Optimum packet length masking,” in *2010 22nd International Teletraffic Congress (ITC 22)*, pp. 1–8, IEEE, 2010.
- [116] L. Bernaille and R. Teixeira, “Early recognition of encrypted applications,” in *International Conference on Passive and Active Network Measurement*, pp. 165–175, Springer, 2007.
- [117] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, “End-to-end encrypted traffic classification with one-dimensional convolution neural networks,” in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp. 43–48, IEEE, 2017.
- [118] N. Apthorpe, D. Y. Huang, D. Reisman, A. Narayanan, and N. Feamster, “Keeping the smart home private with smart (er) iot traffic shaping,” *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 3, pp. 128–148, 2019.

- [119] A. Sheth, S. Seshan, and D. Wetherall, “Geo-fencing: Confining wi-fi coverage to physical boundaries,” in *International Conference on Pervasive Computing*, pp. 274–290, Springer, 2009.
- [120] I. Martinovic, P. Pichota, and J. B. Schmitt, “Jamming for good: a fresh approach to authentic communication in wsns,” in *Proceedings of the second ACM conference on Wireless network security*, pp. 161–168, 2009.
- [121] S. Lakshmanan, C.-L. Tsao, R. Sivakumar, and K. Sundaresan, “Securing wireless data networks against eavesdropping using smart antennas,” in *2008 The 28th International Conference on Distributed Computing Systems*, pp. 19–27, IEEE, 2008.
- [122] Y. Fan, B. Lin, Y. Jiang, and X. Shen, “An efficient privacy-preserving scheme for wireless link layer security,” in *IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference*, pp. 1–5, IEEE, 2008.
- [123] B. Greenstein, D. McCoy, J. Pang, T. Kohno, S. Seshan, and D. Wetherall, “Improving wireless privacy with an identifier-free link layer protocol,” in *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pp. 40–53, 2008.
- [124] K. Bauer, D. McCoy, B. Greenstein, D. Grunwald, and D. Sicker, “Physical layer attacks on unlinkability in wireless lans,” in *International Symposium on Privacy Enhancing Technologies Symposium*, pp. 108–127, Springer, 2009.
- [125] M. Gruteser and D. Grunwald, “Enhancing location privacy in wireless lan through disposable interface identifiers: a quantitative analysis,” *Mobile Networks and Applications*, vol. 10, no. 3, pp. 315–325, 2005.
- [126] T. Jiang, H. J. Wang, and Y.-C. Hu, “Preserving location privacy in wireless lans,” in *Proceedings of the 5th international conference on Mobile systems, applications and services*, pp. 246–257, 2007.
- [127] R. Attarian, L. Abdi, and S. Hashemi, “Adawfpa: Adaptive online website fingerprinting attack for tor anonymous network: A stream-wise paradigm,” *Computer Communications*, vol. 148, pp. 74–85, 2019.
- [128] D. Wagner, B. Schneier, *et al.*, “Analysis of the ssl 3.0 protocol,” in *The Second USENIX Workshop on Electronic Commerce Proceedings*, vol. 1, pp. 29–40, 1996.
- [129] R. Nithyanand, X. Cai, and R. Johnson, “Glove: A bespoke website fingerprinting defense,” in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pp. 131–134, 2014.

- [130] S. Li, H. Guo, and N. Hopper, “Measuring information leakage in website fingerprinting attacks and defenses,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1977–1992, 2018.
- [131] B. Saltaformaggio, H. Choi, K. Johnson, Y. Kwon, Q. Zhang, X. Zhang, D. Xu, and J. Qian, “Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic,” in *10th {USENIX} Workshop on Offensive Technologies ({WOOT} 16)*, 2016.
- [132] F. Schneider, A. Feldmann, B. Krishnamurthy, and W. Willinger, “Understanding online social network usage from a network perspective,” in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, pp. 35–48, 2009.
- [133] A. Iacovazzi and A. Baiocchi, “From ideality to practicability in statistical packet features masking,” in *2012 8th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 456–462, IEEE, 2012.
- [134] F. Tanjeem, M. Y. S. Uddin, and A. A. Rahman, “Wireless media access depending on packet size distribution over error-prone channels,” in *2015 International Conference on Networking Systems and Security (NSysS)*, pp. 1–7, IEEE, 2015.
- [135] E. Garsva, N. Paulauskas, and G. Grazulevicius, “Packet size distribution tendencies in computer network flows,” in *2015 Open Conference of Electrical, Electronic and Information Sciences (eStream)*, pp. 1–6, IEEE, 2015.
- [136] E. Garsva, N. Paulauskas, G. Grazulevicius, and L. Gulbinovic, “Packet inter-arrival time distribution in academic computer network,” *Elektronika ir Elektrotechnika*, vol. 20, no. 3, pp. 87–90, 2014.
- [137] M. Alasmar and N. Zakhleniuk, “Network link dimensioning based on statistical analysis and modeling of real internet traffic,” *arXiv preprint arXiv:1710.00420*, 2017.
- [138] C. Barakat, P. Thiran, G. Iannaccone, C. Diot, and P. Owezarski, “Modeling internet backbone traffic at the flow level,” *IEEE Transactions on Signal processing*, vol. 51, no. 8, pp. 2111–2124, 2003.
- [139] E. R. Castro, M. S. Alencar, and I. E. Fonseca, “Probability density functions of the packet length for computer networks with bimodal traffic,” *International Journal of Computer Networks & Communications*, vol. 5, no. 3, p. 17, 2013.

- [140] R. B. D'Agostino, *Goodness-of-fit-techniques*, vol. 68. CRC press, 1986.
- [141] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.
- [142] S. Theodoridis and K. Koutroumbas, "Pattern recognition—fourth edition, 2009."
- [143] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.
- [144] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, no. 1, pp. 3–24, 2007.
- [145] H. Liu and H. Motoda, *Computational methods of feature selection*. CRC Press, 2007.
- [146] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.
- [147] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.
- [148] L. Hu and L. Zhang, "Real-time internet traffic identification based on decision tree," in *World Automation Congress 2012*, pp. 1–3, IEEE, 2012.
- [149] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [150] X. Chen, J. Zhang, Y. Xiang, and W. Zhou, "Traffic identification in semi-known network environment," in *2013 IEEE 16th International Conference on Computational Science and Engineering*, pp. 572–579, IEEE, 2013.
- [151] C. Wang, T. Xu, and X. Qin, "Network traffic classification with improved random forest," in *2015 11th International Conference on Computational Intelligence and Security (CIS)*, pp. 78–81, IEEE, 2015.
- [152] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [153] J. Cao, Z. Fang, G. Qu, H. Sun, and D. Zhang, "An accurate traffic classification model based on support vector machines," *International Journal of Network Management*, vol. 27, no. 1, p. e1962, 2017.
- [154] R. Yuan, Z. Li, X. Guan, and L. Xu, "An svm-based machine learning method for accurate internet traffic classification," *Information Systems Frontiers*, vol. 12, no. 2, pp. 149–156, 2010.



- [155] G. Sun, T. Chen, Y. Su, and C. Li, "Internet traffic classification based on incremental support vector machines," *Mobile Networks and Applications*, vol. 23, no. 4, pp. 789–796, 2018.
- [156] N. Jing, M. Yang, S. Cheng, Q. Dong, and H. Xiong, "An efficient svm-based method for multi-class network traffic classification," in *30th IEEE International Performance Computing and Communications Conference*, pp. 1–8, IEEE, 2011.
- [157] Y. Hong, C. Huang, B. Nandy, and N. Seddigh, "Iterative-tuning support vector machine for network traffic classification," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 458–466, IEEE, 2015.
- [158] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE transactions on systems, man, and cybernetics*, no. 4, pp. 580–585, 1985.
- [159] G. Ajaeiya, I. H. Elhajj, A. Chehab, A. Kayssi, and M. Kneppers, "Mobile apps identification based on network flows," *Knowledge and Information Systems*, vol. 55, no. 3, pp. 771–796, 2018.
- [160] A. Moore, D. Zuev, and M. Crogan, "Discriminators for use in flow-based classification," tech. rep., 2013.
- [161] V. Petkov, R. Rajagopal, and K. Obraczka, "Characterizing per-application network traffic using entropy," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 23, no. 2, pp. 1–25, 2013.
- [162] M. Perényi and S. Molnár, "Enhanced skype traffic identification," in *Proceedings of the 2nd international conference on Performance evaluation methodologies and tools*, pp. 1–9, 2007.
- [163] D. Rossi, S. Valenti, P. Veglia, D. Bonfiglio, M. Mellia, and M. Meo, "Pictures from the skype," *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, no. 2, pp. 83–86, 2008.
- [164] G. He and J. C. Hou, "On sampling self-similar internet traffic," *Computer Networks*, vol. 50, no. 16, pp. 2919–2936, 2006.
- [165] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE mobile computing and communications review*, vol. 5, no. 1, pp. 3–55, 2001.
- [166] W. Han, J. Xue, and H. Yan, "Detecting anomalous traffic in the controlled network based on cross entropy and support vector machine," *IET Information Security*, vol. 13, no. 2, pp. 109–116, 2019.

- [167] C. Yang, “Anomaly network traffic detection algorithm based on information entropy measurement under the cloud computing environment,” *Cluster Computing*, vol. 22, no. 4, pp. 8309–8317, 2019.
- [168] K. Kalkan, L. Altay, G. Gür, and F. Alagöz, “Jess: Joint entropy-based ddos defense scheme in sdn,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2358–2372, 2018.
- [169] S. Kullback and R. A. Leibler, “On information and sufficiency,” *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

