ANALYSIS OF MULTILAYER-ENCRYPTION ANONYMITY NETWORKS

by

Khalid Shahbar

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
October 2017

It is my genuine gratefulness that I dedicate my thesis to the greatest mother, my mother. You are the reason I reach this stage. Your support started a long time before even when you taught me the alphabetic letters.

To my father, who passed away before I started my PhD., I wish to share this moment with you.

Table of Contents

List of Tables

List of Figures

Abstract

The main goal of multilayer-encryption anonymity networks is to provide a certain level of privacy to their users. At the same time, such networks could be misused to perform harmful network activities. Multilayer-encryption anonymity networks are blocked in some countries. Consequently, different obfuscation techniques are employed by some of these networks to bypass the censorship restriction and enable access to the network by the users.

This thesis studies and analyzes multilayer-encryption anonymity networks. Traffic flow analysis is employed to identify multilayer-encryption anonymity networks. The analysis includes collecting data from the three most popular anonymity networks (namely, Tor, JonDonym and I2P). The collected data (Anon17) is made publicly available for researchers on the field. The study also includes proposing weighted factors to quantify and measure the level of anonymity these networks could provide. The flow analysis is used to identify the multilayer-encryption anonymity networks and to identify the obfuscated traffic, if any.

Moreover, in this thesis, Packet Momentum is proposed to identify multilayer-encryption anonymity networks. Packet Momentum is a set of appropriate features which could identify multilayer-encryption anonymity networks. The proposed Packet Momentum achieved statistically significant high performance with a low number of packets and a low number of features.

List of Abbreviations Used

ARFF	Attribute-Relation File Format
BN	Bayesian Network
BT	BitTorrent
BW	Bandwidth
CV	Cross-Validation
DNS	Domain Name System
DPI	Deep Packet Inspection
FN	False Negative
FP	False Positive
FTE	Format-Transforming Encryption
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure

ICMP	Internet Control Message Protocol
IP	Internet Protocol
IRC	Internet Relay Chat
ISP	Internet Service Provider
NB	Naive Bayes
netDb	Network Database
NIMS	Network Information Management and Security
PCAP	Packet Capture
PT	Pluggable Transports
PTP	Peer to Peer
RF	Random Forest
SOM	Self-Organization Map

SSH	Secure Shell
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TN	True Negative
TP	True Positive
UDP	User Datagram Protocol

Acknowledgements

I would like to express my special appreciation and thanks to my supervisor Professor Nur Zincir-Heywood: you have been a great advisor. I would like to thank you for your support, knowledge, patience, motivation and understanding.

Besides my advisor, I would like to thank the rest of my thesis committee for their insightful comments, guidance and encouragement.

Finally, I would like to thank my family for all their patience, love, support and encouragement. Words cannot express how grateful I am. Thank you.

Chapter 1

Introduction

On December 16, 2013, during final exams time, Harvard University received bomb threats by email [33]. The police evacuated the threatened buildings and started investigating. After searching the buildings, the police did not find anything suspicious in any of the four threatened buildings. The investigation showed that the emails came from an anonymous email address using the Tor network [25]. At the end, it appeared that the emails were sent by a student to avoid the final exam. On the Tor network, the source of the email (Internet Protocol (IP) address) cannot be linked directly to the student (his IP address). Therefore, finding the source of the emails was achieved by using conventional police mechanisms and not by tracing the emails back to Tor. The student connected to the university Wireless's network to send the emails through Tor. It seems that only a few students were connected to Tor at the time the emails were received which finally led to the student who sent the emails.

The Tor network is one example of the multilayer-encryption anonymity networks which grant privacy to users by preserving their identities and their activities. On multilayer-encryption anonymity networks, messages between sender and receiver pass through multiple stations with multiple layers of encryption. Each station only knows the information necessary for passing the message. In this way, not even the final destination can link the message to the sender (more details in Chapter 2).

The growth of the amount of information collected by services websites such as search engines, web servers, or even ISP (Internet service provider) for the purpose of improving the services, data analysis, security enhancement, advertisement, or other reasons increased the demand and growth of multilayer-encryption anonymity networks. Not only this, but these networks facilitated reporting any kind of illegal activities to the authorities without the fear of exposing the reporter's identity, provided space with freedom to express thoughts and ideas and ensured the person's privacy for the journalists' (or other) sources [82]. These are examples and the list can be expanded to many others.

On the other hand, the environment of multilayer-encryption anonymity networks offered the possibility of performing harmful activities with little or no probability of tracing back the source of such harmful activities.

A Botnet is one of the threats to Internet users which infects computers to gain control through the botmaster. One way of communication between the bot and the botmaster is to use a Command and Control (C&C) server where frequently the bots contact the C&C server to get instructions from the botmaster. By blocking the server, the bots could not communicate with the botmaster anymore. Therefore, botnet developers seek more complex methods to communicate without being detected such as using Peer to Peer (PTP), Domain Generation Algorithm (DGA) or IP flux. Moreover, one of the attempts to build undetectable communication on the botnet is by using multilayer-encryption anonymity networks. For example, the 64-bit Zeus [94] botnet could use hidden services on the Tor network to communicate anonymously between the bots and the botmaster. The hidden services are one of Tor's services which allow anonymously hosting servers (such as a web server) on the Tor networks. The user accessing the server does not know the IP address of the server. Thus, the C&C server could be hosted anonymously on the Tor network. The Zeus botnet has not been the only attempt to implement an anonymous botnet over multilayer-encryption anonymity networks, there has been much research and multiple reports about using such an anonymous environment to hide botnet communications [83] [13] [14].

Multilayer-encryption anonymity networks relay users' traffic through multiple stations on the network on the way to the final destination. On the Tor network, the user has the option to select the last station (exit node) which the user's traffic will appear to come from. Also, the user can specify the exit nodes to be from a certain country. This option could be misused by users to access services that are not available, not legal or not allowed in their countries. This is similar to the Netflix bans of proxies and Virtual Private Networks (VPNs) [73] [15]. Some users access the U.S. Netflix content which contains materials only available to U.S. customers based on country-exclusive licensing agreements. By using a proxy or VPN, users change the IP address and have access to the content from outside the U.S., which

Netflix tries to prevent. The detection that the IP address belongs to a VPN or a proxy is easier when compared to finding that the IP address belongs to a multilayer-encryption anonymity network. For example, a list of the IP addresses of the VPN service providers could be blocked to prevent access from the VPN. Also, monitoring the number of logins from a specific IP address could help to identify that the IP address belongs to a VPN or a proxy. While discovering that an IP address belongs to an anonymity depends on the anonymity network being used.

Online anonymous marketplaces are another misuse of the multilayer-encryption anonymity networks. Silk Road, Silk Road 2.0, Evolution and Agora are examples of online anonymous marketplaces hosted on such networks where sellers and buyers can trade anonymously. These marketplaces offer illegal goods such as contraband, weapons, controlled substances and narcotics [27] [19] [26]. The size of the online anonymous marketplaces is about \$300,000-\$500,000 of sales daily [91]. Even though some of these marketplaces (such as Silk Road) are shut down by the authorities, the list keeps changing where new marketplaces appear and others just disappear.

There are some options for identifying multilayer-encryption anonymity networks. Each one of these methods has its weaknesses and disadvantages (detailed in Chapter 3). In general, these methods are costly and require resources which might be only available at the Internet Service Provider (ISP) level. In addition, some of the multilayer-encryption anonymity networks developed and implemented some obfuscation techniques (detailed in Chapter 2) to resist and avoid detection or being blocked by the censorship authorities.

1.1 Research Objectives

The main objective of this research is to investigate how far a machine-based learning approach can be pushed to analyze and identify different users and/or applications on such anonymity networks without decrypting the multi-layer encryption of their traffic. Potentially, such an investigation could shed light into how anonymous these networks are from a traffic analysis perspective. Also, it could provide an indication of what type of attacks could be performed against them. In order to achieve this, the research entails: (i) understanding the multilayer-encryption anonymity networks and measuring the anonymity level of such networks; (ii) collecting / capturing traffic of such networks for analysis; (iii) studying their traffic behaviour; (iv) extracting the best features to describe / model such behaviour; and (v) investigating the effects of running different applications on these anonymity networks.

The analysis of multilayer-encryption anonymity networks distinctly requires acquiring the data in the first place. Unfortunately, there is not any publicly available data set for the multilayer-encryption anonymity networks which covers the diversity of the design in such networks and includes the obfuscation techniques used on these networks (more on this in Chapter 5). Research in this field is based on collecting the required data by the researchers themselves while connecting to the anonymity network as a user(s) or running a station on the network [16] [3] [111] [62]. The challenge of providing publicly available data for anonymity networks is that collecting such data includes also collecting other users' data on the anonymity networks. This then requires additional care for ensuring the privacy of those users. The path a message on the anonymity networks takes could contain two or more stations until the message reaches its final destination. These stations serve multiple users at the same time; this improves the anonymity by making it hard to link the message with the sender. On some of the anonymity networks, users participate in relaying the messages of other users on the network. Therefore, collecting data even at the user's side requires considering how to deal with the privacy of the other users. This thesis provides the first publicly available multilayer-encryption anonymity network data which includes more than one anonymity network in addition to the data of the obfuscation techniques on such networks.

It is important when analyzing and studying the behaviour on anonymity networks to understand what affects the anonymity level and how to measure such incommensurable value. The anonymity set which is presented by Chaum [18] is a way to measure the level of anonymity on the multilayer-encryption anonymity networks. It presents the number of possible choices to which a message on the anonymity network belongs for a specific user. The higher the value of the anonymity set, the better the anonymity becomes. This way of measuring the anonymity level focusses on the probability of linking the message to the user. The level of anonymity is affected by many factors which increase or decrease the anonymity level. For example, the design of the anonymity networks and what level of information is available to the service provider (station) is an important factor to take into consideration when measuring the anonymity level. Therefore, in this research, weighted factors for measuring anonymity services are presented as another way to measure and quantify the anonymity level. The method takes into consideration multiple factors: quantifying, comparing and applying them to evaluate the anonymity level of different use cases (details in Chapter 4).

Another objective of this research is to analyze the traffic of the multilayer-encryption anonymity networks and profile the behaviour of their traffic within other non-anonymous network traffic. Obviously, the traffic of these networks is encrypted; the payload cannot be used to identify this type of traffic. In addition, the port number is not suitable for identifying applications nowadays. A flow analysis will be employed to extract the flow of the multilayer-encryption anonymity network and machine learning algorithms will be used to identify such networks. Furthermore, the usage of multilayer-encryption networks is blocked in some countries through censorship. Therefore, these networks employ different obfuscation techniques to resist being blocked by some censorship. The implemented obfuscation techniques make the identification of the multilayer-encryption anonymity network a very challenging task. For example, these techniques could change the traffic to look like random strings, or change the traffic to make the regex of the encrypted traffic to look like non-anonymous traffic such as HTTP, or even use services such as Google as a domain fronting to hide the traffic behind and make it hard to block through censorships (more in Chapter 8). There are other forms that the obfuscation techniques employ which makes it difficult to use one method to identify the multilayer-encryption anonymity network. The diversity in such obfuscation techniques is employed to evade the different known ways to identify encrypted traffic, especially the multilayer-encryption anonymity networks. In order to explore how much these obfuscation techniques could avoid detection by censorship, traffic flow analysis is used in this research to profile the behaviour of multilayer-encryption anonymity networks and the obfuscation techniques they use. Moreover, a packet-based approach called the Packet momentum technique is proposed and evaluated to investigate such challenges (more in Chapter 9).

1.2 Contributions

What follows is a summary of the contributions of this research.

1- Present the first publicly available data set for multilayer-encryption anonymity networks. Anon17 [5] [87] includes traffic flows for Tor, JonDonym and I2P. Moreover, Anon17 contains traffic of the obfuscation techniques as well as the applications running on top of the anonymity networks.

2- Propose a novel approach to quantify and measure the level of anonymity offered by the multilayer-encryption anonymity networks. The approach first presents five factors which have the highest influence on the users' anonymity. Then, these factors are compared with each other and assigned weights based on the comparison. Finally, the calculated weights and factors could be used together to measure the anonymity level of anonymity networks.

3- Analyze the traffic flow behaviours using machine learning algorithms to identify (i) traces of multilayer-encryption anonymity networks, (ii) applications running on top of the multilayer-encryption anonymity networks [85] [88] and (iii) obfuscated traffic generated by obfuscation tools which some of the multilayer-encryption anonymity networks have employed to avoid blockage or detection [86].

4- Packet Momentum; a novel approach proposed to identify multilayer-encryption anonymity networks efficiently and accurately and the obfuscations techniques they use. The Packet Momentum approach aims to use a small number of features and a small number of packets to identify such networks.

1.3 Structure

The rest of this thesis is structured as follows. Chapter 2 presents the concept of multilayer-encryption anonymity networks and the three anonymity networks analyzed in this research (Tor, JonDonym and I2P). Chapter 3 summarizes related work on Tor, JonDonym and I2P. Chapter 4 introduces the weighted factors to quantify and measure the anonymity level of anonymity networks. Chapter 5 presents Anon17, a network traffic data set of anonymity services. Chapter 6 discusses the methodology of this research which includes data collections, publicly available data employed in this research, machine learning algorithms and the network traffic flow exporter tools employed in this thesis. Chapter 7 presents the evaluations and results for the traffic analysis of the multilayer-encryption anonymity networks. In Chapter 8 the analysis of the obfuscation techniques implemented by multilayer- encryption anonymity networks is introduced. Chapter 9 presents the proposed Packet Momentum method to identify the multilayer-encryption anonymity networks. Conclusions are drawn and future works are discussed in Chapter 10.

Chapter 2

Overview of Multilayer-encryption Anonymity Networks

Multilayer-encryption anonymity networks aim to provide privacy to users by the separation of the users from their final destination. Multilayer-encryption anonymity networks counted on the Mix concept presented by Chaum [17] to provide anonymity to email. The sender sends the message indirectly to the receiver through multiple mixes. The message is encrypted multiple times. Therefore, each mix can decrypt only one layer of the encryption and see the part which belongs to that mix which is the address of the next destination to which the message is going to be sent. For the first mix, this address is the second mix on the path to the final destination. For the last mix, this address is the final destination. The sender encrypts the message multiple times starting with the public key of the last mix and ending with the public key of the first mix. Therefore, when the message arrives at the first mix, the message will be encrypted multiple times based on the number of mixes between the sender and the final destination. Assuming there are three mixes, then the message is encrypted with the public key of the last mix then the public key of the second mix then the public key of the first mix. The first mix then decrypts the message using its private key to obtain the address of the second mix and so on until the message reaches the final destination.

Choosing the mixes which will be included in the message's path between the sender and the receiver varies based on the design of the anonymity networks. This path could be fixed in a cascade way or a variable path based on a selection protocol. The number of mixes on the network, the operator of theses mixes and the bandwidth (BW) they offer to the users also count on the anonymity networks and their design. The following sections introduce the most popular anonymity networks: Tor, JonDonym and I2P [81] [110] [1].

2.1 Tor Network

Tor is a publicly available software which is built to provide users with anonymization when using the Internet. The Tor Network depends on the volunteers who run their machines as relay nodes to forward other users' traffic through Tor. The user starts by establishing a virtual circuit. This circuit consists of three relays. A directory server is used to get the addresses of available routers and their keys to encrypt the data. User data is encrypted with three layers of encryption. First, the data is encrypted with the last relay (exit router) key then the middle relay and finally, by the first relay (entry router). This way, the entry router only knows the source of the data. The middle routers only know where to get the encrypted data and where to send them, but nothing about the source of the data or the destination of them. Finally, the exit router knows the destination of the data but knows nothing about the source. This ensures that the whole path is not known by any of the three nodes (routers) on the Tor network.

The Cell is the building block in Tor. The user's data is divided into small fixed size Cells. The Cell size is 512 bytes. It consists of two parts; header and payload. The header contains information about the circuit that the Cell belongs to and the type of Cell. There are two types of Cells: Control Cells and Relay Cells. The Control Cells direct the relays what to do with the Cell. The Relay Cells contain the user's data. The structure of the Cell's header varies based on the type of the Cell. The header of the Cell consists of the circuit Identity (CircID) and a command. The command type distinguishes between the control Cell and the relay Cell. Commands like CREATE, CREATE_FAST, CREATED, CREATED_FAST, PADDING, DESTROY, NETINFO, CREATE2 and CREATED2 are control commands. They are used to manage the connection inside Tor. The payload of the control Cell contains different information based on the type of the command. For example, when the command is CREATE then the payload contains the handshake challenge. In cases for which the command is DESTROY then the payload holds the reason for closing the circuit. In addition, when the command is RELAY then the structure of the Cell is different from the control Cell. It means that this is a RELAY Cell. Each RELAY Cell has a relay header which comes between the Cell header and the payload. The relay header consists of 11 bytes. The components of the relay header are Stream ID, Digest, Recognized, Length and Relay command. RELAY_BEGIN, RELAY_DATA, RELAY_DROP, RELAY_TRUNCATE and RELAY_EXTEND are some forms of the relay commands.

The user and the relay use encrypted Transport Layer Security (TLS) connections to communicate. Circuits and cells are the components for Tor communication. The circuit is a virtual path from the user machine to the last relay (the exit relay). Inside the circuit, the data are packed into cells. Each cell contains the identifier of the circuit to which it belongs. The circuit ID is unique in the connection between the user and the relay. Through the path, the circuit ID is not the same.

Figure 2.1 shows the sequence for establishing the circuit. When the user wants to send data through Tor, he/she starts by establishing a circuit. The circuit establishment starts with the user of the first relay in a CREATE cell. When the circuit is established, the relay sends back the CREATED cell to the user. The user then extends the circuit to the second relay by using the RELAY_EXTEND cell. The first relay sends the CREATE cell to the second relay. The second relay responds back with the CREATED cell when the circuit is established between the first and the second relay. The first relay replies back to the user with the RELAY_EXTENDED cell to inform the user that the circuit path is now up to the second relay. Then, in the same way, the user extends the circuit to the exit relay. After the circuit is established between the user and the exit relay, the user is ready to send the data into RELAY_DATA cells.

The circuit path changes each time the user makes a connection. The bandwidth of the nodes and the policy of the exit node play a key role in selecting the best circuit path. Figure 2.2 shows how the circuit path could be different according to the path selection protocol. The circuit will be used for a short period; then, another circuit is created. The user has the option to fix the entry node and/or the exit node.

2.2 JonDonym Network

JonDonym/AN.ON is a network of mix cascades, providing anonymity to the users based on multilayer encryption [78]. The cascade consists of two (free) or three (paid) mix servers. The user starts the connection to the JonDonym network by selecting the mix cascade.

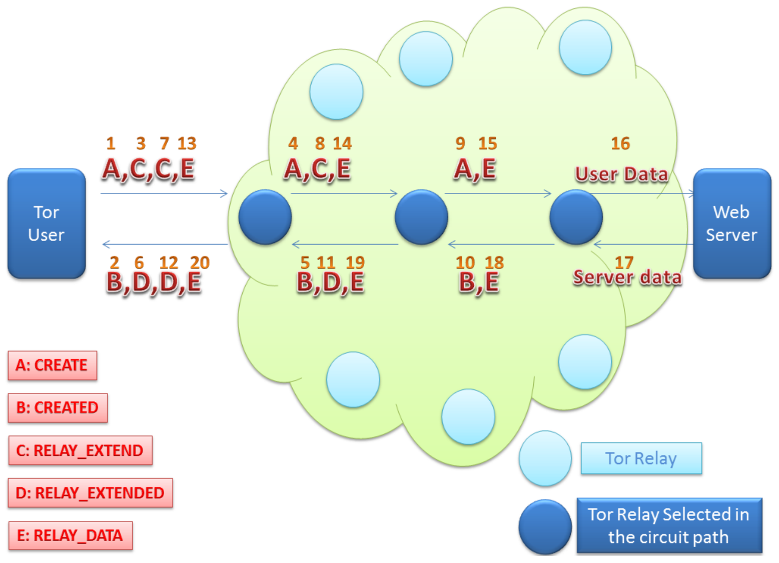


Figure 2.1: The Sequence for Establishing a Tor Circuit

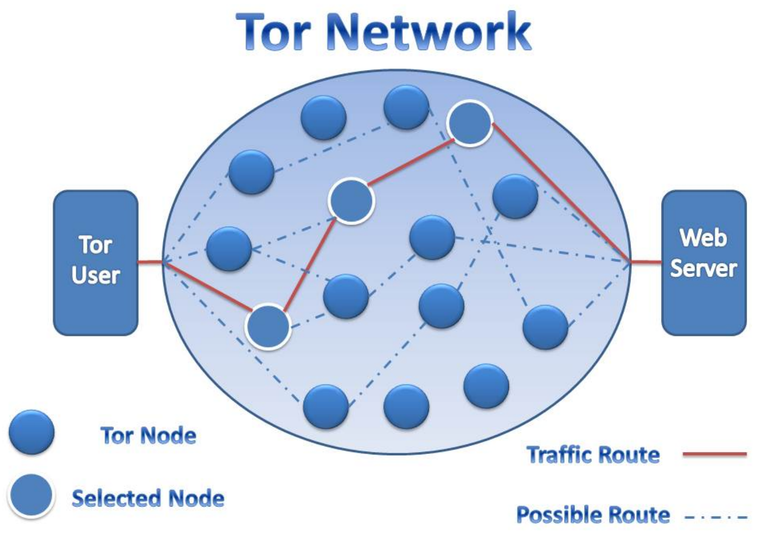


Figure 2.2: Path Selection on the Tor Network

Currently, there are five free Cascades and eleven paid Cascades the user can choose from. JonDo, previously known as JAP, is the client software that connects the user to the JonDonym network.

Only one active connection to one cascade is possible during the user's connection to the JonDonym network. Each HTTP request will create a connection from the browser (JonDoFox) with the client software JonDo. The JonDoFox browser can generate multiple connections with the JonDo. All these connections are multiplexed into one connection to the first mix server, which receives connections from multiple users. All the users' connections are then multiplexed into one Transmission Control Protocol/Internet Protocol (TCP/IP) connection to the second mix, or to the last in case of only two mixes in the cascade.

The information about the available Cascades, the number of users, the loads and the mix status are stored in the InfoService [55]. The user gets the information about the cascades from the InfoService and the last mix sends the users' requests to cache proxies. Multilayered encryption is used during the communication between the user and the last mix, which ensures that even the mixes cannot access the user's data. The path that the user's data takes is fixed based on the chosen cascade. To choose another path (Cascade), the user has to start a new connection to the JonDonym. The user can only have one connection to one cascade at any given time. Figure 2.3 shows that a user is connecting to only one cascade that has a fixed path. It also shows that there are other possible paths (cascades) to which the user connects.

When the connection is established, the IP address that is visible to the websites is the IP address of the last mix. JonDo and mixes use fixed-size packets called MixPacket. The first mix receives multiple packets from multiple users, which are then multiplexed and sent to the second mix. The MixPacket size is 998 bytes, consisting of 4 bytes for the channel ID, 2 bytes for flags and 992 bytes for the data field. The data field is readable only at the last mix. It contains 2 bytes of information about the length of the data, 1 byte of information about the type and 989 bytes for the payload.

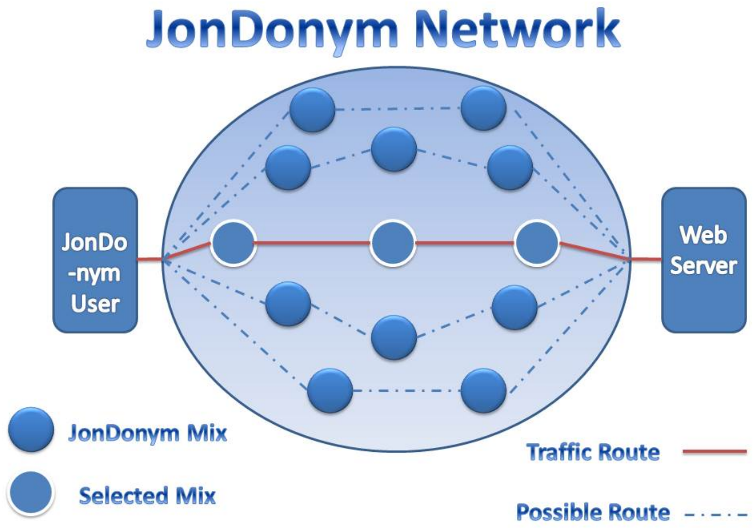


Figure 2.3: Cascades on the JonDonym Network

2.3 I2P Network

The I2P network is a decentralized anonymous network with no central database or server that contains the network database. The Network Database (netDb) [48] is distributed by using the Kademlia algorithm [65], which is used in many applications where P2P communication is needed in a decentralized network. The information that the user gets from the netDb enables the user to build tunnels. Sending and receiving data on the I2P network and building the knowledge about the network is done by building Inbound and Outbound Tunnels [107]. The tunnels are unidirectional [108]: the inbound tunnels are used by the users to receive messages and the outbound tunnels are used to send messages. The default configuration of the users' agents (clients) enables bandwidth participation, which means in addition to the user building his/her tunnels, the user can also participate in building other users' tunnels. The tunnels consist of two or more routers based on the client configuration and the tunnel type. Therefore, when the user participates in building tunnels, his/her role could be the first or the last or one in the middle in forming the tunnel. At the same time, the user could continue to send/receive his/her messages (if any). This aims to enhance the anonymity because it makes it harder to separate a specific user's tunnels from the other participating tunnels.

The netDb contains the leaseSet of the tunnels and routers. LeaseSet shows the routers involved in a tunnel. RouterInfo in the netDb shows how to contact a specific router. The user has the option to modify the number of routers in the outbound tunnel. I2P uses the concept of garlic routing [37], where layered encryption is implemented in addition to binding multiple messages together. The messages within the I2P network are encrypted end-to-end as long as the two communication parties are within the I2P network. However, when the user communicates with an end-system that is outside of the I2P network using an outproxy, then the encryption is not end-to-end.

By default, the user within the network transfers their data and that of other users where the user's machine functions as a resource for the network. The user can change the amount of bandwidth dedicated to the network from the console. The users' contributions in passing the network data are restricted by passing the data only within the I2P network. A different configuration is required when a user wants to pass the I2P traffic to an end-system outside of the I2P network (outproxy). The number of outproxies in the I2P network is limited.

One of the major differences between I2P and other anonymity networks such as Tor and JonDonym is that I2P is designed as a private network. The users mainly communicate within the network. The user builds two tunnels: inbound and outbound. The inbound tunnels are used to receive messages and the outbound tunnels are used to send messages. Figure 2.4 shows the inbound and outbound tunnels on the I2P network.

2.4 Summary

The Tor, JonDonym and I2P networks have similarities and differences. The three anonymity services share the goal of providing anonymity by relaying traffic to multiple stations using multiple layers of encryption. The multiple layers of encryption are used to harden and deny the link between the user and their messages. Tor and JonDonym mainly focus on providing anonymity for their users to access websites outside of their network. On the other hand, I2P provides anonymity to access websites hosted privately within the I2P network itself. At the same time, Tor also has websites hosted within the Tor network (hidden services). Moreover, I2P supports

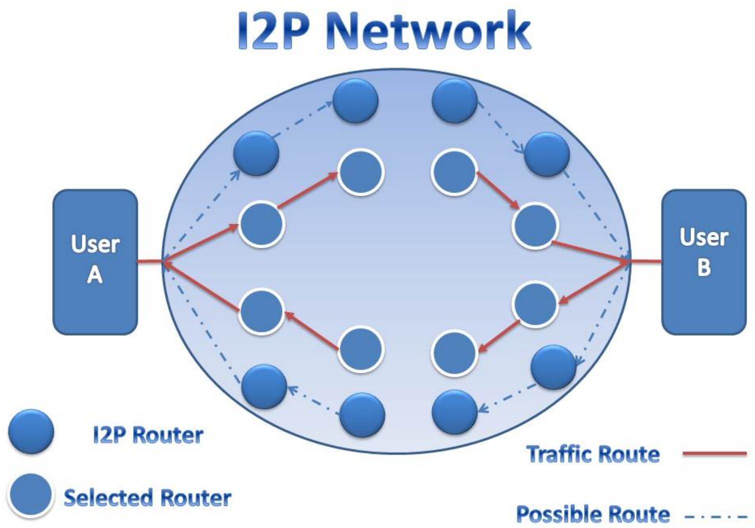


Figure 2.4: Inbound and Outbound Tunnels on the I2P Network

access to websites hosted on the Internet but not on the I2P network using an out-proxy. In terms of the path used to relay the traffic, the path on the Tor network and the I2P network changes and is not fixed, while the path on the JonDonym network is fixed. The duration that the user will stay connected to one path (circuit tunnels cascade) differs based on the anonymity system. The routing technique and the path selection also differ among the three anonymity services.

Chapter 3

Related Literature

Studies of anonymity networks have dealt with multiple aspects. For example, part of these studies worked on the improving and enhancement of the anonymity networks' performance. Others focussed on analyzing vulnerabilities and performing attacks. The following sections summarize the related studies of anonymity networks.

3.1 Measuring Anonymity

Measuring the level of anonymity is one of the concerns of the research field. Measuring the anonymity level is a challenge for a number of reasons. One is the difference in the design and the goal of each anonymity network. On the other hand, there is no single way to measure the anonymity levels of different anonymity networks. In addition, the anonymity level is not directly quantifiable compared to such other network traffic measurements as delay, bandwidth, volume, etc.

Ries *et al.* [81] evaluated five anonymization tools with regard to performance, usability, anonymity, network reliability and cost. The evaluated tools were the Tor, I2P, JonDonym, Perfect Privacy and Free proxies. The performance factors used to evaluate and rank these tools were Round Trip Time (RTT), Inter-Packet Delay Variation (IPDV) and throughput. Additionally, they used installation, configuration and verification of the anonymization connection as factors to define the usability of these tools. The anonymization of the tools was evaluated by using the rankings of the ability of an adversary to perform de-anonymization attacks against the tools. It should be noted here that these evaluations were limited to specific scenarios. Network reliability was measured using the failure rate, which in turn was measured by the mean time between failures (MTBF) and the mean time to recovery (MTTR).

Abou-Tair [1] *et al.* examined the usability of four anonymity tools (Tor, JonDo, I2P and Quicksilver) during the installation phase. They detailed the installation process of these tools, applying four tasks to test the installation phase: success of installation, success of configuration, confirmation of anonymization and ability to disable anonymization. To test the usability of these tools, the authors used eight guidelines taken from Clark [21], which focussed on the user's ability to perform the four tasks mentioned above.

Wendolsky *et al.* [110] compared Tor and AN.ON (JonDonym) from the user's perspective, based on performance and number of users. Latency and bandwidth were used to measure performance and the results showed that Tor performs unpredictably based on the time of day. In contrast, AN.ON (JonDonym) showed more consistent performance.

The above studies focussed mainly on evaluating anonymity services based on their performance or usability, where anonymity was not the focus of the evaluation. On the other hand, there are studies where measuring anonymity was the main goal. The idea of measuring anonymity is synchronized with the proposed ideas to develop anonymity by passing the message between the sender and the receiver through multiple stations until it reaches the final destination (the Mix concept) [17]. This concept aims to separate the ability of an attacker to link the sender and the receiver, even if they communicate over a channel observed by the attacker. To anonymize against such a threat model, Chaum [18] presented the concept of the anonymity set, in which the set is the total number of participants in the anonymity service which may include the sender. When the size of the set is increased, the anonymity level is considered to increase as well. Consequently, if the size of the anonymity set is one, then there is no anonymity and the attacker can identify the sender easily.

Serjantov and Danezis [84] developed the concept of the anonymity set by using the information-theoretic metric based on anonymity probability distribution.

Diaz *et al.* [23] also used an information-theoretic model to evaluate the anonymity level of a system in a particular attack scenario. The model aimed to evaluate the anonymity level of a system by finding the level of information the attacker can statistically gain to connect a user of the anonymity system to his messages. Shannon's definition of entropy is used to calculate this gain.

Murdoch [70] surveyed studies performed on measuring anonymity for low-latency anonymous networks and high-latency email anonymous networks and discussed the development of the techniques used for measuring anonymity.

There are other studies such as Berthold [9] and Toth [105] that also evaluate the anonymity level of the anonymity service in terms of the possibility of linking the message to the sender within the anonymity service.

3.2 Identifying Anonymity Networks by Discovering Infrastructure

Tor bridges [97] are special Tor nodes where their addresses are not announced publicly like the other nodes. A Tor user can connect to the Tor network using Tor bridges in cases where the IP addresses of normal nodes are blocked by censorship. The user can get up to three addresses of Tor bridges per day by sending an email to request these three IP addresses or by accessing Tor bridges web site and requesting the IP addresses. Ling *et al.* [61] used two different methods to reveal the addresses of Tor bridges. The first method was using bulk email accounts to request many bridge IP addresses daily. Since Tor allows only three IP address daily, they created 2,000 email accounts using different tools (iMacros, PlanetLab, the Tor network itself) to automate retrieving the bridge addresses by email and overcome the limitation of IP addresses. In addition to using the email to request the IP addresses of the bridges, 1,000 planetLab nodes were used to request bridge addresses through the web.

The second method seems to be more practical and effective than the bulk email accounts. Tor has its policy to classify routers inside the Tor network as entry, middle and exit routers. This policy includes but is not limited to weighting the bandwidth of each router, measuring their uptime, averaging the bandwidth available in the network, collecting reports about suspicious routers and applying the exit router's policy set by the router itself. By manipulating these factors, they aimed to insert one router in the Tor network and let the directory server chooses it as a middle router. Whenever a user tries to connect to the Tor network using a bridge, if the connection comes through the middle router to establish a connection to a bridge router, then the address of the bridge is obtained through the comparison of the announced entry routers and the address passed through the middle router. If the address of the router is not in the public list of all routers then this address is a bridge address.

The bandwidth of the routers and their number increased the probability of collecting more bridge addresses. To increase the chance of getting more bridge addresses, they used a probability comparison to find out if the number of injected routers or the weighted bandwidth is the main factor for increasing the selection probability of the malicious router as a middle router. They found that using the right bandwidth gives better results even with a limitation of the spread of routers. Using both methods, they discovered 2,365 bridges by using the email method and 2,369 bridges by using only one middle router for 14 days.

Li *et al.* [58] classified Tor nodes depending on their uptime. Nodes which have a higher bandwidth and longer uptime considered as super nodes. To prove that the Tor network has super nodes, data regarding the uptime of Tor nodes was collected from the Tor network. The collected data contained information about Tor nodes such as the IP address and the bandwidth. In addition to the collected data, information from Tor's official metric site was used as well. The high availability and bandwidth of these super nodes could decrease the anonymity of the Tor network. Using the information theory to measure the entropy of the anonymity of the Tor network showed less anonymity when super nodes were recognized. Brute-force attacks against super nodes also showed that they could affect the performance and availability on the Tor network. Super nodes present about 21% of the total nodes on the Tor network and provide 66% of the total bandwidth on the network. Controlling a super entry node or a super exit node has more effect than controlling a normal node. The analysis of the effect of discovering and attacking super nodes was simulated in large scales. The simulation included 3,000 nodes, Tor algorithms, a directory server and a path selection protocol. In the first case, attacks against the simulated Tor nodes assumed *a priori* knowledge of the locations of super nodes were known on the network. In the second case, attacks did not assume such *a priori* knowledge. Brute-force attacks were used for both cases. Results showed that when the super nodes were targetted with higher rates than the normal nodes, the failure on the network increased.

Liu *et al.* [62] presented four methods to discover the I2P routers. They discovered about 95% of all the I2P routers in their two-week experiment. One of their methods of discovering the I2P router was to run an I2P router and monitor the communications with other I2P routers to collect information about them. Another method was to run an I2P FloodFill router to monitor and collect information about routers that make communications with their FloodFill router. The third method for discovering the I2P router was the “crawling reseed URL”. This method used the reseed option in the I2P network to collect the I2P router’s information. The fourth method was “exploiting NetDB”, where the I2P mechanism of router query and response was used to collect router’s information.

Herrmann and Grothoff [42] presented an attack which determines the identity of the HTTP hosting peers (routers) on the I2P network. The attack required using three types of routers. The first type is used to provide information about the tunnel operations to the attacker. The second type is used to direct the user to select the attacker’s routers by performing a DOS attack. The third type is used to perform requests to the Eepsite. A combination of using the three types of routers was used for identifying the hosting router on the I2P network.

3.3 Identifying Application on Top of Anonymity Networks

Alsabah *et al.* [3] used ML algorithms to classify the application used by Tor’s users. The applications are Browsing, Streaming and BitTorrent. Given that Tor traffic is encrypted, they used the circuit level and the cell level information to do the classification. The circuit level information included the circuit lifetime and the amount of data transferred by the circuit. The cell level information included the cells’ inter arrival time and their statistics. The classification included online and offline classification. The online classification used the cell level information to classify the circuit while in use. The offline classification used both the cell and the circuit level information to classify the circuit. In terms of accuracy, the best result they achieved in the offline classification was 91%, whereas the best results achieved in the online classification was 97.8%.

Wagner *et al.* [109] proposed Torinj — a malware code that uses an image tag injection and a semi-supervised learning algorithm for identifying the number of users on Tor and their web browsing activities. A compromised or controlled exit node was assumed to collect the flow between the users and the web servers. All HTTP sessions were collected for further analysis. To classify each user’s session, a label propagation algorithm was used to relate a flow to a specific user. Torinj aims only to classify users’ browsing activities. It does not target either their identity or their location.

The image injection method could be detected by the user using a hash function to compare between a recently viewed web site and its injected version through the Tor network. This way the proposed image injection would not be able to capture the user's traffic. To overcome this, the authors suggested using a probability ratio to inject part of the session rather than all of them. Furthermore, users can harden the protection of their identity by using "Privoxy" to modify the browser information (text replacements, user agents, accepted language, etc.) so it is not shown or changed each time they start browsing. A game theoretic approach was used to solve similar detection issues where the goal was to find the best injection probability while keeping the user's tracking undetectable.

The set up for Torinj took place on the Tor network itself. Three main components were used in the set up: a Tor client, an intercepting proxy and a hidden command and control channel. BIND, a Domain Name System (DNS) server, TCPdump (for DNS queries captures), a Tor exit node, NTP (for time stamps), a Web proxy using Perl (to inject images), an IPtable (to route requests to the web proxy) and an Apache web server (to host images) were used for testing the injection method. SQLite3 and tcpick (a textmode sniffer) were used to process the collected flows.

Panchenko *et al.* [76] used a support vector machine (SVM) classifier with the websites' fingerprints to show that Tor and JAP are not providing total anonymity. The volume, time and direction of the traffic were the features in SVM. In the closed world (the set of known websites), the training data set consisted of browsing different websites using a computer running Debian distribution in the Tor network or JAP and logging the features using TCPdump. The Chickenfoot Firefox plugin was used to automate the browsing and retrieving activities. The data set contained 775 websites with 20 instances for each of them. On the other hand, in the open world part of their experiments (unknown web sites), the training data set included 4,000 URLs chosen from the 1 million most popular websites provided by Alexa [2]. Another 1,000 URLs were added to the test data. The features used in both data sets were: (i) Packet size (52 bytes) — the size of the packet that contains the ACK between the sender and the receiver; (ii) Size marker — a special text label that shows the change in the direction of the packet and is used to sum up all packets in the same direction; (iii) an HTML marker for distinguishing between each HTML request and to check the size of the requested site; (iv) Rounded transmitted bytes for the number of all transmitted packets in both directions (rounded in increments of 10,000); (v) Number of markers group the packet size for all the packets of a flow in one direction; (vi) Occurring packet size — the number of each site's packets in both directions; (vii) the Percentage of Incoming Packets; and (viii) the Number of packets. Using these features improved the results of detecting closed world websites from 3% to 55% in the Tor network and from 20% to 80% in JAP compared to other related works. In the open world data set, the detection was 73% with 0.05% false positive rate.

A simulated Tor network was used by Barker *et al.* [6] to differentiate between encrypted traffic and Tor traffic. The simulated network contained three directory servers and fifteen relays. Salenium browser (a Firefox add-on) was used to gather a sample of thirty website browsing traces. To do the comparison between the two types of encrypted traffic, Hypertext Transfer Protocol Secure (HTTPS) traffic was gathered alone for a period of two weeks then followed by HTTP traffic over the simulated Tor network for two weeks. Lastly, HTTPS traffic over the simulated network was gathered also for two weeks. In total, approximately 25 GB of data were collected for the three different tests. Each sample was 1 MB in size, so to build the whole captured data set, Mergecap was used. Different ML algorithms (Random Forest [RF], J4.8 and AdaBoost) were employed via WEKA. Their results showed that approximately 90% of the HTTP and HTTPS traffic on the Tor network could be detected with a false positive rate of 3.7%.

Timpanaro *et al.* [96] proposed a monitoring architecture for the I2P network to describe how it is used. The proposed system analyzed what type of applications are used on the I2P network. The applications that the monitoring architecture can identify are limited to web browsing and I2PSnark. The results showed that the proposed monitoring architecture could identify 32% of all running applications.

Egger *et al.* [31] presented several attacks that could be implemented against the I2P network. The authors claimed that their attacks against the I2P network could reveal the services that the I2P user accesses, the time of access and the time spent using the service. First, the attacks control most of the nodes that host the decentralized database (netDB) on the I2P network and then monitor the network activities to link the related ones. Denial of Service (DoS) could be used to disable the nodes hosting the netDB and speed the takeover process.

Westermann and Kesdogan [111] presented two attacks against the AN.ON network: the Redirection attack and the Replay attack. The Redirection attack aims to discover the websites which the user of the AN.ON visited. The attacker controls a web server and tries to redirect the user to this controlled web server. Once the user is redirected to the controlled web server, the attacker uses a Cascading Style Sheet (CSS) attack to recover the websites visited by the user. The Replay attack aims to correlate the user's HTTP request with a web server. The attacker monitors and records the user's communications with the first mix. The recorded messages are then replayed by the attacker to the cascade. The attacker then monitors the time it takes to get the response back. At the same time, the attacker communicates with the web server to find a pattern that confirms that the user was communicating with this web server.

3.4 Discovering Hidden Services

The Tor protocol was used by Ling *et al.* [59] to get information about the hidden servers. It was used to locate servers that use the Tor network to provide hidden services. Several entry routers, a client, a rendezvous point and a central server were required to relate the hidden server with its Tor feature. This requires the controlled client to connect to the hidden server while the entry routers are watching for different cell types that have special combinations. It is not necessary that the hidden server will choose the entry routers in its path, so watching is required until the entry routers are chosen by the server. Since the entry routers might be used by other users, cells were manipulated for the client in order to let them get an error from the hidden server when trying to decrypt them. The hidden server then disconnects the established circuit with the client. In this way, the entry routers could confirm that the path comes through them. If that were the case, then the hidden server could be located by the information gathered by the rendezvous point, the client and the entry router. The hidden server chooses an introduction point that introduces the service provided by the server to the Tor network. The Rendezvous point stays in the middle between the client on the Tor network and the hidden server which is using the introduction point. To make a connection between the two circuits (the one between the client and the rendezvous point and the one between the hidden server to the rendezvous point through the introduction point) a central server was used to receive the timing report from the client, the entry routers and the introduction point. It collects the “destroy cell” information from both the entry router and the rendezvous point (cell type, the timestamp of the cell, circuit ID and the source IP address) after the detection of the encryption error. Changing one bit in the cell by the rendezvous point makes this error reported to the central server.

The important part for detecting the hidden server is to let it choose a specified entry router which depends on the number of entry routers and the bandwidth available. To ensure that tracking the “destroy cell” through the Tor network will lead to detecting the hidden server, the configured hidden server was tested with ten entry routers and repeated 1,000 times. The test result was a 100% detection of the hidden server with a 0% False Positive.

The security of hidden services on Tor was analyzed by Biryukov *et al.* [10] to find out the amount of information that could be obtained about hidden services. The analysis included finding the popularity of any hidden service, denying access to the hidden service by impersonating the responsible hidden service directory and revealing the IP addresses of the hidden services. Two main techniques were used in this analysis. The first was inserting nodes with an incorrectly announced high bandwidth. The second technique used Sybil attacks to inject nodes to be selected as hidden service directories (which hold the service descriptor ID). The Tor network uses the bandwidth announced by the relays which gives the relays the chance to set unreal bandwidth values. Even though the Tor network uses a bandwidth measurement technique by establishing a circuit and downloading a file through the node, this will not prevent announcing an incorrect high bandwidth. When a node announces a high bandwidth it increases the probability of being selected in the connection path, the hidden server directory and the introduction point.

It is necessary to know the descriptor ID of the hidden service and the fingerprint of the hidden service directory to find out which one is responsible for the hidden service. From the original directory anyone can get the hidden service descriptor ID. When the hidden service starts to publish its descriptor ID it compares all the hidden service directories with their fingerprints. The descriptor ID number must be less than the fingerprint of the hidden service directory responsible for publishing its ID. Using this information, anyone can use the public key of the hidden service and manipulate the hash function to generate a fingerprint which is greater than the hidden server directory responsible for the hidden server. In this way the hidden server will choose this node as that responsible of its descriptor ID. Therefore, in their analysis, they used this technique to control the hidden service directory of the Tor botnet, the Silk Road hidden service and the DuckDuckgo hidden service. After controlling these directories for several days, they were able to get the number of requests for the hidden service descriptor. Each hidden service publishes its descriptor in three hidden service directories. Tor arranges hidden service directories in a closed ring. Each descriptor ID belongs to the three higher directories. If there are 1,200 hidden services then, in order to collect the directories of all of them, it requires 600 nodes using the technique described above. Instead, by using the shadowing technique this number could be decreased to 24. Tor has a rule which limits the maximum number of nodes to two, for any IP address to run a node as a directory on the Tor network. To overcome this, they used 50 IP addresses where each address runs 24 Tor nodes (from EC2). Tor will pick only 100 of them. The rest (1,100) will be ready to replace any unreachable nodes. Making these 100 nodes unreachable intentionally will let Tor use the other nodes (shadow nodes) as new directories holding the same previous information. This way all hidden server directories can be retrieved.

Elices *et al.* [32] used a time fingerprint to mark a hidden server behind Tor. It showed that even for web servers that use Tor to hide their identity, using a time fingerprint could lead to marking the server. They estimated the time required for the HTTP request to get a response from the web server on the Tor network by using the data field in the HTTP response message with statistical models. In this case, their sent requests were logged on the server with many other users' requests. There was a delay in the server response time due to the latency on the Tor network. To calculate the time for the server response they did two different experiments showing that the delay through the Tor network could be estimated depending on the server characteristics. The data used for these experiments was collected from seven servers around the world (research and university servers) with more than ten million requests. This data was analyzed to find the best distribution (Binomial or Poisson) that modelled the arrival of the server request within a specific time window. This time window was then used in the calculation to distinguish server responses from different users.

3.5 Packet Inspection

Wilde [112] found out that censorship in China uses a scanner to discover where the bridge is. To do so, a connection from the user to the bridge must take place first. Should inspecting the packets show that that there is a connection using the Tor protocol, the scanner starts to search through random Chinese IP addresses. If a certain IP address seems to be a bridge, then it establishes a connection to this IP address. If the connection is a success, then this is a bridge IP and its ports are blocked. Tor connections could be recognized because Tor uses a special Tor TLS client hello message.

The result of Wilde's work was used by Winter & Lindskog [113] to get more details about how China blocks the Tor networks. They did an experiment to understand how clients in China are blocked from accessing the Tor networks. In this experiment, they set up a relay in Russia and bridges in Singapore and Sweden to help the clients in China to be able to connect to the Tor network. They found that once the bridge IP address was discovered by the Chinese authorities, it was blocked within 15 minutes with the port (IP:Port tuple). This way the blocking of the Tor bridges did not affect other Internet traffic. In this case, the blocking was continuous as long as the scanners could connect to the bridge, otherwise the blocking was removed. They also found that only 1.6% of Tor relays could be reached from China. The Packet inspection for Tor connections is applied only for the connection from China to the outside world. The scanning of the bridge in Singapore comes from one IP address. After inspecting this IP address and its behaviour to Ping, Transmission Control Protocol (TCP), User Datagram Protocol (UDP), the Internet Control Message Protocol (ICMP) traceroute and Time to live (TTL), it seems as though it is a spoofed IP address. The scanning of bridges which have been discovered already is repeated every 15 minutes. Inspecting packets will not lead to the discovery of Tor connections in cases using pluggable transports such as "Obfsproxy" that Tor uses in its browser.

3.6 First N-Packets for Traffic Classification

There are many studies that use the first N-Packets of the flows for traffic classification to design an early detection system or online traffic classification. Huyn-Min An *et al.* [4] proposed a statistic signature method to classify application traffic. The statistic signature is generated from the payload size, the direction of packets and the order of packets in a flow. The proposed method aims to find a signature for each application from these features in the first N packets in the flow. In their experiment to classify Dropbox, uTorrent, Nateon, Skype and Kartrider, the achieved precision was between 97-100% and the recall was 25-78%.

The statistical features of the first N-packets is also used by Tabatabaei *et al.* [93] for traffic classification and early detection system. SVM and k-Nearest-Neighbors are employed to classify seven applications: BitTorrent, Gnutella, P2P live-streaming, Skype, Web browsing, Email and FTP. The experiment tested the first 3, 5, 7, 9, 11, 13 and 15 packets of the application flows. The results showed that SVM achieved the best overall accuracy of 84.5% when the number of packets is seven.

Huijun *et al.* [45] tested using the packet length, packet intervals and packet direction of the first N-packets in a simulated environment to early detect and classify traffic. Multiple experimental tests were employed by using a combination of features from the three features (packet length, interval and direction), four machine learning classifiers (C4.5, KNN, KNN1 and SVM) and a variable number of first N-packets of the flow (2 - 10). The results showed that KNN has the highest accuracy when the number of packets is 6.

Gu *et al.* [38] proposed a Bayesian Networks online traffic classification system by using packet sizes and the inter-arrival times of the first N-packets of a flow. The proposed system was tested on collected data for five applications: HTTP, POP3, POP3SSL, SMTP and FTP when using the first seven packets in the flows and the two aforementioned features. The results showed an accuracy between 88-100%. When the number of the first packets is between five and seven, the variance of the accuracy is small.

Bernaille and Teixeira [8] proposed an early recognition system for encrypted application classification by using the size of the first few packets of an SSL connection. The proposed system identifies the encrypted traffic in two stages. Firstly, the SSL traffic is identified, then the early detection is applied to recognize the type of the encrypted traffic. The accuracy of the proposed system is 85%.

3.7 Summary

As shown in the previous sections, research on multilayer-encryption anonymity networks has studied multiple aspects such as design, attacks, development, performance, identification and many others. This thesis focusses on contributing to some of the aspects covered as well as contributing to some areas that have not been investigated.

Much research on the multilayer-encryption anonymity networks has studied measuring the anonymity level. Even though the above studies have been important and significant in measuring anonymity levels, this measurement could be analyzed from a perspective other than that of linking the message to the sender among the anonymized users, since other factors affect the anonymity level, such as user behaviour and the browser setting being used. Even the link between the user and the final destination varies in its theoretical ability to achieve based on the design of the anonymity service itself, which is different from one system to another. Therefore, this research will present a method for measuring the level of anonymity by analyzing the anonymity service from different perspectives and proposing measurable metrics (factors) which enables the quantification of the anonymity of such services.

Research on the identification of multilayer-encryption anonymity networks by discovering the network's infrastructure requires compromising part of the anonymity network. Then the compromised resources are used to identify users on the network or other resources on the anonymity network. Other research has been proposed employing resources out of the anonymity network (such as web servers) and using them to provide access to the anonymity networks' users. Then the information from these resources is used to identify users or profile them on the condition that the user has accessed these resources. On the other hand, this thesis employs flow analysis to identify a multilayer-encryption anonymity network without compromising any part of the anonymity network. The analysis also preserves the users' privacy and does not aim to de-anonymize the anonymity networks' users. Moreover, the analysis has been extended to investigate the possibility of identifying those applications running on the multilayer-encryption anonymity networks without compromising any network's resources.

In addition, this thesis uses flow analysis to identify the obfuscated traffic employed by some of the multilayer-encryption anonymity networks which is an area which lacks or has very weak research which covers this issue. Moreover, features that best suitable for identifying the multilayer-encryption anonymity networks for better performance are presented in this thesis.

The first N-packets were employed in many studies for designing an early detection system or online traffic classification. Most of these studies focussed on application classification, some of them used the first N-packets for encrypted traffic classification. The features employed in such studies are different based on the type of applications under study. Packet size, inter-arrival time, packet direction and duration are mostly the features which accompanied the first N-packet studies with different machine learning algorithms. In this thesis, the first-N packet is used to design a system that could classify multilayer-encryption anonymity networks.The statistical information from these N-packets is used to generate new features which fit such a classification task.

In summary, the proposed approaches in this thesis for identifying multilayer-encryption anonymity networks do not need to compromise or use any resources within the multilayer-encryption anonymity networks. In some of the research mentioned above the identification of multilayer-encryption anonymity networks requires the use of marking technique for the users' traffic within the network or compromising/operating resources within the network (such as nodes) or resources at the final destination (such as web servers). This could interfere easily with the users' privacy. In addition, these approaches are limited to traffic passing through the compromised resources.

Chapter 4

Weighted Factors for Measuring Anonymity Services

There are many tools, applications and websites on the Internet claiming to protect the privacy of their users. The level of provided privacy protection for these services is different based on the way they work. For example, a VPN (Virtual Private Network), which can be provided either as a free or a paid service, hides the user's identity while surfing the Internet anonymously. At the same time, the service provider has access to the user's identity and his/her activity on the Internet. Some of these service providers also keep the logs of their users. This is also the case with free proxy websites, which claim that they protect the user's identity.

In fact, a user's privacy in such services depends on the amount of trust the user places in these service providers. On the other hand, there are other systems that claim to provide anonymity service without logging user activity by relaying the user connection to the final destination (such as a web server) via multiple stations. The design of such systems aims to prevent the stations from linking between the user request and the final destination.

Tor, JonDonym and I2P are popular anonymity services. They provide anonymity to their users to hide their identity from Internet web servers and hide the websites they have accessed. These systems prevent not only the web servers from revealing users' identities, but also the operators of the systems themselves from identifying the users. However, there are many details behind this kind of anonymity which might not be clear or obvious to the user. For example, changing the default setting in some of these systems' browsers (such as JavaScript or Cookies) could lead to a breach of user anonymity. These systems provide anonymity and at the same time give the user the ability to customize the settings of the system to control the level of anonymity. For example, JavaScript could be enabled or disabled as a default in these systems depending on which system is used. Many websites require that JavaScript be active to show website contents properly; however, the user has the ability to enable or disable JavaScript, which might conflict with how these anonymity services work. Tor and JonDonym have their own browsers, which are modified to ensure the users' privacy. However, the user can use any other available browser of their choice and thus have the ability to configure it to work with these anonymity services. In this case, it is the user's responsibility to ensure a proper configuration that guarantees anonymity. Even with the proper configuration or using the default browser, user privacy and anonymity concern not only the setting, but also user behaviour and the tools employed.

Browser fingerprinting is one of the examples of how the anonymity of the user could be breached. The browser itself could provide considerable information about the user's environment and consequently their identity. This type of information is obtained from the HTTP that is used for communication between the web browser and the web server. The HTTP header of this protocol contains information about the browser name, version, operating system, language and other information. For example, enabling cookies could lead to the storage of third-party cookies, which then could provide the ability to track users by the web sites they visited. Browser fingerprinting is not limited only to this; there are many studies on the application of different methods of implementing browser fingerprinting [68] [69] [76].

Eckersley [29] collected a sample of 470,161 browsers that visited the website http://panopticlick.eff.org. A fingerprinting algorithm was then applied to the sample based on the information available in the HTTP request field (stored in the web server access log files) and the JavaScript was running in the browser to test the ability to define how unique the browsers were. The results showed that browser fingerprinting was possible with a promising performance, especially when the browser supported Flash or Java utilities.

Therefore, the anonymity level of the users is not the same, even when using an anonymizing tool. The reason/goal behind using an anonymity service varies from one user to another. This variation of goals could affect the anonymity level and the choice of the right anonymity service to achieve this goal. The design of the anonymity tools varies based on: (i) which services such a tool offers to users, (ii) how the user decides or measures anonymity level, given all the different anonymity services. To answer these questions, this chapter presents a method of calculating and comparing user anonymity levels that takes into consideration the different needs of different users. Therefore, this aims to assist the user in choosing the most suitable anonymity service for their needs. The proposed method depends on evaluating anonymity systems based on five factors. To measure anonymity level using this method, the factors are converted to numeric values in order to assign weights and scores. In addition, each factor is compared with the others according to the goal or purpose of anonymity. Therefore, the relative weights (importance) of the factors are determined based on who is using the anonymity service and why. In doing so, the objective is to provide a comprehensive measurement technique which could be used to evaluate the level of anonymity based on the environment in which the anonymity service is used.

4.1 Proposed Factors

Multilayer-encryption anonymity networks differ in the design and the main goal they aim to achieve. For example, the I2P network is designed as a private network. Consequently, the best performance and anonymity it could achieve is when it uses internally available resources. On the other hand, the Tor network performs better when browsing Internet web sites even it supports Hidden Services. This kind of differences between the multilayer-encryption anonymity networks has their effect on the level of anonymity. Therefore, the proposed factors aim to include and take into consideration what could affect the anonymity of such networks. The following section presents the five proposed factors to analyze the anonymity level of the aforementioned anonymity systems (Tor, JonDonym and I2P).

4.1.1 The Level of Information Available for the Service Provider

When a Tor user connects to the Tor network a virtual circuit is created. The circuit consists of three nodes; the first node has the actual IP address of the user and therefore his identity, but it does not have knowledge of his Internet activity. Tor uses the concept of Entry Guard, which means assigning the Tor user to a specific node which acts as a permanent gate to the Tor Network for this user. The goal of this process is to increase the privacy of the users in case of a compromised node. By using this process, the probability of using such a compromised node is low. On the other hand, using Entry Guard provides other information about the users to the operator of the entry node. For example, Internet browsing habits such as time of the day, online duration and the amount of data transferred could be obtained by the operator. This information could be used to perform attacks that depend on the correlation between the duration, data and the server. The exit nodes, through which all user requests pass, have a considerable amount of information, as they are the links between the Internet and the Tor users. The operator of the exit node has the ability to know and statistically evaluate the user's activities on the Tor network. For example, McCoy *et al.* [66] provide percentages of the Tor Internet activities based on exit node observations. Another important fact about the exit node that might not be clear for nontechnical users is that the encryption of the requests through the exit node are all based on the encryption of the original requests and has nothing to do with the three levels of encryption on the Tor Network. Therefore, the exit node alone can breach the anonymity of the users if they use their login information to access their email or any web server without sending an encrypted request.

Furthermore, JonDonym works similarly to Tor in terms of connecting the user with the requested destination without revealing user information to the destination. The first point on the JonDonym network (First Mix) receives the connection request from the user which has the information about the connection duration and the user's identity. The last point (Last Mix) does not know the user's identity, but it has the activities or the websites that the users request. Even though the user data passes through several mixes, the operators of these mixes do not have the ability to access the data. The encryption layers used by JonDonym and Tor overlay networks protect the data, even from the operators; an exception occurs when the data sent by the user to the web server is not encrypted when the last node/mix has the ability to access the data sent by the user. The anonymity mechanism in Tor/JonDonym depends on relaying the user data through multiple points (Node/Mix). Each node/mix only knows part of the connection information, not the whole information required to connect the user to the request for the web server; the assumption is that even a compromised mix/node will not be able to find the complete connection information.

On the other hand, what if all the nodes/mixes on the path between the user and the server are compromised or attacked? On the Tor network, the three nodes in the circuit path are selected by using the path selection protocol [24], which specifies the three nodes the user will use to relay the data in conjunction with the policy that the exit node operator defines. When the user's request does not match the exit node policy, the path selection protocol finds another exit node that permits such traffic. For example, an exit node might allow only port 80. In addition, the user has the ability to override the path selection protocol and to choose a specific exit/entry node. In this case, the chosen exit node will not change for any circuit created by the user. This flexibility and randomness in node selection make it harder for an attacker to target a specific user by trying to compromise the three nodes that the user selects. On the other hand, it might be possible to compromise a node on the Tor network. Potentially, volunteering to run a node on the Tor network does not require information about the operator beyond the IP address and the nickname. However, running and compromising three nodes does not mean that these three nodes will be selected by the path selection protocol.

On the JonDonym network, this type of attack is also possible; the difference lies in the operation of the mixes. The number of mixes on the JonDonym is fewer than the number of nodes on the Tor network. On the other hand, the operators of the mixes are registered with their identities. They also sign an agreement with JonDonym not to exchange information between operators of the mixes and not to save user data. One of the differences between Tor and JonDonym is that JonDonym mixes do not change and the path is always the same. In the case of cooperation between all the mixes, it is possible to breach user anonymity on the JonDonym network.

Last but not least, the goal of the I2P network is different from both Tor and JonDonym. I2P is designed to provide anonymity for the users within the I2P network; however, that does not mean that I2P services are limited by the network boundaries. Browsing web pages outside the I2P network requires configuring the user's machine to use an outproxy. In this case, the information available to the outproxy is similar to Tor's exit router or JonDonym's last mix. The outproxy has access to all traffic passing through; if the traffic is not encrypted, the outproxy can see sensitive information.

The common point among the three anonymity services is that at any point during use part of the network has some user information. This information could be the IP address of the user which is available at the first point on the anonymity network that connects the user to the network. Alternately, it could be the amount of traffic that the last point can see when sending traffic to its final destination. As a result, the difference in the design of the anonymity services regarding how to relay the traffic affects the difficulty of linking the user with their traffic.

4.1.2 Blocking Anonymity and Obfuscation Options

The anonymity systems can hide user activity on the Internet, but cannot always hide the fact that such a system is in use. Sometimes, using anonymity systems might raise questions about why such a system is in use. In some countries, the IP addresses of the hosts running such systems are blocked in order to prevent access to such networks.

On the Tor network, a bridge [97] is a special node (hostrouter) connecting the user with the Tor network. The IP address of the bridge is not announced like the Tor nodes. The user sends an email to the Tor network (bridges@torproject.org) to get an IP address for a bridge. The user can also use the bridge database website to get the IP addresses (https://bridges.torproject.org/). The Tor network provides the user with the IP addresses of three bridges during a 24-hour period. This is to prevent censorship organizations from obtaining and blocking all IP addresses.

Furthermore, Pluggable Transports (PT) [102] work as an interface between the Tor user and the Tor network. The user connects to a pluggable transport which sends the connection request to the Tor network on behalf of the user in order to hide the connection between the user and the Tor network. There is more than one pluggable transport tool available for the Tor users to choose. These tools work differently, using different techniques to resist different blocking methods.

In addition to blocking Tor by blocking the IP addresses of the nodes [101], there are cases in which the Tor service is blocked by other techniques. The encryption in the Tor network is based on using TLS between the communication parties; the user to the first node, the first node to the second node and so on. Therefore, fingerprinting the Tor TLS is one of the blocking techniques. Another blocking technique is Deep Packet Inspection (DPI). DPI is used to find a pattern that recognizes Tor. Toward this end, the handshake phase in establishing a TLS Tor connection could be used to identify Tor. Therefore, changing the content to look like something other than a TLS is used by some of the pluggable transports to hide their connections to Tor. In fact, Obfs3 [100] is one of the pluggable transports that obfuscates the Tor TLS to look like random strings using another layer of encryption to wrap the TLS handshake used by Tor. Even though Obfs3 aims to hide the TLS from an observer, the packet length and the timing of the packets are still the same as normal Tor connections because Obfs3 mainly focusses on preventing the Tor TLS from being fingerprinted.

It is possible to intercept a TLS handshake to extract the destination IP address. In the case of Tor, this IP address is the bridge or the node IP address. After obtaining the IP address, the censors can establish a Tor connection to this IP address. When a reply is received, it confirms that this IP address does belong to the Tor network. This active probing method is also used to find bridges and to block them [112] [113]. Scramblesuit [114] is one of the Tor pluggable transports and is designed to prevent such active probing. To resist against active probing a password and a ticket are used to connect to the Scramblesuit server. In the Tor network, the Scramblesuit password is exchanged by requesting the password from the bridge database (email/website).

Blocking the IP address of a bridge prevents Tor users from connecting to this bridge. Even though the IP addresses of the bridges are not announced, they can still be discovered [61]. Flashproxy [34] is another pluggable transport that works around IP blocking by using the IP addresses of the visitors to a website, which change based on the IP address of the website that supports Flashproxy. When a website chooses to provide Flashproxy to the Tor users, it includes a JavaScript code that is activated when visitors access the websites. The code uses the website visitors' browsers to pass the connection between the Tor user and the Tor relay. Therefore, the Flashproxy IP address always changes based on the IP addresses of the visitors to the Flashproxy-supported websites. Accordingly, blocking those websites which support Flashproxy does not affect the ability of the Flashproxy to connect the Tor user to the Tor relay. Once the visitors leave the websites, their IP addresses are not used anymore. However, blocking their IP addresses does not prevent the Tor users from connecting to the Tor relays because new website visitors will simply take over the connection task. This makes blocking IP addresses challenging and less efficient. On the other hand, Flashproxy by itself does not work on changing the form or pattern of the connection to Tor; rather, it depends on the Obfsproxy framework to accomplish this.

The user needs to install the Flashproxy client transport plugin (included in the Tor browser bundle) and define a specific port in the configuration to receive Flashproxy connections. When the user starts Tor, the client plugin sends an encrypted message to a facilitator containing the IP address of the user. The facilitator keeps track of all users who need to communicate with the Flashproxies and sends their IP addresses to the Flashproxies. The client communicates with the facilitator indirectly by connecting to Gmail and sending a message. The facilitator then obtains the users' IP addresses from the server. In this way, blocking the facilitator does not prevent the user from contacting the facilitator. In order to prevent the user from communicating with the facilitator, services such as Gmail need to be blocked, which is a challenge to the censors.

Flashproxy is distributed through the websites of the volunteers; they install and activate Flashproxy when they get visitors to their websites. When Flashproxy is activated on a volunteer's browser, it communicates with the facilitator which in turn provides the volunteer's browser with the IP address of the Tor user. The volunteer then sends a connection to the Tor user via the browser to the port through which the Tor user is configured to receive the Flashproxy connection. Also, the volunteer's browser sends a connection to the Tor relay and starts to transfer the data between the Tor user and the Tor relay. Again, blocking the websites of the volunteers that host Flashproxy will not prevent the Tor user from communicating with the Tor relay. This is because Flashproxy runs on the volunteers' browsers and the IP addresses of the Flashproxies are their own IP addresses.

Whitelisting is another method that can be used to block Tor. In this case, all the allowed traffic is profiled and anything which does not match with this list is blocked. Format-Transforming Encryption (FTE) [28] is a pluggable transport that takes a ciphertext and transforms it into another format that matches a regex. In the Tor case, FTE changes the Tor traffic to look like HTTP traffic, generating an HTTP regex out of Tor traffic that matches what DPI expects from HTTP traffic. Meek [99] is a pluggable transport which uses the concept of domain fronting which hides a Tor message inside an HTTPS request. Meek uses GoogleAmazonAzure for domain fronting to send Tor messages on behalf of a Tor user.

Last but not least, network traffic flow analysis is another technique which can be used to detect Tor. To evade the network flow analysis, Scramblesuit forms the traffic in a way which does not resemble a specific shape (form). This includes the packet length and the interarrival time for every Scramblesuit server. For example, it changes the packet length distribution to mislead classifiers. This way each server has its own flow characteristics. To this end, the server starts by generating a 256-bit seed randomly which is used in PRNG to generate two random distributions. The packet length is then changed by using a padding (0-1520 bytes) of random sampling from the distribution of all the packet lengths.

JonDonym has two options for bypassing blocking of the service. The first one is using a TCP/IP forward method in which the user will use an encrypted connection to another user who has unblocked access to the JonDonym network. Speed and stability suffer when using such a method. The connection also depends on the forwarder to stay alive. The second method uses Skype to tunnel the blockage of the JonDonym service which is more reliable than using the TCP/IP forward method.

On the I2P network there are no obfuscation options similar to Tor pluggable transports and it is thus possible for an observer to collect the routers' IP addresses with a harvesting attack [41]. Currently, the I2P network has not developed an obfuscation option which could provide the users with a way to connect to the I2P network if that network is blocked by using such an attack. However, the I2P network implemented other improvements in the design of the transport layer to obscure the identification of the I2P network traffic. I2P employed random port numbers, point-to-point encryption, DH key exchange and the use of both TCP and UDP. In addition, several obfuscation options are still considered by the developers of the I2P network, including using padding techniques at the transport layer to achieve random length, studying the signature of the packet size distribution and studying the technique used to block Tor.

It should be noted that anonymity services do not generally hide the fact that the users are connecting to the service. Consequently, in a regular situation in which the user is connected directly to the anonymity service, any observer can see that an anonymity service is in use.

In cases using an obfuscation option, an observer who wants to de-anonymize the user needs to determine that the user is connecting to the anonymity service in the first place. Therefore, the existence of obfuscation options is a factor which should be taken into consideration when measuring the level of anonymity.

4.1.3 Application and Anonymity

The common way to use an anonymity service is to use the default browser of the aforementioned services (Tor, JonDo, I2P, etc.) to browse the web. However, these anonymity services can be used with other applications in addition to web browsing. This requires the user to configure the application and the anonymity service to work together. For example, JonDonym enables the user's e-mail service to work with JonDonym and also supports any application that has the ability to configure the proxy. Tor supports any application that has the ability to pass all its traffic through a proxy. However, using any application other than the default browser on the Internet raises the chances of breaching the user's anonymity.

The configuration for these applications is not that simple for non-technical users. When configuring any application to work with an anonymity service it is important to understand fully how this application works to ensure that user information is not leaked. For example, the DNS request that accompanies many applications might leak the user's data and this, in turn, might breach the user's anonymity. Applications might not use the anonymity service to resolve the DNS name, even if they are configured to do so [103].

The user can run any application on the I2P network which depends on TCP or UDP; the I2P messages are based on UDP. TCP applications count on using an I2PTunnel which passes the TCP stream within the I2P network. For example, Eepsites [30] and IRC (Internet Relay Chat) use I2PTunnel [107] to work within the I2P's UDP-based network. Eepsites are websites hosted anonymously on the I2P network. The user accesses these websites without getting any information about the one(s) creating the website(s). At the same time, the website cannot determine the real identity of the users. These types of applications work by default only within the I2P network. To browse outside of the I2P network, an outproxy is needed to pass/forward the traffic. The I2P network supports many applications for blogging, file storage, DNS, email, file sharing, web hosting and others. These applications differ when working within the I2P network or outside the I2P network. Some of these applications are supported by third parties; therefore, the anonymity and security level varies among them.

The configuration of the application and how the user sets the application on the anonymity network is an important issue. For example, the web browser contains many details other than what anonymity system the user is using. The anonymity tools aim to make their browsers undistinguishable in order to raise the anonymity level. The Tor browser is a modified version of Firefox based on Mozilla's Extended Support Release (ESR) Firefox branch [98]. It includes HTTPS-Everywhere [44], NoScript [75], modifying some of the default Firefox settings and modifying some of the default extension settings. JonDoFox is the JonDonym browser and is a modified version of Firefox [54].

Even when using the default browser for anonymity services, the right setting of the browser is important to ensure the safety of the user against many Internet websites that track their visitors. To this end, some of the tools used by web sites could also identify the user or their behaviour for the purpose of advertisements, collecting data for different types of studies or building a database about the visitors of the website. Thus, it is crucial to know the policy and the default setting for a browser with such tools. The question to consider here is: how such tools address the trade-off between browsing the websites with full offered services and preserving the anonymity of the users [52].

Table 4.1 presents the information as to how the Tor Browser and JonDoFox, i.e. the JonDonym browser, deal with these trade-offs. Compared to Tor and JonDonym, the I2P network does not have a specific browser preference. After connecting to the I2P network, the user configures the proxy setting manually in any browser to use the I2P network. The network encrypts the traffic between the users within the network, regardless of the application used. The I2P network is designed to work as a private network on the Internet. The browser could be used to configure the router of the user. For example, configuring the bandwidth up and down, participation on the FloodFill, starting or stopping services such as IRC and web hosting are all possible on the I2P network.

	JonDoFox	**Tor Browser**
Cookies	Disabled	Enabled
Third-party Cookies	Disabled	Disabled
JavaScript	Disabled	Enabled
Flash-Cookies (LCO)	Disabled	Disabled
WebGL	Enabled	Disabled
Flash Plug-in	Disabled	Enabled
Java	Disabled	Enabled
Silverlight	Disabled	Enabled

Table 4.1: Default browser settings for anonymity services.

The applications supported by the anonymity services are not the same. The method used to run applications other than web browsing also varies from one anonymity service to another. How well the anonymity service is structured to support a number of applications affects the level of anonymity. For example, using the default anonymity browser or configuring the user's browser could make a difference in the anonymity level. Therefore, it is not only the anonymity service which affects the anonymity level; it is also what application is used along with that anonymity service.

4.1.4 Authority and Logs

There is no doubt that the policy of the anonymity services about co-operating with the authority (operator or regulator) and the keeping logs affects the level of privacy. For example, JonDonym's agreement with the operators requires keeping no logs and not exchanging information between operators of the mixes. The reason behind this policy is that the identities of the operators are known and they work according to the regulations in their own countries. Therefore, in JonDonym, there are several points that must be taken into consideration when evaluating the anonymity of such a system, as listed below.

- The mixes that construct the path are fixed. That means knowing that the user employs one of these mixes (e.g., the last mix) implies knowing the first and the second mix.

- The number of mixes on the JonDonym network is very limited compared to the Tor network. On the JonDonym network there are only nine Cascades; six are operated by companies and three by individuals.

- The operators of these mixes are known and registered. They work according to the regulations in their countries regarding cooperation with the authorities.
- These cascades are fixed. This makes it easier for authorities to approach and investigate them.

On the other hand, on the Tor network there is a different approach, as indicated below.

- The nodes that construct the user's path are not fixed. The user connects to three nodes that change periodically. Therefore, knowing that the user connects to a specific exit node does not necessarily imply knowing the first or the middle node.
- The number of Tor's nodes is around 8,000, which makes it relatively harder to get information about them.
- The operators of these nodes are not known. Tor does not require their users to identify themselves when offering to run a node. This might help to protect the operators' identities, but it does not guarantee that the operators are to be trusted.
- The nodes on the Tor network are supposed to be online as much as possible. However, that is no guarantee since most of these nodes are run by volunteers.
- On the other hand, keeping the log for the created circuits is an available option for the nodes' operators. When the debug option is enabled in the configuration file, then the log file will contain information about the circuits and cells. Also, the operator of the node has the ability to modify the Tor source code in order to log additional information about the cell. It can be used to extract information and analyze it later (as shown in Chapter 7). Getting this extra information does not mean that these tools do not provide anonymity; it indicates that there is a specific amount of information available to the operators of the nodes of which the user should be aware.

Furthermore, on the I2P network there other options as indicated below.

- The I2P user has the option of modifying the number of routers used when exchanging messages. In addition, end-to-end encryption is used. The concept of garlic routing is used as well when exchanging messages. In this way, messages that pass through the routers are not distinctive, which means the purpose or the content of the messages cannot be extracted or inferred easily. For example, information as to whether the messages are from an extension to the number of routers in the tunnel or whether they contain data would not be extracted from the messages.
- The I2P network is decentralized, so there is no single point that is responsible for or represents the network.
- The user does not need to know all the routers in the network to be able to use that network's resources.
- The I2P network's design is different from Tor and JonDonym; it is designed basically to provide a private network within the Internet. The number of outproxies is very limited. Also, this makes the browsing outside the network low compared to Tor and JonDonym. Therefore, the possibility that the user will use the same exit point frequently is high. This does not mean that it is a threat but it does increase the probability of correlating the user with their traffic based on factors such as access time, duration and the amount of data used.

Based on the above, it can be seen that the lower the possibility of compromising all of the nodes on the user's path, the better the anonymity level. In addition, what information the service provider (operator) has about the user and the operator's willingness to provide this information when asked to do so is important as well in measuring the level of anonymity.

4.1.5 Threat Models

Ideally, anonymity services provide anonymity to their users and protect their privacy. However, there are possible threats which could breach the anonymity of such services.

The anonymity services are based on the separation of the user's identity from the data sent or received by the user. One of the threats that such services face is correlating user data and final destination data. This is possible by monitoring the first point in the anonymity network and the last point that connects the user with the final destination (web server). Through analysis of the amount of data, it is possible to correlate the user and their final destination when there is variety in the data size. The path on the JonDonym network is known and if the attacker has the ability to monitor the traffic from the first mix and the last mix (out of the last mix), then connecting the users of this cascade and the amount of sent and received data is possible. The path on the Tor network is not fixed, but the correlation is also a possible threat. To this end, there are studies on using marking techniques to trace user activities, but they are often limited to a specific user, a specific web server or even a specific exit node. The attacker could compromise both an entry node and an exit node, in which case the traffic out of the entry node is marked. The attacker then watches for the mark to appear at the exit node. Indeed, the probability of the user who is using the compromised entry node of using the compromised exit node as well is very low, but it is still possible. The mark might be used as well to track the web server instead of the exit node [60]. In this case, the attacker compromises an entry node and watches for the users who are using this entry node to access this specific web server. On the other hand, the design of the I2P network makes this kind of correlation a low threat. The path is not fixed or specified; users build inbound and outbound tunnels which do not count on the type of the router. All routers on the networks can be part of any path. The encryption mechanism provides for the confidentiality and the integrity of the messages. However, if the attackers have the resources to monitor all routers, then they may have enough data to discover paths.

As for the JonDonym network, this type of attack can target a mix server. A mix server has a limit on the number of users it can serve. The attacker could use this limit to break the anonymity of the mix server. If the attacker connects to a mix server to fill its capacity (n) to the point (n-1) when the user connects to the only space left in the mix server, the attacker could then isolate and detect the user's traffic.

The threat models are not the same for all anonymity services; what is considered a threat to one service may not be applied to another anonymity service. Even when they share the same threat to a certain saturation point, the level of the risk is not always the same. Therefore, to measure the anonymity of any anonymity service, the threat model should be taken into consideration, based on the environment or the purpose for which the anonymity service is used.

Accordingly, evaluating the level of anonymity should be done in a comprehensive way that takes into consideration the purpose, the design and the environment. As a result, the following factors will be used to measure the effectiveness of any anonymity service: the level of information available to the service provider, the obfuscation options, application anonymity, the authority, the logs and the threat models.

4.2 Evaluation

This section discusses how the aforementioned factors can be used to measure the anonymity of Tor, JonDonym and I2P. These factors are dynamic, so they change over time based on many variables such as user behaviour, the anonymity system used, the configuration of the system, the purpose of using the anonymity system, etc. Therefore, these factors will be quantified in order to be able to measure and compare them against each other. This process is called the weighted anonymity factor. What follows summarizes the weighted anonymity factor measurement.

4.2.1 Factor Calculation

To quantify these factors, they are grouped into three categories, as shown in Table 4.2. These categories (High, Mid and Low) are converted into numerical values as 100, 67 and 33, respectively. The exception is the obfuscation, where it is labelled as Yes or No depending on whether an obfuscation is used. The reason is that some of the anonymity systems involve obfuscation techniques and others do not. Additionally, the use of these techniques (if they are available) is optional. Therefore, the value is set to 100 (for No) and 0 (for Yes). The higher the values for these factors, the lower the anonymity level of the system. For example, a 100 in the Threat model factor is applied whenever the threat in the case under study is very strong (i.e., highly probable). The three categories are represented by 100, 67 and 33 as an approximation for High, Mid and Low. These values could be expanded and detailed to a scale from 10 to 100 to increase accuracy. This will be explored further in Section 4.2.5.

Level of Information	**High**	**Mid**	**Low**
	100	67	33
Obfuscation	Yes	No	
	0	100	
Authority and Log	High	Mid	Low
	100	67	33
Application Configuration	Low Security Configuration	Mid Security Configuration	High Security Configuration
	100	67	33
Threat Model	Low Cost	Mid Cost	High Cost
	100	67	33

Table 4.2: Proposed anonymity factors.

4.2.2 Weight Calculation

Given that the weights of the factors may vary from one evaluation environment to another, quantifying these factors to measure anonymity is necessary but insufficient by itself. Also, the weights of the factors have to be considered. Therefore, the "Pairwise Comparison" technique is employed to evaluate the weight of these factors. Each one of the factors is compared with all other factors; then, the weight for the factor is calculated based on these comparisons. The higher the weight of a factor, the more important it becomes for the anonymity of a given service. Calculating the weights is performed until all factors are evaluated comparatively, as shown in Table 4.3.

γ_1				
γ_2	$2\gamma_1$			
γ_3	$\gamma_1 \gamma_3$	γ_3		
γ_4	γ_4	γ_4	$\gamma_3 \gamma_4$	
γ_5	γ_1	$\gamma_2 \gamma_5$	$\gamma_3 \gamma_5$	$\gamma_4 \gamma_5$

Table 4.3: Calculating the weights.

γ_1 refers to the first factor "level of information available to the service provider".

γ_2 refers to the second factor and so on.

Table 4.4 shows the weights of the five factors after the comparison and their total value. Based on the weight value it is important to notice the three factors with the same weight (the level of information, the application configuration and the authority and log. The obfuscation has the lowest importance, compared to the other factors. The weights represent the importance of each factor compared to the other factors. Using the pairwise comparison helps in deciding how to rank or weight the factors compared to the others.

γ_1	γ_2	γ_3	γ_4	γ_5	Total
4	1	4	4	3	16

Table 4.4: Final weights of the factors.

4.2.3 Weighted Anonymity Factor

Eq. 4.1 and Eq. 4.3 are applied after calculating the values of the factors and weights. Eq. 4.2 is the total of the weights of the factors. (f) represents the value of a factor.

$$Weighted\,Anonymity\,Factor\,(WF) = \gamma_1 f_1 + \gamma_2 f_2 + \gamma_3 f_3 + \gamma_4 f_4 + \gamma_5 f_5 \tag{4.1}$$

$$= \sum_{i=1}^{n} \gamma_i\, f_i \;\; where\, n = number\, of\, factors \tag{4.2}$$

$$Total\,Weight\,(T_\gamma) = \sum_{i=1}^{n} \gamma_i \tag{4.3}$$

The measurements may vary from one environment to another, where different factors are applied or when the numerical conversion is different from what is used in Table 4.2. To generalize measurements, Eq. 4.4 shows the conversion of the calculated values based on the factors used to a percentage by using the minimum and maximum values from Eq. 4.1.

$$Weighted\,Anonymity\,Factor\,(\%) \;=\; (1 - \frac{WF - Min(WF)}{Max(WF) - Min(WF)}) * 100 \tag{4.4}$$

Eq. 4.4 can be rewritten after calculating the weights to the form in Eq. 4.5.

$$Weighted\,Anonymity\,Factor\,in\,Percentage\,(\%) \;=\; (1 - \frac{WF - 495}{1600 - 495}) * 100 \tag{4.5}$$

4.2.4 Evaluation Case Study

In this scenario, three users are assumed for whom the levels of anonymity will be compared. It is important to note that the evaluation does not aim to identify the best anonymity service; it aims to evaluate the level of anonymity according to the environment in which these anonymity services are used.

The first user (A) uses the standalone Tor to browse Internet web sites. The user configures the Chrome browser to work with Tor by setting the browser to access Tor via Socket. To increase the anonymity level, the user adds Scramblesuit as an obfuscation option to his "torrc" file to access Tor via a bridge. The user (A) browses websites on the Internet which include a compromised web server by an attacker. The web server injects the coming request to force the browser to request images from another website that belongs to the attacker. The attacker aims to identify the user by forcing the browser to send requests without using the Tor network.

User (B) chooses to use JonDonym as an anonymity service. The user does not have a technical background. All the settings are left to default. The only addition to the default setting is that the user chooses to use the TCP/IP forwarder. User (B) wants all the activities that she performs on the Internet to be anonymous. Therefore, user (B) uses JonDoFox to browse all the Internet websites. She visits usually web sites such as news, videos, email, Internet shopping and her bank account.

User (C) lives in a country where the Internet is censored and some websites are blocked. Therefore, user (C) uses Tor to gain access to the blocked Internet blogs. User (C) browses these blogs and participates on them via Tor. This user is concerned about hiding his identity, so he uses the Internet from the company where he works. It seems that user (C) is the only person who is using Tor in this company. The user organizes his time so that he only accesses Tor at the end of the day between 5-6 pm on weekdays.

According to the scenario above, Table 4.5 shows how it is converted to measurable numeric values, using the proposed factors.

	Level of Information	**Obfuscation**	**Authority and Log**	**Application Configuration**	**Threat Model**
A	33	0	33	100	67
B	100	0	67	33	100
C	67	100	100	33	67

Table 4.5: Evaluated factors for users (A), (B) and (C).

Table 4.5 is calculated based on the given information about the scenario above and how the users (A), (B) and (C) are using these anonymity services. For example, user (C) did not include an obfuscation option when using the anonymity service; therefore, the obfuscation value is measured as 100. User (A) prefers to use his favorite browsers instead of using the default Tor browser. Therefore, the possibility of a DNS leak is higher, especially when accessing suspicious websites or when using any application other than browsing. Based on that, user (A) gets 100 on the application configuration. Even though user (B) uses some sort of obfuscation, she misses the fact that browsing any website already linked to her real identity such as her email or bank account, even while using an anonymity service, does not mean that she is anonymous. Furthermore, the information available to the exit node, in this case, is high, even if the information does not contain passwords. The level of information is evaluated as 100 in this case. The same applies to user (C); he uses Tor at the same time daily from the same place where no one else is using Tor. Using Table 4.5 and Eq. 4.1, the weighted factors will be calculated as below.

$$WF = \gamma_1 f_1 + \gamma_2 f_2 + \gamma_3 f_3 + \gamma_4 f_4 + \gamma_5 f_5$$

$$WF = 4f_1 + f_2 + 4f_3 + 4f_4 + 3f_5$$

$$WF_A = 4 * 33 + 0 + 4 * 33 + 4 * 100 + 3 * 67$$

$$= 865$$

$$WF_A(\%) = (1 - \frac{865 - 495}{1600 - 495})$$

$$= 66.5\%$$

$$WF_B = 4 * 100 + 0 + 4 * 67 + 4 * 33 + 3 * 100$$

$$= 1100$$

$$WF_B(\%) = (1 - \frac{1100 - 495}{1600 - 495})$$

$$= 45.2\%$$

$$WF_C = 4 * 67 + 100 + 4 * 100 + 4 * 33 + 3 * 67$$

$$= 1101$$

$$WF_C(\%) = (1 - \frac{1101 - 495}{1600 - 495})$$

$$= 45.16\%$$

Based on the above calculations, user (A) has a higher level of anonymity than either user (B) or (C).

4.2.5 Expanding the Quantification

The mechanism used in this chapter to measure anonymity level is first based on determining the factors. Then, each factor is divided into three levels (two for the obfuscation) in order to be able to evaluate each factor numerically. The importance of each factor is then determined based on a comparison of all the factors. Figure 4.1 shows the sequence for measuring anonymity. The second step, which is converting the factors into measurable values, has three levels: High, Mid and Low. According to these values, the factors are converted into numeric values. This could be considered to be the applicable form of measuring anonymity. However, it is possible to expand this step to improve the accuracy of the quantification of the factors by: (1) instead of using three levels, the factors could be evaluated as a scale (for example, from 10 to 100), (2) At the same time, each value on the scale should represent the level of the anonymity of the factor in a predefined way. In this way, the value of the factors is determined more accurately. For example, if the extended scale is applied to the "Threat Models" factor, then the values will be from 10 to 100 instead of 33, 67 and 100. The threats or attacks on the anonymity systems should be ordered to match the scale from 10 to 100. This requires the study and evaluation of all possible threats on the anonymity systems and their applicability. In this way, the scale has predefined values for every possible threat against the anonymity systems in the threat models factor. The same step could be repeated for the other factors. This type of approach could increase the accuracy of the proposed factors in measuring the anonymity provided by the services.

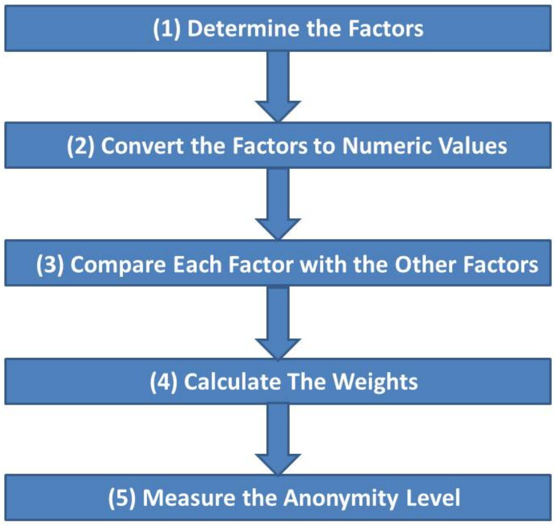

(1) Determine the Factors
(2) Convert the Factors to Numeric Values
(3) Compare Each Factor with the Other Factors
(4) Calculate The Weights
(5) Measure the Anonymity Level

Figure 4.1: Sequence for Anonymity Measurement

4.3 Summary

This chapter presents a different method for measuring the level of the anonymity on the multilayer-encryption anonymity networks. First, the method applies five factors which influence the anonymity on such networks. The five factors are: the level of information available for the service provider, blocking anonymity and obfuscation options, application and anonymity, authority and logs and threat models. Then these factors are compared with each others to rank their importance (weight). Afterward, depending on the case under study, these weighted factors could be applied for measuring the anonymity level. As an example for applying these factors, the anonymity level is measured for three cases with different environments for using the anonymity network which mimic some of the real use of such anonymity networks. In short, the measurement of the anonymity networks provides a quantifiable way of measuring the level of anonymity of multilayer-encryption anonymity networks.

Chapter 5

Anon17: Network Traffic Dataset of Anonymity Services

One of the difficulties that face researchers in the anonymity networks field is the lack of an anonymity data set. Researchers need to collect their data to conduct research or to use a simulated environment to collect the data they need. The lack of such a data set is due to the nature of the anonymity networks. These anonymity networks aim to provide a certain level of privacy for the users. There are many studies conducted on these anonymity tools. The studies include a wide range of aspects related to the anonymity field such as improving the design, performing attacks on the anonymity tool, analyzing the user's behaviour on the anonymity network, studying the performance and delay, revealing the user's identity and many others.

In some of the anonymity research papers the used data is collected in a simulated environment [6] [7]. Others used data collected by the researchers themselves [16] [3] [111] [62]. The most common issue that faces researchers in the anonymity field is that these anonymity tools provide anonymity to the users; thus collecting the data and making it publicly available might affect the privacy of the users of the anonymity tools. Consequently, the research in this field ends up using data collected from a simulated environment or collected by the researchers themselves.

The traffic on the anonymity networks relies on passing the users' data through multiple stations on the network (nodes, for example). These stations pass traffic for multiple users; collecting the data from these stations will include traffic for other users which means that usually the research needs to run a station (node) and modify the way they collect the data to include only their traffic.

Anon17 is a data set for three anonymity tools: Tor, JonDonym and I2P. This data is prepared and made publicly available without affecting the privacy of the users. To this end, the IP addresses of the users have been removed. The payload information is used only for statistical measurement and then is removed. This is because Anon17 aims to provide a publicly available anonymity data set which could be used to study the aforementioned anonymity tools. The data set includes several applications used on these anonymity tools as well as several obfuscation techniques that are used on some of these tools. Consequently, the data set could be used for multiple types of research.

5.1 Data Collection and Traffic Types

Anon17 [5] was collected at the Network Information Management and Security (NIMS) lab [74] between 2014-2017 in a real network environment. The data set is labelled based on the information available on the anonymity services themselves. For example, in the Tor network, the IP addresses of the Tor nodes are available. Therefore, whenever data related to a node on the Tor network is collected, the IP address is used to label this traffic as Tor. The same applies for all the labelling in the data; none of the application classification tools was used to label the data. The Anon17 data set contains the data listed below.

5.1.1 Tor Data

The Tor data set contains Tor traffic. The traffic includes the circuit establishment and the user activities on the Tor network, such as browsing the Internet websites.

5.1.2 TorApp

The TorApp data set contains flows for three machines (computers) running three applications on the Tor network (Browsing, Video streaming and File sharing). Consequently, there are three classes on the TorApp data set (Browsing, Streaming and BitTorrent [BT]). The Browsing class contains connection flows between a user and an entry node on the Tor network when the user is using Tor to browse different Internet websites. The Streaming class has the connection flows between the user and the entry node when the user is watching videos on Tor. The last class, the BitTorrent class, contains flows between the user and the first node when the user is using Torrent files on the Tor network.

5.1.3 Tor PT

The TorPT data set contains flows for Tor pluggable transports. TorPT has five classes: Obfs3, Meek, Flashproxy, FTE and Scramblesuit. TorPT is collected by connecting to the pluggable transports from the NIMS lab and capture the traffic. The Obfs3 data is collected from two different Obfs3 bridges. The Scramblesuit traffic is collected by connecting to 22 different Scramblesuit servers. This is to ensure that the effect of changing the flow behaviour that Scramblesuit pluggable transport aims to achieve is included in the data.

5.1.4 I2PApp80BW

These traffic flows are collected while running three applications on the I2P network. The applications (classes) are Eepsites (the web sites browsing on the I2P network), jIRCii (Internet Relay Chat [IRC] plugin on the I2P network) and I2Psnark (the file sharing plugin on the I2P network). The bandwidth sharing on the I2P client is set to default which is an 80% sharing rate of the user bandwidth. In this data set each class contains the application flows in addition to the management traffic flows. For example, the Eepsite flows contain flows for the Eepsite Tunnels in addition to the Tunnels used for the management of the I2P network and tunnels used to share bandwidth such as the Exploratory and the Participating Tunnels.

5.1.5 I2PApp0BW

This data set is similar to the I2Papp80BW; the difference is that the amount of shared bandwidth is set to 0%. This will reduce the amount of management traffic flows on each class.

5.1.6 I2PUsers

This data set contains the traffic flows for three users on the I2P network. The classes are named PC1,PC2 and PC3. The data set I2PUsers is the same data set used in I2PApp80BW. The difference is that the data is classified differently. Instead of labelling the data set based on the application, the data set is labelled based on the user traffic. Consequently, any class on this data set will contain the three applications used on the I2Papp80BW data set. For example, the PC1 flows contain flows for the machine of the first users when the users have used Eepsites, jIRCii and I2Psnark on the I2P network.

5.1.7 I2PApp

This data set contains traffic flows for the same three applications used in I2PApp80BW. The difference is that this data set contains separate classes for the management tunnels. The total number of classes in this data set is five: Eepsites, jIRCii, I2Psnark, Exploratory Tunnels and Participating Tunnels. Consequently, the application tunnels do not contain any management tunnel flows.

5.1.8 JonDonym

The JonDonym data set contains traffic flows for the JonDonym network. The data set contains flows for the whole free mixes on the JonDonym network. Table 5.1 shows the Anon17 data set and the number of instances on each part of the data set.

Tool	**Type of Traffic**	**Dataset Name**	**Classes**	**Number of Instances**	**Total Number of Instances**
Tor	Normal Tor Traffic	Tor	Tor	5,283	5,283
	Applications on Tor	TorApp	Browsing	84	252
			Streaming	84	
			Torrent	84	
	Tor Pluggable Transports	TorPT	Obfs3	14,718	353,384
			Meek	43,152	
			FTE	106,237	
			Scramblesuit	16,953	
			Flashproxy	172,324	
I2P	I2P Applications Tunnels with other Tunnels 80% Bandwidth	I2PApp80BW	Eepsites	149,997	449,987
			jIRCii	149,998	
			I2PSnark	149,992	
	I2P Applications Tunnels with other Tunnels 80% Bandwidth	I2PApp0BW	Eepsites	127,349	195,081
			jIRCii	29,357	
			I2PSnark	38,375	
	I2P Users Traffic	I2PUsers	Pc1	150,000	449,998
			Pc2	150,000	
			Pc3	149,998	
	I2P Applications	I2PApp	Eepsites	145	640
			jIRCii	221	
			I2PSnark	62	
			Exploratory Tunnels	86	
			ParticipatingTunnels	126	
JonDonym	JonDonym Traffic	JonDoNym	JonDonym	5,440	5,440

Table 5.1: The number of traffic flows in each data set.

5.2 Dataset Features and Format

Tranalyzer [106] is used to extract the flows from the PCAP files captured in the NIMS lab. Tranalyzer has 91 features such as Number of bytes sent, Number of bytes received, Statistics about the Interarrival time and Number of connections, etc. Some of the unrelated features are removed from the data set, such as the ICMP features and VLAN features because they do not provide useful information for this research. IP addresses and payloads of the packets are removed as well from the data set to protect the privacy of the users. In the data set the values of some features might have zeros. For example, the I2P network works on both TCP and UDP. Therefore, if the I2P data set contains UDP connections then all the TCP-related features will have zero values. The data is formatted into into the Attribute-Relation File Format (ARFF) file format used in the open source data mining software tool, WEKA [40]. Table 5.2 summarizes the features included in the Anon17 data set.

Tranalyzer Features	**Description**
dir time_first, time_last, duration	Flow direction, time and duration of the flow
numPktsSnt numPktsRcvd numBytesSnt numBytesRcvd minPktSz maxPktSz avePktSize pktps bytps pktAsm bytAsm	Counting of Packets and Bytes
ip_mindIPID ip_maxdIPID ip_minTTL ip_maxTTL ip_TTL_Chg ip_TOS ip_flags ip_Opt ip_OptCnt	The IP Header-related features such as TOS, TTL, etc.
tcp_PSeqCnt tcp_SeqSntBytes tcp_SeqFaultCnt tcp_PAckCnt tcp_FlwLssAckRcvdBytes tcp_AckFaultCnt tcp_InitWinSz tcp_AveWinSz tcp_MinWinSz tcp_MaxWinSz tcp_WinSzDwnCnt tcp_WinSzUpCnt tcp_WinSzChgDirCnt tcp_AggrFlags tcp_AggrAnomaly tcp_AggrOptions tcp_MSS tcp_WS tcp_OptCnt tcp_S-SA/SA-A_Trip tcp_S-SA-A/A-A_RTT tcp_RTTAckTripMin tcp_RTTAckTripMax tcp_RTTAckTripAve tcpStates	The TCP Header-related features such as Window size, sequence number, etc.
connSrc connDst connSrc_Dst	Counting of number of connections between source and destination/ source to different destinations.
min_pl max_pl mean_pl low_quartile_pl median_pl upp_quartile_pl iqd_pl mode_pl range_pl std_pl stdrob_pl skew_pl exc_pl	Packet length statistics
min_iat max_iat mean_iat low_quartile_iat median_iat upp_quartile_iat iqd_iat mode_iat range_iat std_iat stdrob_iat skew_iat exc_iat nfp_pl_iat ps_iat_histo	Inter arrival time statistics
TrafficType	The Classes

Table 5.2: Anon17 data set features.

5.3 Summary

This chapter presents Anon17, the first publicly available data set for the multilayer-encryption anonymity networks studied in this thesis. Anon17 contains data for three popular anonymity networks, namely, Tor, JonDonym and I2P. Moreover, Anon17 contains data for several applications running on multilayer-encryption anonymity networks such as browsing web pages and videos and file downloading, among others. These applications' data is collected at the NIMS lab by running multiple computers connected to the anonymity networks running different applications. As well, Anon17 contains different obfuscated traffic used on Tor such as Meek, FTE and ScrambleSuit, among others. Moreover, Anon17 is ready-to-use for researchers in the form of Arff files.

Chapter 6

Research Methodology

This research aims to analyze multilayer-encryption anonymity networks. Tor, JonDonym and I2P are well-known examples and the most popular multilayer-encryption anonymity networks. Therefore, these aforementioned anonymity systems/networks have been chosen to be studied in this research. As shown previously in Chapter 4, analysis of such networks starts by understanding the anonymity they aim to provide. Several factors are proposed which describe the anonymity on such networks. In addition, these factors are quantified (weighted) in order to be able to apply them to different anonymity environments.

6.1 Data Collections

The lack of a data set in this research field is a challenge to study multilayer-encryption anonymity networks efficiently. The analysis of such networks in this research requires obtaining various data sets to cover the multiple aspects the research will study. The required data is the traffic on Tor, JonDonym and I2P. The configuration and the setup for these networks are not the same, so collecting data for each one of them requires a different setup. In Chapter 5, Anon17 is presented where the anonymity data is collected and made publicly available. This research will use Anon17 and other data sets collected by different universities and labs.

Experiments on the analysis of traffic flow behaviour of the multilayer-encryption anonymity networks are conducted to study the possibility of identifying such networks. Furthermore, identifying the obfuscation techniques employed by some of the multilayer-encryption anonymity networks is attempted using the traffic flow analysis.

6.2 Machine Learning Algorithms

Decision Tree C4.5, Random Forests, Naive Bayes (NB) and Bayesian Network (BN) are the machine learning algorithms employed in this research for analyzing the extracted flows of the multilayer-encryption anonymity networks. What follows is an overview of the machine learning algorithms employed.

6.2.1 C4.5

C4.5 [80] is a supervised machine learning algorithm which builds a tree that defines a relation between the instances, attributes and classes on a training set of data. The tree can be used then to classify unseen instances based on the relationships built during the training phase. The tree divides the training set into subsets starting at a root node. The root node represents an attribute that split the training set best. The tree then split again on another decision node based on another attribute. The split decision is based on entropy and information gain. The entropy of a training data (T) with c classes is calculated as shown in Eq. 6.1:

$$Entorpy(T) = \sum_{i=1}^{c} p_i \, log_2 \, p_i \tag{6.1}$$

The probability p_i is calculated as the number of instances in class i over the total number of instances. The information gain of an attribute A on the training data (T) is calculated as shown in Eq. 6.2. v represents the possible values of attribute A. T_v is a subset of the training data (T) that has the value v of attribute A.

$$InformationGain(T, A) = Entropy(T) - \sum_{v} \frac{|T_v|}{T} Entropy(T_v) \tag{6.2}$$

C4.5 is the successor of Iterative Dichotomizer 3 which is known as an ID3 algorithm. ID3 was presented by Quinlan [79] in 1979. The ID3 algorithm [67] builds a decision tree by using a top-down greedy search from the possible decision trees. The algorithm is shown in Algorithm 1 [43]. ID3 starts by deciding the root attribute of the tree by testing all the attributes in the training set using the information gain. The root node is then branched based on the possible values of the attribute. Then the training instances are sorted to the appropriate branch. The process is repeated to select the best attribute for classifying the training examples based on the information gain.

Algorithm 1 : ID3

```
inputs R: a set of non-target attributes
       C: the target attribute
       S: a training set. returns a decision tree;

  if $S$ is empty then
    Return a single node with value Failure
  end if

  if $S$ consists of records all with the same value for the target attribute then
    Return a single leaf node with that value
  end if

  if $R$ is empty then
    Return a single node with value as the most common value of the target attribute
    values found in $S$
  end if

  $D \leftarrow the\ attribute\ that\ has\ the\ largest\ Gain\ (D, S)\ among\ all\ the\ attributes\ of\ R$

  $d_j j = 1, 2, ..., m \ \leftarrow \ Attribute\ values\ of\ D$

  $S_j\ with\ j = 1, 2, ..., m \leftarrow The\ subsets\ of\ S\ respectively\ constituted\ of\ d_j\ records$
  $attribute\ value\ D$

  $Return\ a\ tree\ whose\ root\ is\ D\ and\ the\ arcs\ are\ labelled\ by\ d1, d2, ..., d_m\ and\ going$
  $to\ sub-trees\ ID3\ (R - D, C, S1), ID3(R - DC, S2), .., ID3(R - D, C, Sm)$
```

6.2.2 Random Forests

Random forests [12] is an ensemble of trees working together to predict. Random forests build multiple decision trees, then for a given input, a vote decides the classification results based on the majority of the result of the trees. A tree on the random forests ensemble is built by selecting a set of features randomly at each node and split on theses and splitting these features based on the best possible split. The tree is built on about 66% of the training set. The instances are selected randomly from the training set with replacement. The rest of the training set (Out Of Bag) is used to calculate the error of the tree (OOB error). One of the features that random forests offers is to avoid overfitting. Random forests showed good performance and enhancement on such decision trees as C4.5. At the same time, random forests lacks the interpretability that C4.5 offers.

Bootstrap aggregating or Bagging is an ensembles method which could make different models on different random samples from the training data set. The bootstrap samples are drawn with a replacement which makes the bootstrap sample different from the original data set. Subspace sampling builds each tree from a different random subset of the features. This sampling reduces the training time of each tree. Algorithm 2 shows the random forests algorithm [35].

6.2.3 Naive Bayes

Naive Bayes [115] is a supervised classifier known for its simplicity to build. Naive Bayes is based on Bayes's theorem with an assumption of independence between features. The assumption that a feature (x) on a class (c) is independent from other features is called the class conditional independence. If the likelihood $P(x|c)$ is the probability of feature x given class c, $P(c)$ is the prior probability of class c and $P(x)$ is the prior probability of feature x. Then, the posterior probability of class c, given feature x, $P(c|x)$ is calculated as:

$$P(c|x)\,\frac{P(x|c)P(c)}{P(x)} \tag{6.3}$$

The likelihood function $P(x|c)$ is evaluated for the observed data x (feature) and it is shown as a function of c [11].

Algorithm 2 : Random Forests

```
Input:  data set D; ensemble size T; subspace dimension d.
Output:  ensemble of tree models whose predictions are to be combined by voting
or averaging.
for t=1 to T do
    build a bootstrap sample $D_t$ from $D$ by sampling $|D|$ data points with replacement;
    select $d$ features at random and reduce dimensionality of $D_t$ accordingly;
    train a tree model $M_t$ on $D_t$ without pruning;
end for
return  $M_t|1 \leq t \leq T$
```

The likelihood is not probabilistically distributed over c. Therefore, the posterior probability could be rewritten as:

$$Posterior \propto Likelihood \times prior \tag{6.4}$$

The posterior probability can be expressed in terms of the prior distribution and the likelihood function by integrating equation 6.3 with respect to c:

$$P(x) = \int P(x|c)\, P(c)\; dc \tag{6.5}$$

6.2.4 Bayesian Network

Bayesian Network (Belief Network) [36] is a probabilistic graphical model which uses a combination of graph theory and probability theory. The graph describes the relations among random variables shown as nodes on the graph. Links on the graph represent the relation (probabilistic dependencies) among the random variables. Bayesian uses a Directed Acyclic Graphs (DAG) model where links are directed to show the causality among the random variables. Nodes on the graph are represented as well by Conditional Probability Distribution (CPD) which in a discrete model can be represented as a Conditional Probability Table (CPT) for quantifying the relations among the nodes.

To demonstrate how the Bayesian Network represents a direct graph for a probability distribution [11], assume a joint distribution $p(a, b, c)$ for the variables a, b and c. The joint distribution $p(a, b, c)$ can be rewritten using the following probability rule:

$$p(X, Y) = p(Y|X)\, p(X) \tag{6.6}$$

The joint distribution $p(a, b, c)$ can be rewritten as follows:

$$p(a, b, c) = p(c|a, b)\, p(a, b) \tag{6.7}$$

$$p(a, b, c) = p(c|a, b)\, p(b|a)\, p(a) \tag{6.8}$$

Equation 6.8 could be represented by a graph where the three random variables are represented by three nodes. The conditional probabilities are represented by links on the graph between the nodes. The $p(c|a, b)$ could be presented by links from node

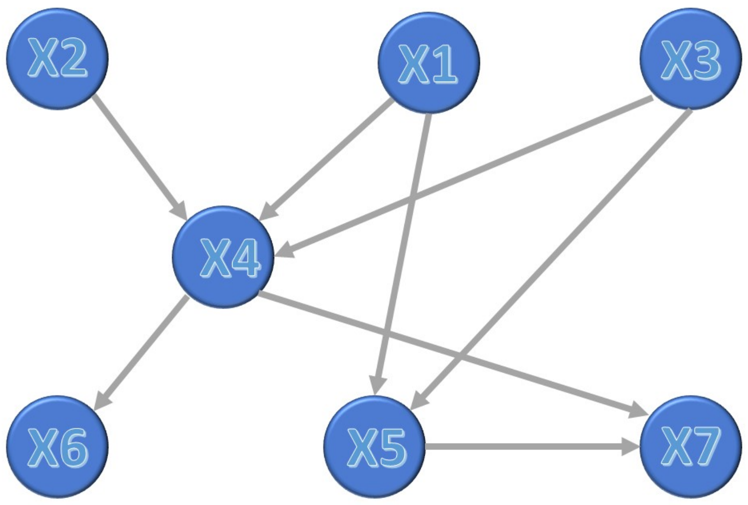


Figure 6.1: Directed Acyclic Graph for the Joint Distribution of Variables x_1 to x_7

a and node b to c. A link from node a to node b will present $p(b|a)$. The graph will not show any link for $p(a)$ because it has no conditional probability. A graph could be drawn for more than three variables. For example, if we have a K number of variables, the joint distribution $p(x_1, ..., x_K)$ could be written using the product of conditional distribution as follows:

$$p(x_1, ..., x_K) = p(x_K|x_1, ..., x_{K-1})...p(x_2|x_1)p(x_1) \tag{6.9}$$

Assume we would like to draw a graph for the joint probability distribution $p(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ which is written in the product of conditional distributions:

$$= p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5) \tag{6.10}$$

the variables x_1, x_2 and x_3 in the parts $p(x_1)p(x_2)p(x_3)$ have no conditional probability. Therefore, as shown in Figure 6.1, the nodes of these variables have no links going to them. The nodes for variables x_4, x_5, x_6 and x_7 show links coming from the other nodes representing the conditional probability.

6.3 Flow Exporters

Flow exporter tools use the following five tuples to define a flow: the source IP address, the destination IP address, the source port, the destination port and the protocol. Flow analysis employs statistical information extracted from the packet header such as packet size, average packet size and minimum and maximum packet size. In addition, information about packet transmission rates such as average inter arrival time, minimum inter arrival time and maximum inter arrival time is employed by the flow analysis as well. It is possible also to use information extracted from the used protocol. For example, if the protocol is TCP, then window size, TCP flags, Sequence and other protocol-related information could be used in the flow analysis.

There are many flow analysis tools [39] commercially available such as Cisco NetFlow [20], Juniper J-Flow [56] and InMon sFlow [51] or open source such as Softflowd [89], YAF [116], Maji [63], Tcptrace [95] and Tranalyzer [106]. Most of these tools support extracting flows from captured network traffic in forms like PCAP or directly from the network. Extracting flows first requires collecting data (flow exporter), then these flows are sent to a flow exporter to archive them, finally analyzing these flows. Some of the flow analysis tools contain the flow exporter only and require a collector, others have the exporter and collector integrated into the tool itself [39]. Figure 6.2 shows how the flow analysis tools have been employed in this thesis to extract and analyze multilayer-encryption anonymity networks. Even though flow analysis tools use mostly the same 5-tuple to extract flow information, they differ in multiple ways such as the number of generated features, the definition of flow timeout and the ability to configure this value, the number of generated flows, the supported form of captured traffic, as well as other features. Few of the open source flow analysis tools have been tested for the purpose of analyzing multilayer-encryption anonymity networks. Based on the performance of Tranalyzer over Tcptrace as shown in Chapter 7 and in our published work [85], Tranalyzer has been selected for use as the flow analysis tool in this thesis.

In this research, once the flows are exported using Tranalyzer, information such as the source IP address, the source IP address, source and destination numbers as well as flow start and end times are removed from the analysis to ensure that the classification process is not biased using this information. Not using the port numbers in the analysis eliminates the bias as well in terms of linking a port number to an application. This is important since many applications use dynamic port numbers on today's Internet. In most of the multilayer-encryption anonymity networks the port

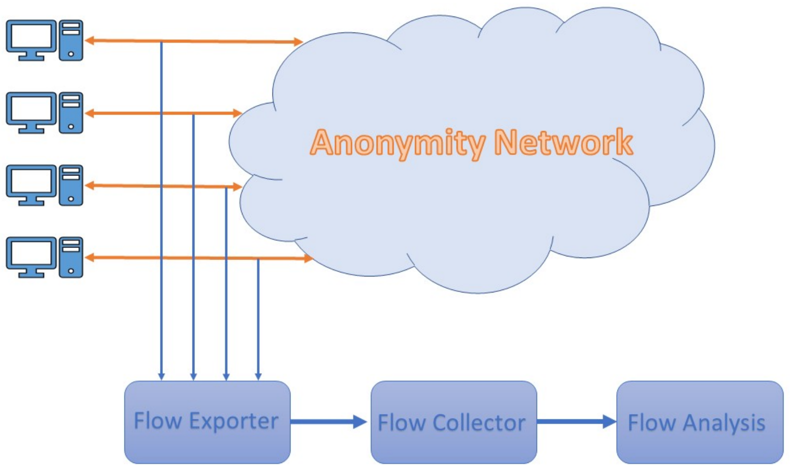


Figure 6.2: Flow Analysis on Anonymity Networks

number is configurable. For example, Pluggable transports can be configured even to use well-known ports such as port 80 or port 443.

6.4 Summary

This chapter demonstrates the methodology employed in the thesis. This includes describing the data sets used in this thesis (Anon17, plus data collected from other universities or labs). In addition, this chapter illustrates the machine learning algorithms used in the thesis. The algorithms used are C4.5, Random Forests, Naive Bayes and Bayesian Networks. The flow analysis approach and how it is implemented in this thesis to perform analysis on the multilayer-encryption anonymity networks is explained as well.

Chapter 7

Experiments on the Identification of Anonymity Networks

Multilayer-encryption anonymity networks provide privacy for the network's users by the separation between the users' requests and the final destination. The separation is achieved by relaying the users' data through multiple stations in encrypted traffic to the final destination. The multiple encryptions are used to protect the data even from any station on the path to the final destination. In this chapter flow analysis is used to study the amount of information which could be extracted from the encrypted traffic. Moreover, the analysis studies the application's behaviour on the multilayer-encryption anonymity networks and the possibility of identifying these applications.

7.1 Tor Behaviour to Circuits and Flows Analysis

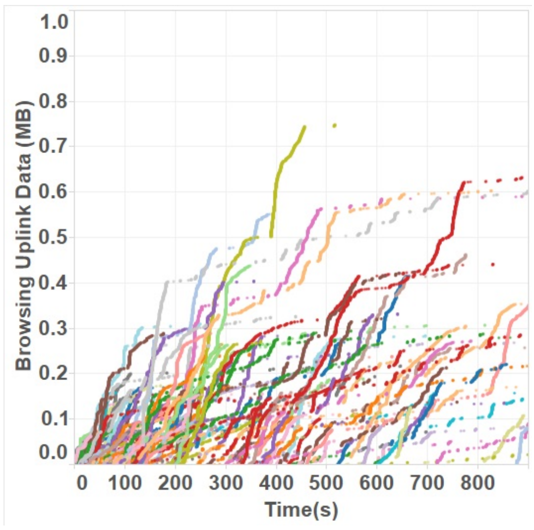
Tor provides its users with anonymity by hiding their browsing activities and locations. This level of anonymity is achieved by relaying the users' activities through three Tor nodes. When the user connects to the first relay, it knows the IP address of this user. The user activities stay encrypted while moving from the first relay to the second one and from the second one to the third one. The third relay sends the user activities on behalf of the user without any information regarding the identity of the user.

This section analyzes the amount of information which can be extracted from the encrypted Tor traffic without decrypting the traffic. To explore this, two different approaches for the classification of user activities are employed. The first approach is the Flow level classification. It depends upon analyzing the TCP communication between the user and the relay to predict the type of user activities in the encrypted traffic. The second approach is the circuit level classification. The encrypted circuits have characteristics which can be extracted and calculated to classify the type of traffic in the circuits. The main difference between the flow level classification and the circuit level classification is the level of access to the Tor network. The circuit level classification requires having access to the relay node, itself [3]. That means only the one who runs the relay can gather the required statistics about the circuit. By contrast, flow level classification only requires access to any point between the user and the relay. This means the collection of the encrypted TCP communication between the user and the relay can be on the user machine, the relay machine or the ISP.

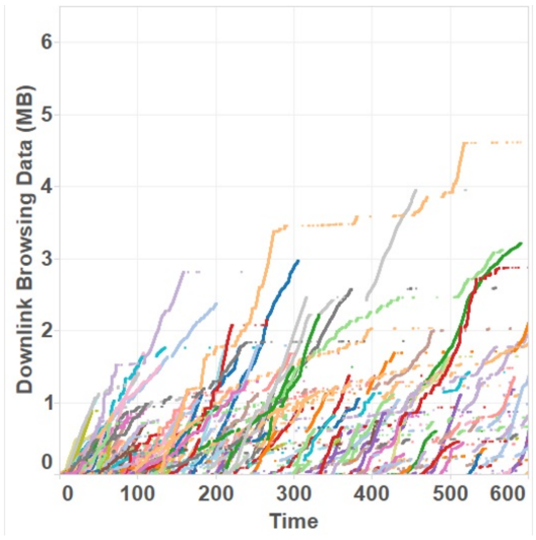
7.1.1 Cells in the Tor Network

Even though the cells in the circuit are encrypted, they have different behaviours and characteristics according to the type of traffic they carry. Cell timing, the number of downlink cells, the number of uplink cells and the number of cells in the duration of the circuit vary based on the type of traffic in the cells. There is a directly proportional relationship between the type of traffic and the number of cells that flow in the circuit: the higher the amount of data, the higher the number of cells. The uplink cells that flow from the user to the relay depend on how the application reacts to send data requests. For example, as shown in Figure 7.1, the number of uplink cells for web browsing is much lower than the uplink cells in BitTorrent. This reflects the natural behaviour of the applications. The web browsing activity uses few requests to visit a web site. By contrast, BitTorrent sends frequent requests to locate the files that the user wants to download. The circuit itself stays active for a longer duration in BitTorrent than it does in web browsing. Even if two circuits stay active for the same duration, the circuit utilization is higher in BitTorrent. When these cell statistics and timings are combined together they provide a good view for distinguishing among the different types of traffic in the cells.

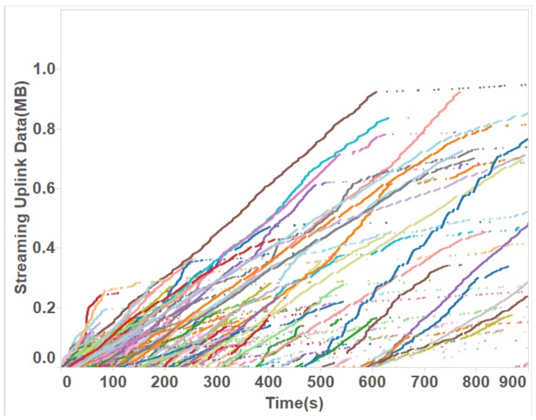
Figure 7.2 shows how the rate of uplink cells to downlink cells changes with time in web browsing circuits. In most of the circuits the rate reaches its maximum value. This rate stays only for a short period when it starts to decay quickly. The rate then starts to fluctuate up and down until it reaches a very low rate. This observation shows conformity with normal web browsing requests in which the user enters a URL and waits for the web server's reply message to download then enters another URL and so on.



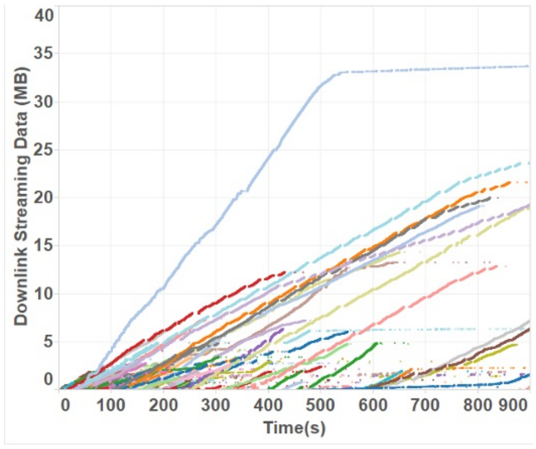
(a) Browsing Uplink



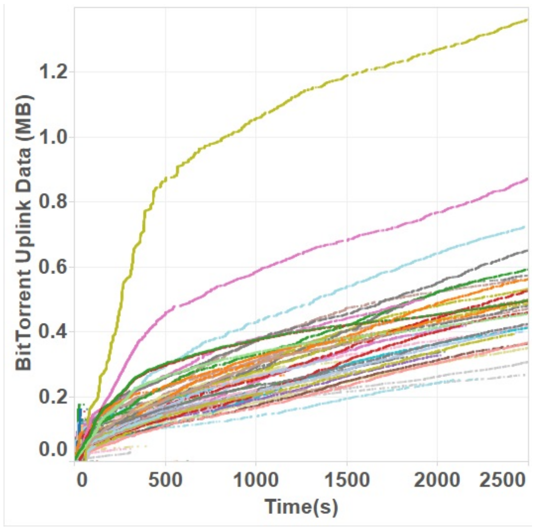
(b) Browsing Downlink



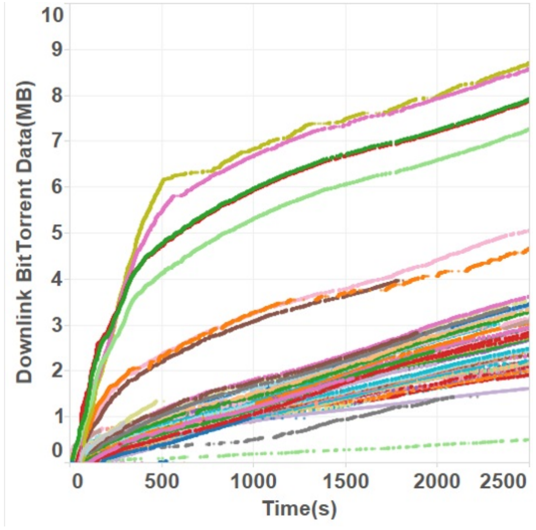
(c) Streaming Uplink



(d) Streaming Downlink



(e) Torrent Uplink



(f) Torrent Downlink

Figure 7.1: Data Transferred in the Uplink and Downlink Communications for Different Applications

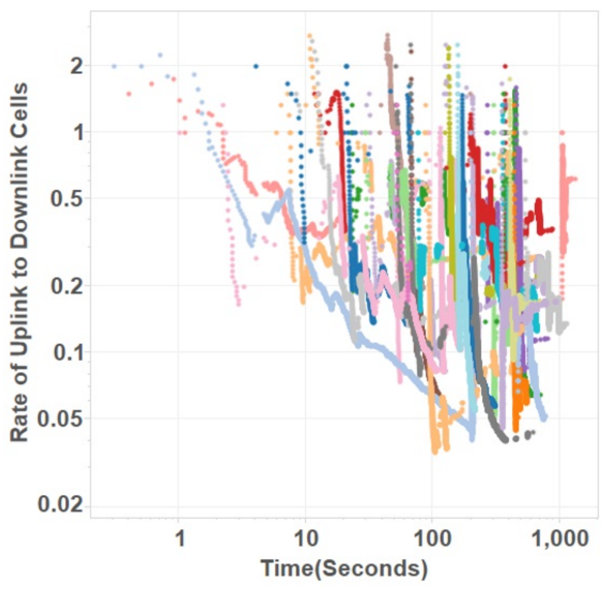


Figure 7.2: Browsing Circuit Data Rate - Different Colours Show the Rate for Different Circuits

The amount of data in the requests sent is lower than the amount of data downloaded from the web sites themselves. This causes the rate to decay with time.

Figure 7.3 shows how the rate behaviours differ for the video streaming circuits. The rate is higher than for the web browsing. The decay period is longer and smoother. The rate remains within a certain range for the rest of the circuit duration time. The BitTorrent circuits show yet another behaviour in the rate as shown in Figure 7.4. In this case, the rate reaches to 100 and decays slowly. The circuit duration is longer than both the web browsing and the video streaming. BitTorrent uplink cells stay high in numbers compared to the web browsing or the streaming. However, since the number of the downlink cells is high in BitTorrent, the rate decays with time.

7.1.2 Circuit Level Classification

This approach depends upon using statistics extracted from the circuits and their cells to classify the user activities in the Tor traffic without decrypting the traffic. It is similar to the work done by AlSabah, Bauer and Goldberg [3] but is extended with the attributes described below.

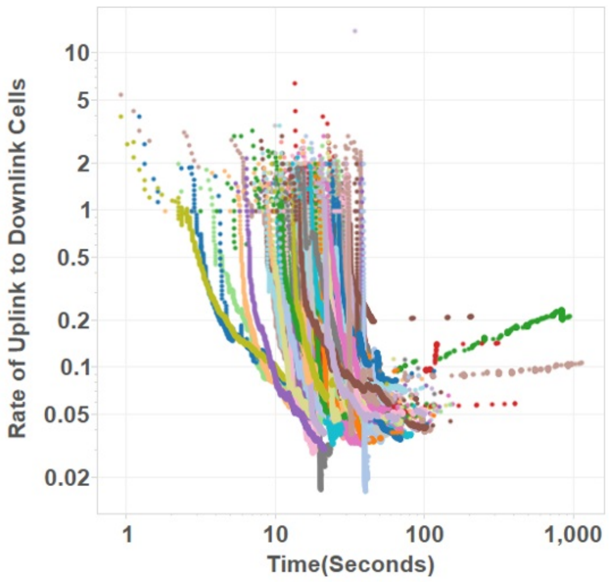


Figure 7.3: Streaming Circuit Data Rate - Different Colours Show the Rate for Different Circuits

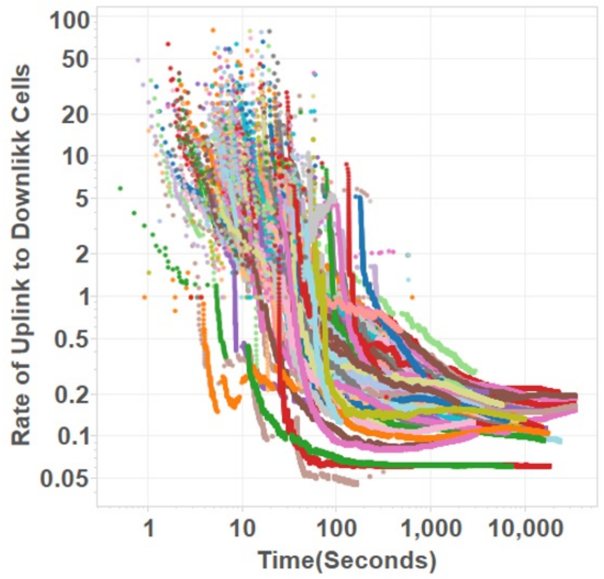


Figure 7.4: Bittorrent Circuit Data Rate - Different Colours Show the Rate for Different Circuits

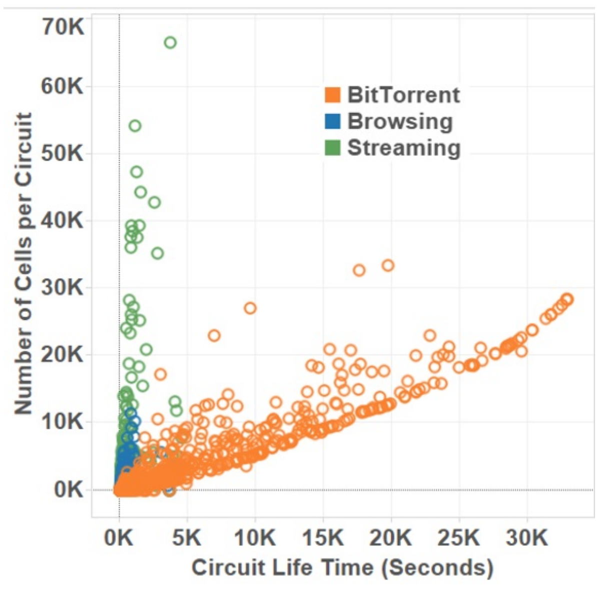


Figure 7.5: The Number of Cells During Circuit Life Time

7.1.2.1 Cells Per Circuit Life Time

The amount of data delivered by a circuit is a good indication of the type of traffic carried by the circuit. However, without considering the time the circuit is alive, this indication might be misleading. For example, two circuits might deliver the same amount of data in two different durations. As shown in Figure 7.5, the numbers of streaming circuits and BitTorrent circuits are both high compared to the number of browsing circuits. The circuit life time is what distinguishes them.

7.1.2.2 Uplink Cells

Relays' cells are transferred in both directions: uplink and downlink. BitTorrent seems to transmit a higher amount of uplink cells than the browsing or streaming applications. In other words, the number of cells required to send a request might be a good indicator of the type of traffic running on the Tor network.

7.1.2.3 The Ratio of the Downlink Cells to the Uplink Cells

Even though the uplink data is a good indication of the type of traffic the circuit has, the ratio of the uplink to the downlink is also important. If the circuit life time is long enough, it could transmit a considerable amount of uplink cells even if it is

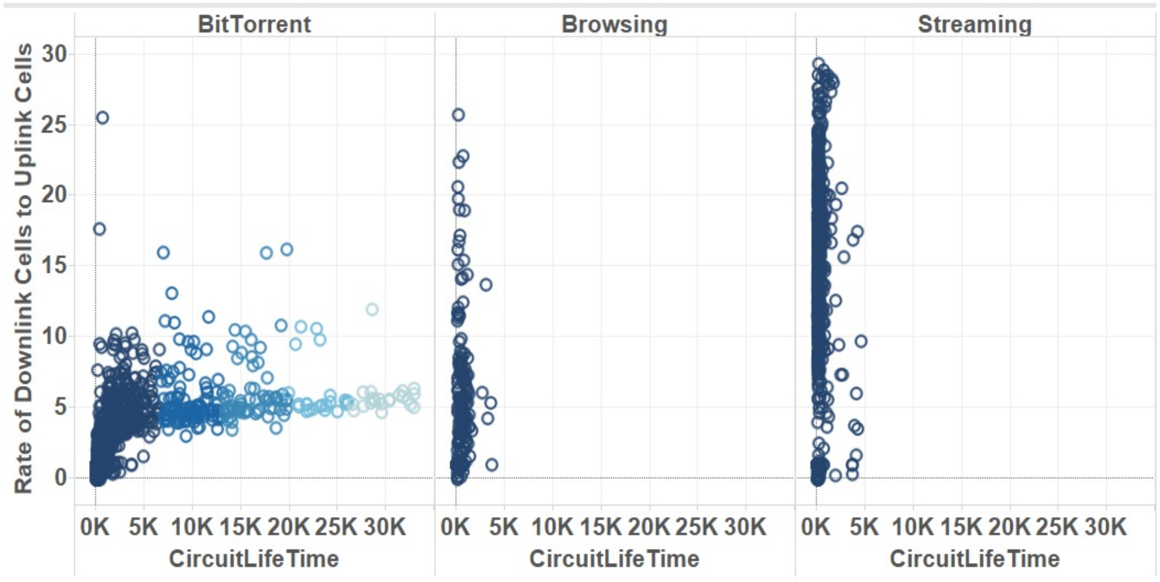


Figure 7.6: The Ratio of Downlink Cells to Uplink Cells During the Circuit Life Time

used for browsing. However, most of the time, the rate will vary based on the type of traffic no matter how long the life of the circuit is. Figure 7.6 shows the ratio of the downlink cells to the uplink cells with the circuit life time.

7.1.2.4 Exponentially Weighted Moving Average (EWMA)

EWMA is a good indicator because it can differentiate between two circuits when both have the same life time. Eq. 7.1 is used to calculate the EWMA of the cells [72]:

$$\bar{x}_k = \alpha\,\bar{x}_{k-1} + (1-\alpha)\,x_k \tag{7.1}$$

where $\bar{x}_k$ is the current average. $\bar{x}_{k-1}$ is the previous average. x_k is the current measurement. α is a factor that indicates the weight for the old value, $0 \leq \alpha < 1$.

7.1.3 Flow Level Classification

In this approach, the traffic flows between the node and the client are employed to classify the user activities in the Tor traffic. Two flow exporting tools are used to generate the flows and to extract the attributes of the flows between the client and the Tor node. They are described below.

Tranalyzer2. This is an open source tool which generates flows from a captured traffic file or directly by working on the network interface. It is based on the libpcap library. Its output can be exported directly to a text file without the need for an external collector [106]. The output depends on the type of plugins enabled when using Tranalyzer2 to analyze the captured traffic. Choosing the proper plugins is important for generating the features that are useful for the classification task.

Tcptrace. This is a tool used to do a TCP analysis. It can handle different types of captured traffic formats. Different connection graphs can be drawn using Tcptrace. The output can be exported to a text file and a CSV file [95]. The attributes vary based on the tools used. Each one of them generates a different number of attributes. The number of attributes depends on the options (plugin) enabled in the tool itself. Selecting which attributes to include in the classification of Tor is very important.

The flow between the client and the node contains much other information which is used by the flow exporters to generate the attributes. For example, the IP address, the port number, the MAC address, the time when the flow starts or ends, the sequence number and any information related to the physical client but not to the data between the client and the node. Table 7.1 shows the number of attributes that Tranalyzer2 and Tcptrace provide in the first column. The second column shows the number of attributes selected by the classifiers from the full set provided. Tor uses the TCP protocol for establishing the connection between the user and the relay. Thus, any attributes which are based on any protocol other than the TCP protocol are excluded. Furthermore, the source port number, the destination port number, the source IP address and the destination IP address attributes are also excluded in the used set. The IP address is used to label the user's activity since it is known which IP address runs which application on this testbed. Finally, the payload is not employed in any attribute set, only statistics about the payload.

Tool	**Number of Attributes**	**Number of Selected Attributes**
Tranalyzer2	91	62
Tcptrace	88	50

Table 7.1: Flow exporter attributes.

7.1.4 Evaluation of Circuit and Flow Level Approaches

Two different sets of experiments were performed to verify the two approaches employed in this research (circuit level classification and flow level classification). What follows details these experiments.

7.1.4.1 Setup

In these experiments a testbed is set up with three clients. These three clients are used to generate the three traffic types automatically: Browsing, Streaming and BitTorrent. All the clients are configured to use this research node as the sole entry node. iMacro [50] is used to automate the browsing and the streaming activities. Both of them start by searching for random words generated by the 26 letters in the English alphabet. They pick one of the results randomly. In the case of Browsing, the iMacro script picks one of the search results and navigates through the site by clicking a random hyperlink existing on that web site. It keeps moving from one link to another. For each link, the client will stay for a random period of time on that web page then move again to another link.

The setting is similar in the streaming activities. The script will move from one video to another. The time spent for watching a video or moving from a link to another is set randomly. The main difference between the times spent in the browsing and the streaming is that the streaming one is longer than the browsing time setting. Deluge [22] is used as the BitTorrent software. A proxy is used to be an interface between the clients' traffic and the Tor node. The socket port in Tor is enabled to listen for the proxy requests.

The Tor source code is edited to log only incoming circuit creation requests from the node set for this research. The IP address of each client is used to enable the logging and also to label the circuit as one of the three traffic types. This maintains the privacy of the users on the Tor network who connect to the node used in this research.

7.1.4.2 Circuit Level Classification Data

The data in the circuit level classification is entirely different from the data in the flow level classification. In the circuit level classification the data includes information about the cells and the circuits between two given Tor nodes and between the clients and the node. By contrast, in the flow level classification the data includes the flows between the clients and the node. The circuit level classification is based on the cell statistics per circuit to distinguish between each traffic type. The Tor source code is edited to relate each circuit to its source IP. Even for the relay cells where Tor switches the circuit ID, the Tor logging is set to log this ID in its previous circuit ID, not the new switched ID. In this way, the uplink and the downlink cells related to each class can be tracked.

The log for each circuit contains the circuit ID, the class, the time when the circuit is created and the time when the circuit is destroyed. The cell's log contains the class and the circuit ID each cell belongs to, the cell arrival time and the direction of the cell. The attributes needed in the circuit level classification are then extracted from the circuit's log and the cell's log files. To maintain the privacy of other users, circuits related to the other users using the node run by this research have not been logged.

7.1.4.3 Flow Level Classification Data

PCAP network traces are collected in the entry node machine for the duration of the experiment (ten hours). The required connection information is the traffic between the entry node in this research and each client. To keep the privacy of other users who use the node run by this research a filter is applied for the capturing so only the connections from the clients in this research are logged. For the duration of the experiment, 4.2 GB of data are captured.

7.1.5 Performance Metrics

To obtain a clear picture of the results, the same metrics as used by AlSabah, Bauer and Goldberg [3] are calculated (accuracy and F-measure). This enables a comparison between the two approaches as well. The first metric Accuracy is defined as the summation of True Positive (TP) and true negative (TN) values divided by the total number of instances (N), as shown in Eq. 7.2. For example, when measuring the accuracy of the classification for the browsing application, TP is the total number of instances classified correctly as browsing. TN is the total number of instances classified correctly as non-browsing. As shown in Eq. 7.3, Precision is the ratio of TP divided by the summation of TP and False Positive (FP). If the classifier classifies an instance as browsing and the right type is not browsing then this is considered as an FP measure. The opposite occurs when the classifier classifies an instance to be non-browsing while it is a browsing instance, then this is a False Negative (FN). Eq. 7.3 defines the Recall as the division of TP over the summation of TP and FN. The relation between the precision and recall is shown in Eq. 7.5.

$$Accuracy = \frac{TruePositive(TP) + TrueNegative(TN)}{NumberofInstances(N)} \tag{7.2}$$

$$Precision = \frac{TruePositive(TP)}{TruePositive(TP) + FalsePositive(FP)} \tag{7.3}$$

$$Recall = \frac{TruePositive(TP)}{TruePositive(TP) + FalseNegative(FN)} \tag{7.4}$$

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{7.5}$$

7.1.6 Results and Discussion

The circuit level and the flow level approaches both give a high level of classification accuracy. Even though the machine learning algorithms used in this research are similar in both approaches, the same algorithm gives a different accuracy depending on the approach used. The details of these results are discussed in the following sections.

7.1.6.1 Circuit Level Classification Results

In this approach the circuit level classification uses the attributes of the cell. The data set consists of 60% browsing, 20% streaming and 20% BitTorrent activities. Table 7.2 shows the results for the offline circuit level classification. In this case, accuracy reaches to 100% when using the C4.5 classifier with 70% of the instances as the training set. When 10-Fold cross-validation was used the best accuracy was 94.9% using the Random Forest classifier. In Table 7.2 (and thereafter), “split” refers to

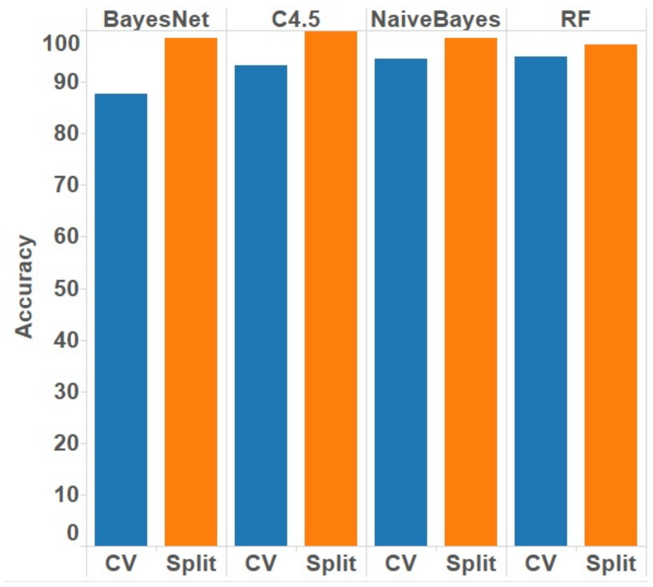


Figure 7.7: Circuit Level Classification Accuracy

the results obtained on the test data set (30% of the data) using the model generated from the training data set (70% of the data).

Classifier		Accuracy	F-Measure		
			Browsing	Streaming	BitTorrent
Bayes net	SPLIT	98.6%	0.99	0.97	1
	CV	87.7%	0.89	0.82	0.98
Naive Bayes	SPLIT	98.6%	0.99	0.97	1
	CV	94.5%	0.95	0.88	1
C4.5	SPLIT	100%	1	1	1
	CV	93.2%	0.94	0.83	1
RF	SPLIT	97.2%	0.98	0.97	0.96
	CV	94.9%	0.96	0.88	1

Table 7.2: Circuit level classification results.

On the other hand, "CV" refers to the results obtained using the 10-fold-Cross-Validation. Figure 7.7 shows the results for offline circuit level classification for all classifiers using the 70-30 training/testing split and the 10-fold cross validation.

7.1.6.2 Flow Level Classification Results

In this case the data set consists of the uniform distribution of the three classes. The number of flows is approximately the same for each class. Even though the amount of data transferred by the Streaming and the BitTorrent circuits is larger than the browsing circuits, they are similar in terms of the number of flow instances. Table 7.3 shows the results when using the uniform distribution of the three classes. The results are 88% to 100% when using 70% of the instances as the training set. When using 10-Fold cross-validation results are 86% to 99%. It should be noted here that the data set employed by AlSabah, Bauer and Goldberg [3] consists of 60% browsing instances, 20% streaming instances and 20% BitTorrent instances. Those authors [3] generated such a data set to mimic the traffic distributions of Tor users. To make the results of this research comparable to theirs, the data is downsampled to the same percentages.

	Classifier		**Accuracy**	**F-Measure**		
				Browsing	Streaming	BitTorrent
Tranalyzer2	**Bayes Net**	SPLIT	100%	1	1	1
		CV	99.2%	0.99	0.99	1
	Naive Bayes	SPLIT	94.7%	0.98	0.91	0.94
		CV	93.3%	0.98	0.90	0.93
	C4.5	SPLIT	98.7%	0.98	0.98	1
		CV	97.2%	0.96	0.98	0.98
	RF	SPLIT	98.7%	0.98	0.98	1
		CV	98.8%	0.99	0.98	0.99
Tcptrace	**Bayes Net**	SPLIT	97.4%	1	0.96	0.96
		CV	97.7%	0.98	0.97	0.99
	Naive Bayes	SPLIT	97.4%	0.96	1	0.96
		CV	92.2%	0.96	0.89	0.92
	C4.5	SPLIT	94.87%	0.96	0.92	0.97
		CV	96.1%	0.96	0.95	0.98
	RF	SPLIT	97.4%	1	0.96	0.97
		CV	97.7%	0.99	0.96	0.98

Table 7.3: Flow level classification results - uniform classes.

Table 7.4 shows the results when the data is downsampled to get the same distributions in the work done by AlSabah, Bauer and Goldberg [3]. In this case the results still show high accuracy in both the split and cross validation cases. The Bayes Net classifier achieved 100% in both the split and the cross validation cases. Figure 7.8 shows the accuracy of all the classifiers when using cross validation

	Classifier		Accuracy	F-Measure		
				Browsing	Streaming	BitTorrent
TRANALYZER2	**Bayes net**	SPLIT	100%	1	1	1
		CV	100%	1	1	1
	Naive Bayes	SPLIT	100%	1	1	1
		CV	95.7%	0.99	0.89	0.91
	C4.5	SPLIT	97.6%	0.98	1	0.93
		CV	99.3%	0.99	1	0.98
	RF	SPLIT	100%	1	1	1
		CV	98.6%	1	0.96	0.96
TCPTRACE	**Bayes net**	SPLIT	81.8%	0.95	0.71	0.73
		CV	93.3%	0.98	0.84	0.90
	Naive Bayes	SPLIT	77.3%	0.95	0.55	0.71
		CV	82.3%	0.94	0.60	0.73
	C4.5	SPLIT	81.8%	0.82	0.91	0.73
		CV	92.0%	0.94	0.83	0.94
	RF	SPLIT	90.9%	1	0.83	0.83
		CV	94.7%	0.98	0.86	0.93

Table 7.4: Flow level classification results - downsampled classes.

7.1.6.3 The Performances of the Classifiers Employed

Table 7.5 shows a comparison between the best results achieved in this research and the work done by AlSabah, Bauer and Goldberg [3]. The performance improved by 9%. In the work done by AlSabah, Bauer and Goldberg [3], the accuracy was 91% when using functional tree with 10-Fold Cross-Validation. The classification level was a circuit level. In this research the accuracy reached 100% by using the circuit level classification with the extended attribute set as well as when using the Flow level classification. For the circuit level classification this result is achieved when using the C4.5 classifier with a 70% training set.

For the Flow level classification the accuracy is 100% when using Bayes Net, Naive Bayes and Random Forest classifiers with 70% of the training data. The same result is achieved as well when using the Byes Net classifier with 10-Fold Cross-Validation.

The improvement in the circuit level classification compared to the result achieved by AlSabah, Bauer and Goldberg [3] is due to the features added to the classification. The BitTorrent circuits tend to have a longer lifetime than the browsing circuits. However, there is a small percentage of the BitTorrent circuits that stay active for a short time period and then are destroyed. These circuits let the circuit behaviour of BitTorrent look similar to the browsing circuit behaviour. This similarity reduces

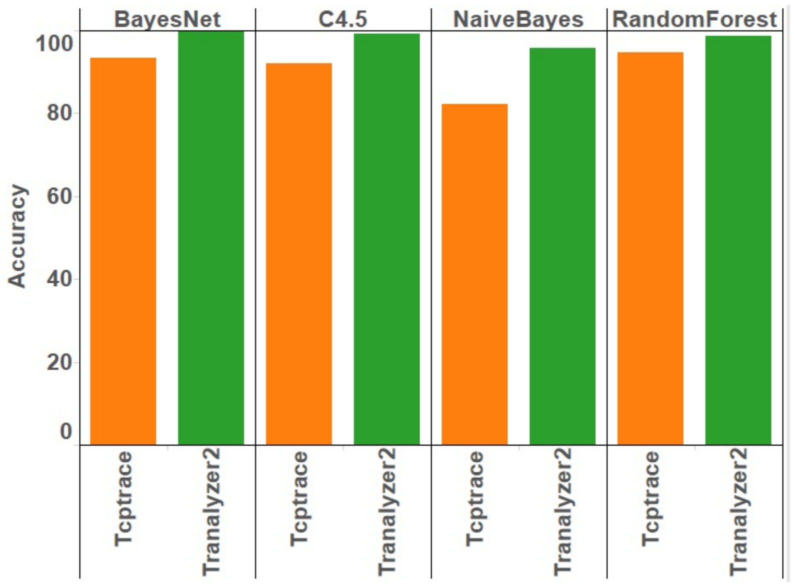


Figure 7.8: Flow Level Classification Accuracy Using Cross-Validation

	Accuracy
Results in the work done by AlSabah, Bauer and Goldberg [3]	91% Circuit Level Classification Functional Tree 10-Fold Cross-Validation
Results of this Research	100% Circuit Level Classification C 4.5 Classifier 70% training set
	100% Flow Level Classification Bayes Net, Naive Bayes and Random Forest classifiers 70% training set
	100% Flow Level Classification Bayes Net classifier 10-Fold Cross-Validation

Table 7.5: Methods used to achieve the best accuracy.

the classification performance in the work done by AlSabah, Bauer and Goldberg [3]. It can be seen that by adding the number of cells transferred during the circuit lifetime makes the circuits more distinguishable. On the other hand, in the browsing circuits, the amount of data per circuit fluctuates between high and low values based on the user's browsing habits and the website content. The same applies to the BitTorrent circuits where not all circuits transfer high amounts of data. Consequently, differentiating between the combined uplink and downlink data per circuit increases the classification accuracy. Consequently, the uplink data is added alone as a feature to address this issue. While adding the uplink data provides a good indication, there is a similarity between the browsing circuits and the streaming circuits in the number of uplink cells. Consequently, employing the rate of the downlink to the uplink cells is useful for improving the distinction between the classes.

The circuit level classification requires access to the network connection traffic between the user's machine and the Tor relay. Moreover, in the circuit level classification, access to the relay itself is required. That means the circuit level classification has its limitations in terms of who can use it. On the other hand, the flow level classification can be performed on any captured data.

From the availability perspective, flow level classification is easier to apply compared with circuit level classification. The tools used in the flow level classification are available online. These tools include traffic capturing (such as Tcpdump and Wireshark) and flow exporters for generating the flow statistics (such as Tcptrace and Tranalyzer2).

Capturing similar data in circuit level classification (as done in this research) which is ready to be classified by the ML algorithms requires additional effort compared with flow level classification data. Tor itself and Tor's tools can be used to provide such information about the usage of Tor's circuits and cells. However, both of these provide a limited amount of information for doing the classification.

The circuit and the cell information in the Tor code maintains a certain level of privacy for the Tor users. For instance, the cell statistics feature (CELL STAT) which was introduced in Tor version 0.2.5.2-alpha is only available when Tor is working in the test mode. This mode works only when the "TestingTorNetwork" value is set in the Tor configuration file as described in the Tor control protocol [104]. That means using this feature will work only on private networks, not on the live Tor network. STEM [92] can provide the user with information about Tor as well. It provides a library for the user to communicate with Tor using the Tor control protocol. It can generate useful information about the circuit, bandwidth usage, etc. However, for the same previous example, STEM cannot retrieve the cell statistics by using the CELL STAT event unless Tor is working in the test mode.

Finally, the results show high accuracy in both techniques. The flow level is based on the assumption that each flow represents only one class. In the circuit level classification this assumption is always true where each circuit will represent one of the classes.

7.2 The Effects of Shared Bandwidth on I2P Tunnels

The I2P network uses separate tunnels for the outgoing and incoming traffic. That means it is impossible to link between the user's sent and received data while observing a tunnel. In addition, the inbound tunnels of the users are used for receiving any messages from any source. In this way it is not possible for an attacker who is observing the connection to detect the source of a message sent to a user.

The tunnels are used to send and receive messages, to communicate with the netDb and to manage the tunnels. Consequently, the messages which travel through the user's tunnels do not always represent only the messages travelling between the users. So, if the tunnels contain this type of control and user messages mixed together and the incoming/outgoing tunnels are separated, then this section investigates the following research which follow. What is the effect of such a design in terms of anonymizing the NetFlow behaviour of a user's activities? Can a user's activities be completely anonymized by this design or do they rely on the amount of other user traffic which shares the bandwidth?

The tunnels in the I2P network are short-lived which lessens the possibility of profiling the user's activity based on monitoring the tunnels. To overcome this, is it possible to collect information about multiple short-lived tunnels (related to the same user) in order to profile the user's activities? Does this give an indication of the level of the overhead (the influence of the overhead due to routing traffic of other users) when the user participates on the netDb or when other users' traffic is carried to hide activities? To find answers to these questions, this section studies: (i) the ability to identify the type of application the user is using; (ii) the effect of bandwidth participation on the ability to identify the type of an application; (iii) the effect of bandwidth participation on the ability to profile the users; and (iv) regardless of the application used, the ability to profile the user and to distinguish between different users by observing the tunnels.

7.2.1 Data Collection and Setup

Three machines (computers) were used to collect data on the I2P network. The version of the I2P software used on these machines was (0.9.16). The applications under study (on the I2P network) are browsing, chat and file downloading. The reason behind choosing these applications is that they are the most used ones. On each machine only one application is running at a time while collecting the data. This is to ensure the ground truth of the data. All the traffic of the applications and the traffic of the users are collected from machines in NIMS lab and do not include any other user traffic. Where the three machines employ other users' tunnels, the other users' privacy has been preserved. The encryption used on I2P keeps the users' data private. In addition, before analyzing the traffic, all the IP addresses and payloads are removed.

7.2.1.1 Browsing

To collect the browsing data a list was prepared with the available Eepsites on the I2P by default. This list includes the built-in (bookmarked) Eepsites on the I2P software such as (i2p-projekt.i2p). In addition to these web sites, some other Eepsites were added to the list by using Eepsites which provide a "search" service on the I2P network. After the list was ready, iMacro [50] was used to automate the browsing. To this end, a script was written which browses the first address on the list. Then it waits for a random period of time before it navigates through the Eepsite by clicking randomly on a link on the Eepsite. After moving (traversing) from one link to another multiple times by using this approach, the script picks the second link in the list and so on. The randomness in picking the link ensures that the visited Eepsites keep changing from one iteration to another. Some of the Eepsites contain links to websites outside of the I2P network. This results in the data collection includes traffic to websites hosted outside of the I2P network as well but still accessed through the I2P network. This requires using an outproxy (a router on the I2P network works as a proxy to access websites outside of the I2P network). To this end, the default outproxies of I2P were used (false.i2p and outproxy-tor.meeh.i2p). To be able to collect real-life data all the tunnel configurations were set to default.

7.2.1.2 Instant Relay Chat

For IRC each machine in this research was set up to work independently from the others as well. Again, only one type of application was working while collecting the data. During this process the jIRCii [53] plugin [47] was chosen and installed on the three machines. Then the machines were connected to the Irc2P network (this is the Instant Relay Chat for I2P) by using the Irc2P Tunnel and one of the following servers: (irc.dg.i2p), (irc.postman.i2p) or (irc.echelon.i2p). The machines stayed connected 24/7 on the Irc2P network and joined multiple channels such as #i2p, #i2pchat and #i2people.

7.2.1.3 Downloading Files Using Torrent (I2PSnark)

To download files on the I2P network, I2PSnark [49] was used on all machines. It is one of the built-in applications within the I2P network. The downloaded files included files such as videos, documents, music, movies, etc. The sizes of the files vary from small to big. The torrent files were obtained from the Eepsite (diftracker.i2p and tracker.postman.i2p). The data of the torrent includes both the uplink and the downlink of the files.

7.2.2 Data Analysis

The analysis of the collected data includes tunnels-based data analysis, applications and user-based data analysis and tunnel clustering as described below.

7.2.2.1 Tunnel-Based Data Analysis

In this case, the focus is on differentiating application tunnels from Exploratory and Participating tunnels [46]. Exploratory tunnels are used for management (administration/control traffic of the I2P network) and also for testing purposes. The Participating tunnels are those used to relay other users' traffic. In the training phase of the classifier, to train a decision tree model in order to differentiate the application tunnels from Exploratory and Participating tunnels, the I2Psnark, Irc2p and the shared clients' tunnels are labelled as the applications tunnels class. The Exploratory tunnels and the Participating tunnels (when the bandwidth setting is set to a default

80% for participating) are labelled as one class, called “others”. The reason behind this is to investigate the ability to distinguish application traffic from management or other users’ traffic. In this way, it is a binary classification problem in which one represents the “applications” and the other represents “other” shared traffic. In this case, the analysis shows that these two groups of traffic can be differentiated in I2P tunnels with up to 82% accuracy. Table 7.6 shows the performance of the classifier on the test data, which was unseen by the classifier during the training, for this analysis.

	TP Rate	**FP Rate**	**TN Rate**	**FN Rate**
Applications Tunnels	0.875	0.288	0.712	0.125
Others (Exploratory & Participating Tunnels)	0.712	0.125	0.875	0.288
Accuracy	**82.04%**			

Table 7.6: Binary classifier on the tunnels.

The other goal is to analyze for what purpose a tunnel might be used. In this case, if an application is running (such as 2Psnark), then the tunnels related to I2Psnark are extracted and labelled as I2Psnark. The same applies for jIRCii and Eepsites. The Eepsites tunnels, which are the client tunnels, might be used for another application on the I2P network. They also stay alive all the time that the user is online. On the other hand, the I2Psnark (Irc2P) tunnels stay alive as long as the user uses the application. The shared client tunnels could be used for I2Psnark (if the user changes the setting) but the default setting is for using the Irc2P tunnels. The Exploratory and Participating tunnels stay alive as they are. Aiming to shed light on the purpose for which a tunnel might be used, is a very challenging problem. However, the results still achieved 70% accuracy (on the unseen test data) in predicting the potential purpose of a tunnel on the I2P network by just analyzing the flow features. Table 7.7 presents the results of this analysis.

7.2.2.2 Applications and User-Based Data Analysis

In this part of the analysis the effect of bandwidth participation on the I2P network was examined using two scenarios: the first one entails the identification of the type

	TP Rate	**FP Rate**	**TN Rate**	**FN Rate**
I2Psnark	0.661	0.033	0.967	0.339
jIRCii	0.778	0.084	0.916	0.222
Eepsites	0.531	0.143	0.857	0.469
Others (Exploratory & Participating Tunnels)	0.755	0.152	0.848	0.245
Accuracy	**70.3%**			

Table 7.7: Classification results for the tunnel based traffic analysis.

of application the user is running (Traffic Profiling); and the second one is the ability to profile the users based on the amount of shared bandwidth (User Profiling). In both scenarios, an investigation of the protocol used (TCP or UDP) on improving the detection rate was included.

In the traffic identification scenario (Traffic Profiling), the data are ladled as Eepsites, I2PSnark and jIRCii. In this way the traffic of one application includes the behaviour of the traffic of multiple users using the same application. The important difference in this part is that when running an application, for example, I2PSnark, all the tunnels (exploratory, shared client and participant if any) are intentionally labelled as I2PSnark. This way it is possible to test if the overhead of the Exploratory tunnels and the Participant tunnels will affect the ability to distinguish the application type.

In the user identification scenario (User Profiling), the data is labelled as Machine 1, Machine 2 and Machine 3, since each machine was used by only one user. In this case the Machine 1 traffic will include the I2PSnark, jIRCii and Eepsites data generated from Machine 1. The same applies to Machine 2 and Machine 3. The purpose of combining different traffic from each machine into one class is to mimic user behaviour while using multiple applications and to evaluate the ability to analyze the I2P user behaviours.

On the I2P network the traffic could be in the form of TCP or UDP Traffic. Consequently, the analysis includes as well the separation of the traffic based on the protocol in both scenarios (the traffic and the user profiling) and on both bandwidth cases.

What follows is a summary of the results of both scenarios in addition to the effect of the protocol separation on the test data.

With Bandwidth Participation Table 7.8 shows the accuracy per class for the Traffic and User profiling when the amount of shared bandwidth is 80%, which is the default case on the I2P network. The accuracy measures the percentage of correctly classified instances out of all instances. It should be noted here that even though both the IP addresses and the port numbers were not used in the analysis, the result can achieve 80-86% accuracy for differentiating one user from another. However, it seems like differentiating traffic behaviour in terms of protocols is much more challenging.

80% Bandwidth Participation		
	Number of Instances (flows)	**Accuracy (%)**
Traffic Profiling	190,000	47.4
Traffic Profiling TCP Only	61,453	61.7
Traffic Profiling UDP Only	128,547	56.3
User Profiling	189,906	81.8
User Profiling TCP Only	62,882	86
User Profiling UDP Only	127,024	79.8

Table 7.8: Summary of traffic and user profiling performance.

Without Bandwidth Participation The configuration used in Section 7.2.1 was achieved by activating the default bandwidth configuration (300 KBps In, 60 KBps Out) of an I2P client. Under this setting, the bandwidth participation is 80% which is equal to 48KBps. To observe and study the effect of this amount of participation on anonymity, the bandwidth participation parameter was configured on the I2P client to 0%. In both cases, the floodfill was disabled. Table 7.9 presents the results of the analysis of the traffic and user profiling when the bandwidth participation was set to 0% and effectively not allowing any bandwidth sharing. In this case, while the user profiling drops by 15%, traffic profiling increases by 20%. Intuitively, this was expected because with no traffic sharing finding patterns in the tunnels is more likely to happen. However, under the same conditions differentiating users/machines without using IP addresses and port numbers is more challenging.

0% Bandwidth Participation		
	Number of Instances (flows)	**Accuracy (%)**
Traffic Profiling	195,081	73.7
Traffic Profiling TCP Only	40,075	65.6
Traffic Profiling UDP Only	155,006	75.7
User Profiling	195,081	66.7
User Profiling TCP Only	40,075	81.7
User Profiling UDP Only	155,006	63.2

Table 7.9: Summary of traffic and user profiling performance without bandwidth sharing.

7.2.3 Clustering Tunnels Using SOM

Based on the analysis in Section 7.2.1, the classification of tunnels seems to be more challenging than the classification of users. Also, the confusion matrices of the classifiers show that there is an overlap between the tunnel classes. Therefore, an artificial neural network based on an unsupervised learning algorithm, namely Self-Organization Map (SOM) [57] was employed to cluster and visualize the different patterns (if any) which may exist in the data of the tunnels captured in this research. For this purpose the Matlab [64] SOM toolbox [90] was used. Figure 7.9 presents the visualization of the SOM Clusters (groupings) on the data consisting of four classes: I2PSnark, jIRCii, Eepsites and Exploratory & Participating Tunnels. In this figure, the four clusters are represented in four different colours.

SOM is an unsupervised learning technique, so no labelled data is used during the training phase. However, post training labels were used to analyze the performance of this clustering algorithm on the data sets. Figure 7.10 shows the hits of the four post-training classes on the SOM Map introduced in Figure 7.9. This means Figure 7.10 is obtained by projecting the instances of the labelled data on the SOM Map in Figure 7.9.

The ideal case occurs when each class is represented by a separate cluster on the map which means that the map has a good representation of the data. In Figure 7.10, one cluster, the yellow hexagons, represent the I2PSnark tunnels. Another cluster, the magenta hexagons, represent the Exploratory & Participating Tunnels. The third cluster shown in red represents the hits of the jIRCii tunnels on the map. The

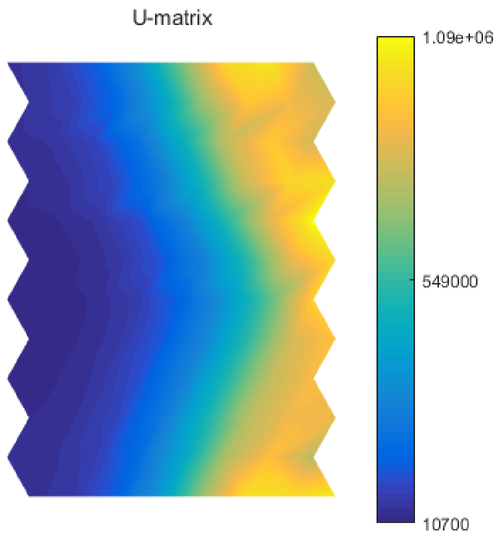


Figure 7.9: Tunnels on the SOM Map - “Sheet” Shape

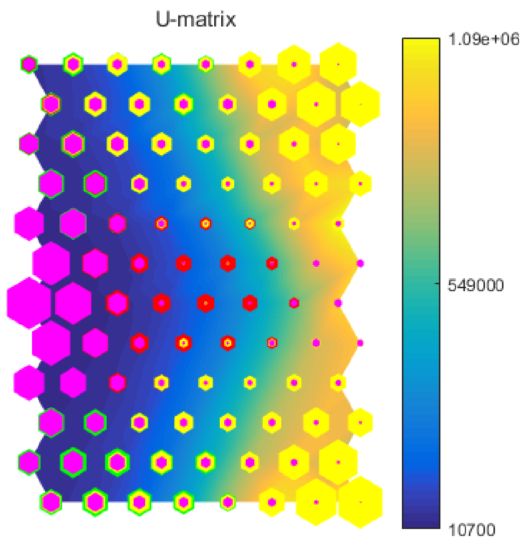


Figure 7.10: Hits on the SOM Map for All Classes

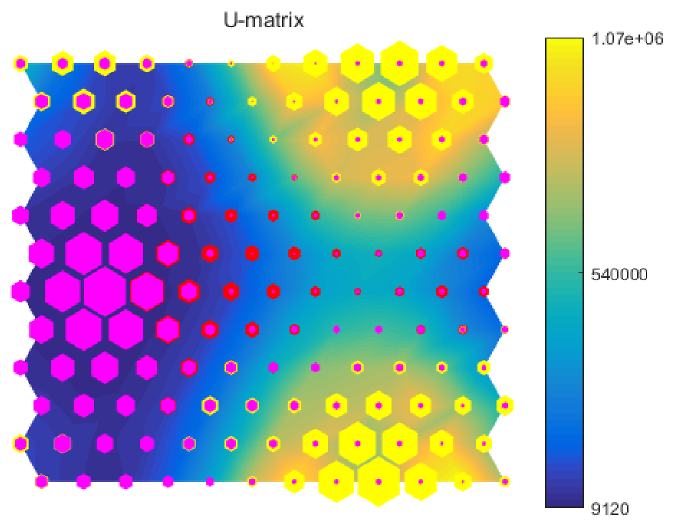


Figure 7.11: Hits for the Merged Eepsites and Exploratory & Participating Tunnels - “Cyl” Shape

green ones represent the Eepsite hits on the map. Based on how these clusters are distributed on the SOM, the Eepsite data flows seem to overlap with the Exploratory & Participating Tunnel data flows, namely, the magenta ones. Thus, based on the SOM output, the Eepsite and the Exploratory & Participating tunnels (green and magenta) seem to be grouped together. Actually, this matches with how the I2P tunnels are used. The I2PSnark and the jIRCii both use separate tunnels (I2PSnark & Irc2P Tunnels). The client Tunnels are the tunnels which are used for the Eepsites. Therefore, in Figure 7.11 the Eepsites were grouped with the Exploratory & Participating tunnels to form one class.

7.2.4 Discussion

When collecting the data, the information of the client tunnels was used to label the data for a better level of accuracy. For example, if the machine is running jIRCii and connecting to a participant in one of the inbound or outbound tunnels and that participant is labelled for IRC traffic, that does not mean that that participant will not be part of any other. Indeed, this adds a challenge to the data analysis problem in this research.

The resource sharing (bandwidth participation) increased the anonymity level when profiling the applications. The shared client tunnels are used for the Eepsite application. They could be configured to be used for other applications as well. The default is to use client tunnels for Eepsites, while separate tunnels are used for I2PSnark and Irc2P. Furthermore, when application tunnels are grouped as one class and the Exploratory tunnels in another class, this increased the accuracy of profiling the applications. Therefore, based on the results, forcing all the applications to use the client tunnels will improve the users' anonymity on the I2P network. On the other hand, increasing the bandwidth participation improves the ability to profile the users. When the user allocates more resources to participate on the network that means more traffic flows on the network belong to the user. This seems to enhance the profiling of the users. Moreover, the unsupervised learning algorithm SOM shows that the Eepsite tunnels tend to have similar behaviours with the Exploratory and Participating tunnels. Therefore, when the Eepsite tunnels were merged with the Exploratory and Participating tunnels to find the hits on the SOM, it showed more consistent behaviour. This reinforces that the separation between the tunnels for different applications seems to enhance the application profiling. Consequently, changing the default setting on the I2P client to force applications such as IRC to use the "shared client" tunnels hardens the application profiling which consequently improves the anonymity level.

7.3 Summary

Several experiments on the identification of the multilayer-encryption anonymity networks were implemented on this chapter. At first, the infrastructure of the Tor network (Circuits, Cells, Nodes) was used to collect data and extract information about the application type running within the encrypted Tor's traffic. Then, the achieved results were compared with the results of using the flow analysis to identify applications on Tor without the need to have access to the Tor resources (Nodes). Both approaches showed a high performance while the flow analysis does not require running a node to extract the needed information to identify the applications.

On the other hand, the experiments included studying the influence of sharing the users' bandwidth on the I2P networks at the level of anonymity the I2P network provides. The experiments included studying user and application profiling when bandwidth sharing is enabled and when it is disabled. After collecting the data on the I2P network for three selected applications for three users, flow analysis was used to analyze the collected data for both the user and application profiling. Moreover, an unsupervised learning algorithm (SOM) was employed to understand the behaviour of the Tunnels on the I2P networks when running different applications. The experiments showed that the effect of bandwidth sharing on the anonymity level of the I2P network is clear.

Chapter 8

Traffic Flow Analysis of Obfuscated Traffic

In this chapter flow analysis will be used to explore the behaviour of Tor and JonDonym when anti-censorship techniques are used with these tools. The I2P network does not implement such techniques in their network yet, so the analysis will include Tor and JonDonym only.

8.1 Tor Pluggable Transports

Pluggable transport systems [102] work to provide access to the Tor network in adversarial (censorship) environments. Most of the pluggable transport tools concentrate on hiding the content of the packets in a way that makes it hard for the adversaries when using deep packet inspection (DPI) to detect the connection to the bridges. But DPI is not the only method used to detect Tor traffic. Active probing and flow analysis are some of the other popular methods used to detect Tor traffic. In this analysis the flow analysis technique is used to explore the resistance of Tor pluggable transports against such techniques.

8.1.1 Data Collection

In this experiment, four virtual machines and one Ubuntu Desktop 12.04 were configured to collect Tor pluggable transport data. All the machines were configured to use one pluggable transport at a time to connect to the Tor network. The traffic data was collected from these five machines. Once the machines are connected to the Tor network, an automated script starts to browse different websites and then closes the connection after the browsing (or watching the videos, etc.) activities are completed. This process repeats until a sufficient amount of data is collected.

Given that pluggable transports hide the Tor traffic using different protocols, the data included HTTP, HTTPS, Secure SHell (SSH), BitTorrent (BT) and Encrypted BitTorrent traffic as background traffic. Table 8.1 and Table 8.2 present the number of flows for the background traffic and the pluggable transports, respectively. All of the Tor pluggable transport data generated and captured in this thesis is made publicly available for the research community at large [5].

	Background Traffic				
Type	HTTP	HTTPS	SSH	BT	Encrypted BT
Number of flows	182,725	8,058	54,214	116,440	198,302
Total	**559,739**				

Table 8.1: Total number of flows of the background traffic.

	Tor Pluggable Transports Traffic				
Type	Obfs3	FTE	Scramblesuit	Meek	Flashproxy
Number of flows	15,356	106,549	16,953	43,152	172,331
Total	**354,341**				

Table 8.2: Total number of flows of the pluggable transport traffic.

What follows details the experiments for each pluggable transport and the amount of data collected.

8.1.1.1 Obfs3 Traffic

The data for the Obfs3 bridge connection has been collected from connections to two bridges. The first bridge was configured by using the recommended bridge setting in the Tor browser (Obfs3). The port used in this bridge was port 80 (one of the well-known ports assigned for HTTP). Even though the flow characteristics do not depend on the port number to identify the type of protocol used in the connection, HTTP traffic is included in the background traffic to compare the ability of the classifier to distinguish between two different applications while both use the same port number.

The second bridge was configured by running a node as a bridge which could accept Obfs3 connections. Then four virtual machines running Ubuntu Desktop 12.04 were configured to use this Obfs3 bridge to connect to the Tor network. The port number used was a dynamic port number. The total amount of Obfs3 traffic captured in these experiments was 20 GB with 16,953 flows.

8.1.1.2 FTE Traffic

The FTE data was collected from five machines with Ubuntu Desktop 12.04 as the operating system. Four of them were running virtual machines. The data was gathered via connecting to five different FTE servers. The total amount of FTE traffic collected was 23 GB. The number of collected flows was 106,549.

8.1.1.3 Scramblesuit Traffic

In addition to the active probing of DPI resistance, Scramblesuit is designed to resist flow analysis by generating different flows for every Scramblesuit server. For this reason, the data was collected from multiple Scramblesuit servers to generate different flows. By using the bridge database, Scramblesuit data was collected by connecting to 22 different Scramblesuit servers. The importance of having different servers is to have a variety of behaviours based on the design of Scramblesuit which changes the server flow for every server. The total number of flows collected from these 22 servers was 10,649. The total amount of Scramblesuit traffic collected was 22 GB.

8.1.1.4 Meek Traffic

Meek makes connections with popular websites which provide services used by a wide range of users. These services include Google, Amazon and Azure. For example, when Google is used as the front domain for Meek, then the multiple addresses which appear with this setting all belong to Google. In these experiments, the total number of flows was 43,152. The data size was 22 GB.

8.1.1.5 Flashproxy Traffic

In Tor it is normal for the user to start the connection to the bridge. However, in Flashproxy, it is the other way around; the Tor user will receive connections from the visitors of the Flashproxy-supported websites. This requires that the user has the ability or the access to do port forwarding if he/she is behind a Network address translation (NAT) or has an open port configured to listen for incoming connections. The number of connections is high compared to the other pluggable transports. The total number of Flashproxy flows was 172,331. The data size was 11 GB.

8.1.1.6 Other Traffic

Pluggable transports are used by Tor to obfuscate Tor traffic in different flavours. To study the efficiency of these tools, pluggable transport traffic should be compared to the flavour of the traffic they are trying to mimic and with different types of encrypted traffic. Thus, five different types of traffic were added as the background non-Tor (normal) traffic as follows: 26 GB of peer-to-peer BitTorrent traffic, 24 GB of encrypted BitTorrent traffic, 29 GB of SSH traffic, 1 GB of HTTPS (SSL) traffic and 0.5 GB of HTTP traffic.

8.1.2 Pluggable Transport Flow Analysis

Three different forms of the collected data sets were generated from the flows of the pluggable transports and the other traffic (background) on the network. The first data set contains all the instances (914,080 instances) with all the features (65 features) and it is labelled (ground truth) using ten classes. The experiments on this data set was performed by splitting the instances into training and testing instances and by using a cross-validation technique as well.

The second data set contains all the instances and the labels of the first data set, but in this case, a smaller number of features is used (three), to represent the data. The third data set contains all of the instances and features of the first data set, but in this case the data set was labelled using only two classes, namely Tor and non-Tor. What follows details the three forms of data set.

8.1.2.1 Split and Cross-Validation Analysis

The first data set will be analyzed by first splitting the data into 66% for training and 34% for testing. Then, 10-Fold cross-validation will be used on the same data set as indicated below.

Splitting the Data into 66% Training / 34% Testing The data is split into a training set (2/3) and a testing set (1/3). The training set is 66% of the whole data. The testing set is the remaining instances. Thus, the goal of using this approach is to investigate whether the classifier would be able to learn the properties of each of the ten classes on the training set only. Then the classifier is tested as to how well it learned the content of the unseen test data set (the set not seen during training). When the C4.5 traffic classification system is used on this data set, the percentage of correctly classified instances is 97%. The performance measurements are shown in Table 8.3.

	Class	**TP Rate %**	**FP Rate %**	**Precision %**	**F-Measure %**
Background Traffic	HTTP	99	0.1	99	99
	HTTPS	94	0.1	94	94
	SSH	99	0	99	99
	BT	94	2.6	80	88
	BTecr	89	0.9	96	92
Pluggable Transport Traffic	FTE	99	0.1	99	99
	Scramblesuit	98	0.1	92	95
	Meek	99	0	99	99
	Flashproxy	99	0.1	99	99
	Obfs3	99	0	99	99
Overall Correctly-Classified Instances	**97%**				

Table 8.3: Results on the first data set using the splitting technique.

10-Fold Cross Validation In this case the classifiers are evaluated using only the 10-fold cross validation technique on the first data set. The detailed results of this approach are shown in Table 8.4. Figure 8.1 shows the F-Measure of the 10-fold

	Class	**TP Rate %**	**FP Rate %**	**Precision %**	**F-Measure %**
Background Traffic	HTTP	99	0.1	99	99
	HTTPS	94	0	95	95
	SSH	99	0	99	99
	BT	94	2.5	84	89
	BTecr	89	0.9	96	92
Pluggable Transports Traffic	FTE	99	0	99	99
	Scramblesuit	98	0.1	92	95
	Meek	99	0	99	99
	Flashproxy	99	0.1	99	99
	Obfs3	99	0	99	99
Overall Correctly Classified Instances	**97%**				

Table 8.4: Results on the first data set using the 10-fold cross validation technique.

cross validation technique and the split technique for the background traffic and the

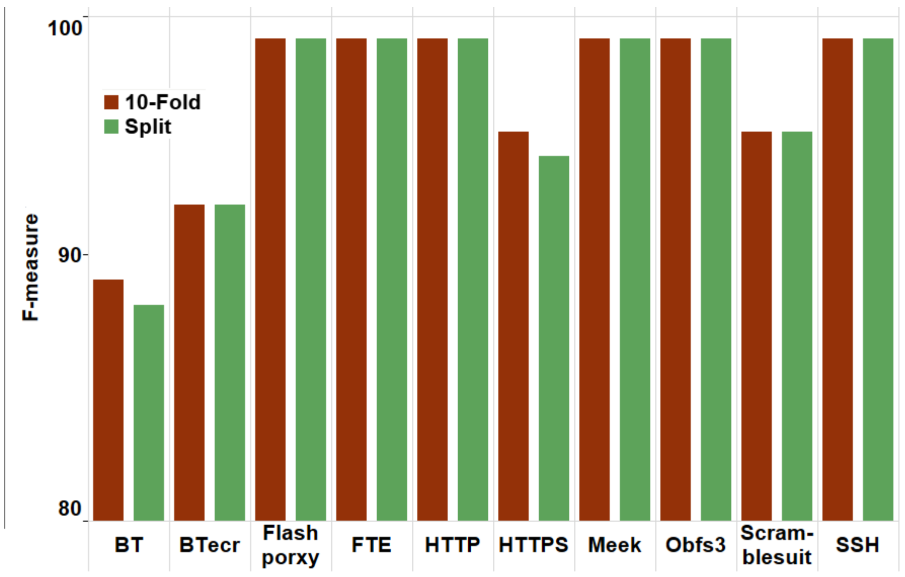


Figure 8.1: Comparison of F-Measure Values Between the 10-Fold Cross Validation Technique and the Splitting Technique on the First Dataset

pluggable transport traffic. The lower F-measure values in these experiments indicate that the BT, the encrypted BT, the HTTPS and the Scramblesuit traffic flows are more difficult to identify compared to the other applications. The traffic flows of these applications have differences between the 10-fold cross validation results and the split results. The traffic of the other applications showed consistent results between the 10-fold and the split technique. This indicates that the training set contains instances that did not appear on the testing test and vice versa when using the split technique for these applications.

8.1.2.2 Reduced Number of Features

The number of features used to represent the traffic in the above experiments was 65. These include all the relevant features of Tranalyzer for the purpose of this analysis, i.e. the analysis of Tor pluggable transport flows. The non-relevant features (such as ICMP and VLAN) have been excluded from the flow analysis. However, this large number of features, when combined with the number of instances (914,080), makes the classification computationally costly. So a feature selection technique (Ranker from WEKA) was used to reduce the number of features. Ranker is a search method for arranging the features from the most important to the least important. Based on the results of Ranker, the most important features were selected (Duration, Number of Bytes Sent and Maximum Packet Size). The performance of the classifiers using only these three features is shown in Table 8.5.

	Class	**TP Rate %**	**FP Rate %**	**Precision %**	**F-Measure %**
Background Traffic	HTTP	97.4	0.8	96.9	97.1
	HTTPS	79.1	0.1	85.2	82
	SSH	98.7	0	99.9	99.3
	BT	82	2.7	81.7	81.9
	BTecr	86.3	3	88.8	87.5
Pluggable Transport Traffic	FTE	97.7	0.3	97.4	97.6
	Scramblesuit	84.6	0.1	92	88.1
	Meek	95	0.4	91.7	93.3
	Flashproxy	97.8	1.2	95	96.4
	Obfs3	98.3	0	98.8	98.6
Overall Correctly-Classified Instances	**93%**				

Table 8.5: Results using three features only.

8.1.2.3 Binary Classification

In this experiment all the pluggable transports in the data set are labelled as Tor and all the other traffic labelled as non-Tor. This brings up the total number of Tor traffic to 354,341 instances and for the non-Tor traffic to 559,739 instances. In this case the percentage of instances classified correctly is increased to almost 100%. The results of this experiment are presented in Table 8.6.

Class	**TP Rate %**	**FP Rate %**	**Precision %**	**F-Measure %**
Tor	99.7	0.3	99.5	99.6
NonTor	99.7	0.3	99.8	99.8
Overall Correctly-Classified Instances	**99.7%**			

Table 8.6: Results for binary classification.

In this case all the background instances are put together into one group and all the pluggable transport instances are put into another group (Figure 8.2). The x-axis represents the duration and the y-axis represents the summation of the average

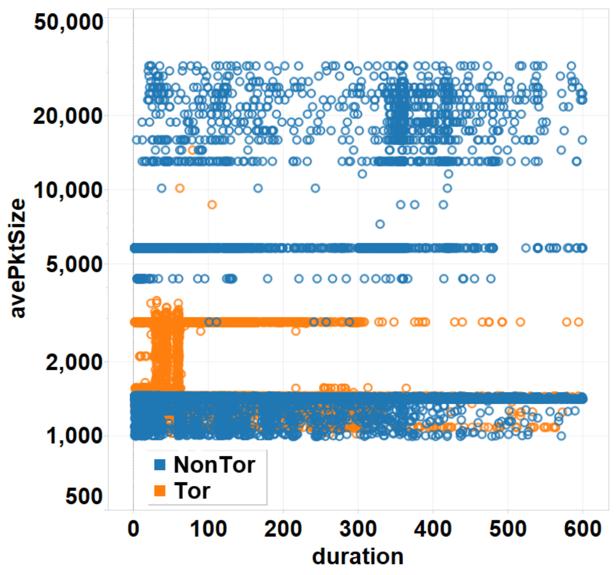


Figure 8.2: The Distribution of Average Packet Size

packet size for every flow in each group. The y-axis is the aggregation of the average packet size with time. The number of instances is large and sampled from all the data sets. The duration is limited to ten minutes. The average packet size for an individual flow of pluggable transports lies in the middle area of the graph with few outliers. The average packet size of the other (background) traffic is scattered all over the graph. This relationship between the data points indicates that the average value of the data transmitted by Tor can be used to distinguish Tor from non-Tor traffic. It seems as though the tools that change the packet length (such as Scramblesuit) do not change the average amount of the data transmitted in a way that makes it indistinguishable. This is because the amount of padding used to change the length of the packets is small. In return, this does not change completely the total amount of Tor data transferred compared to the non-Tor data. The use of this phenomenon is important because hiding the 512-bytes cells by padding still does not change the total amount of data transferred.

8.1.3 Discussion

The results above show that an attacker who has the means to perform flow analysis against Tor could achieve a very high performance in detecting the Tor pluggable transports under all the evaluated conditions.

The analysis shows that when the feature selection is used for the flow analysis of the pluggable transports one can identify the prominent features that describe the pluggable transport behaviour. In this experiments the duration, the number of bytes sent and the maximum packet size seem to be the most important features.

The pluggable transports are designed to hide or obfuscate the content of the Tor connection, not the flow. Thus, the flow analysis could identify Tor traffic even with the existence of such obfuscation techniques. Even though Scramblesuit can change the distribution of the packet length and the inter arrival time of the traffic, the flow analysis could still profile different Tor traffic behaviours with an 85% true positive rate (just by using the aforementioned three features). If more features are used then the detection rate goes up to 98%. Additionally, under other obfuscation techniques, Tor behaviour classification could even reach a 98% true positive rate.

Having said this, the detection rate might change based on the background traffic characteristics. For example, FTE obfuscates by making the regex of the Tor encrypted traffic look like the regex of HTTPS traffic. This makes the HTTPS traffic an important factor in the training data set.

Furthermore, the feature selection indicates that the packet size, the number of bytes sent and the maximum packet size are the three primary features that profile the pluggable transports. In fact, this observation is consistent with how Tor works as well. For example, Obfs3 does not change the packet size nor the inter arrival time. This makes the packet size an important feature that profiles the Obfs3 traffic. Obfuscating the TLS handshake has nothing to do with the amount of data transferred between the user and the pluggable transport server. The duration of the connection is one of the features that could profile Tor traffic with or without the pluggable transports. Regular connections stay active for a shorter time than the duration that a Tor user connection does.

Moreover, based on the observations on the network traffic, Flashproxy changes the connection to the user based on the IP addresses of the Flashproxy-supported websites. This causes the importance of the duration in the detection of Flashproxy to become less compared to the importance of the packet size. Scramblesuit changes the packet size (to a certain level) but the duration is still a factor. In summary, the evaluations seem to indicate that pluggable transports are designed mainly to evade deep packet inspection (DPI). However, pluggable transports need to consider the flow analysis in their design to improve the obfuscation of the Tor traffic.

8.2 JonDonym Traffic Forwarding

The obfuscation technique on the JonDonym network is different from the one on the Tor network. The JonDo client software includes an option for the user to enable the obfuscation; while on the Tor network, the obfuscation tools are separated from the client and are needed to be configured first. In addition, the obfuscation on JonDonym counts on forwarding the connection to the network to another JonDonym user instead of connecting directly to the network to obfuscate the connection to the JonDonym network.

JonDonym has two forwarding options: TCP/IP forwarding and Skype forwarding. To study the flow behaviour of the forwarding techniques used on JonDonym, the JonDonym traffic without any obfuscation will be studied first then will be compared with the TCP/IP and Skype forwarders indicated below.

8.2.1 JonDonym Flow Behaviour

The JonDonym data employed in this analysis is the JonDonym part from the Anon17 data set. The JonDonym data is collected from three machines at the NIMS lab by connecting to all the free cascades on the JonDonym network.

For a background traffic, the LBNL/ICSI data set [77] was employed as the background traffic. It contains network traces collected from more than 100 hours of activities for several thousands of hosts. The data size is 11 GB. The data is publicly available in a Packet Capture (PCAP) form. The data is distributed over several small PCAP files. For this analysis a total of 211,370 flows was extracted from approximately 1.5 GB of data. Table 8.7 shows the result for the JonDonym traffic with the background traffic. The result shows that JonDonym flows can be distinguished from background traces with a high accuracy level.

The next part of the analysis is performed by labelling the background data to application/protocol names based on the port number. The background data contains a vast number of applications and protocols. Some of the applications or the protocols appear just a few times while others have a high number of appearances in

	TP Rate	**FP Rate**	**Precision**	**F-Measure**
JonDonym	0.997	0	1	0.998
Background LBNL/ICSI	1	0.003	1	1
Accuracy	**99.99%**			

Table 8.7: JonDonym flow analysis results.

the data. Consequently, the top appeared applications in the data are labelled with their application name, the rest are labelled as "Others". Thus, instead of having one class (Background), the data will have 12 classes (HTTP, HTTPS, IMAPS, SNMP, NETBIOS-SSN, DNS, POP3, LPD, EPMAP, SMTP, SSH and Other). Table 8.8 shows the results of the application and JonDonym analysis. The results dropped by 2% compared with the previous analysis in which all the background traffic was grouped into one class.

	TP Rate	**FP Rate**	**Precision**	**F-Measure**
HTTP	0.986	0.013	0.981	0.984
HTTPS	0.897	0.004	0.919	0.908
IMAPS	0.88	0.001	0.900	0.888
SNMP	0.998	0.000	0.996	0.997
NETBIOS-SSN	0.974	0.002	0.971	0.972
DNS	0.997	0.000	0.998	0.998
POP3	0.982	0.000	0.969	0.975
LPD	0.998	0.000	0.998	0.998
EPMAP	0.993	0.000	0.988	0.990
SMTP	0.948	0.000	0.961	0.955
SSH	0.455	0.000	0.613	0.522
OTHER	0.973	0.006	0.976	0.974
JonDonym	0.999	0.000	1.000	1.000
Accuracy	**97.99%**			

Table 8.8: Results of the applications and JonDonym analysis.

8.2.2 TCP/IP and Skype Forwarding

JonDonym's users can use the TCP/IP forwarding option to connect to the network through other users of the network. The user needs to solve a strong CAPTCHA (Completely Automated Public Turing Test to tell Computers and Humans Apart) before establishing the connection. The other option is to use the Skype forwarding option. The user needs to have a Skype account and log in. Then after logging in, the user selects Skype as the forwarding option. The JonDo client sends the connection to Skype to forward the connection to the JonDonym network. Table 8.9 shows the results of JonDonym, Skype Forwarder, TCP/IP Forwarder and the background applications. The results show a high performance for the classifier.

	TP Rate	**FP Rate**	**Precision**	**F-Measure**
HTTP	0.986	0.012	0.982	0.984
HTTPS	0.900	0.004	0.920	0.910
IMAPS	0.888	0.001	0.906	0.897
SNMP	0.997	0.000	0.997	0.997
NETBIOS-SSN	0.973	0.002	0.970	0.972
DNS	0.998	0.000	0.998	0.998
POP3	0.979	0.000	0.977	0.978
LPD	0.998	0.000	0.998	0.998
EPMAP	0.992	0.000	0.991	0.992
SMTP	0.956	0.000	0.958	0.957
SSH	0.437	0.000	0.591	0.502
OTHER	0.974	0.006	0.977	0.975
JonDonym	0.999	0.000	1.000	1.000
SKYPEFWD	0.992	0.000	0.983	0.988
TCPIPFWD	0.988	0.000	0.996	0.992
Accuracy	**98.04%**			

Table 8.9: Results of Skype and TCP/IP forwarder flow analysis.

8.3 Summary

Flow analysis was used in this chapter to identify the obfuscated traffic on the multi-layer
-encryption anonymity networks. On the Tor network there are many tools used for the obfuscation of the Tor traffic. The JonDonym network has two obfuscations which could be used to forward the connection to the JonDonym network. The I2P network does not have any obfuscation techniques yet. The results of using flow analysis to identify obfuscated traffic in multilayer-encryption anonymity networks demonstrated a high performance.

JonDonym uses two traffic forwarding options to obfuscate the connection to the JonDonym network: TCP/IP and Skype. On the other hand, the obfuscation options on the Tor networks have more choices. In addition, the Tor obfuscation options are more reliable compared to JonDonym. In order to use the forwarder in JonDonym, the user first has to solve a strong CAPTCHA. Moreover, the connection is not guaranteed even if the CAPTCHA is solved correctly using the Skype option. Even after solving the CAPTCHA, the connection faild and data could not be transferred. The TCP/IP option for JonDonym also has a CAPTCHA which needs to be solved but in these experiments this option seemed to be a more reliable way to forward the data.

Chapter 9

Packet Momentum

Identifying the traces of anonymity networks is a challenging task. One of the important reasons for this is that these networks are designed to provide some level of privacy. This in turn results in hiding or changing the traffic. Even though the anonymity networks do not hide the users' connections to the network, they do change the default form of some activities such as browsing in one way or another. The users who employ the anonymity network to surf websites on the Internet do not necessarily use the standard HTTP port (port 80) for this purpose. Moreover, HTTP requests are wrapped within the multilayer of encryptions that the anonymity networks use to cover them. Additionally, some countries' censorships block anonymity networks which results in the anonymity network investigating and adapting more methods such as obfuscation techniques to hide the connection and bypass the blockage.

DPI, active probing and flow analysis are examples of some of the few techniques used to identify anonymity networks. These methods have limitations: for example, the disadvantage of using DPI is that encryption makes the packets opaque so DPI will be irrelevant/useless. The IP addresses of the anonymity networks' routers could be used to identify such networks. Using the obfuscation techniques or the bridges will reduce the effectiveness of such method. Some of the obfuscation techniques are designed to resist active probing as well. Flow analysis has satisfactory results in identifying anonymity networks even with the existence of the obfuscation techniques. However, flow analysis has its limitations as well. One of the obstacles in using flow analysis is the computational cost: the higher the amount of data traffic and the number of features, the higher the cost. In large scale network this requires high CPU resources and time. Consequently, using flow analysis for identifying anonymity networks inline (i.e. classifying the flow while it is still active) is a difficult challenge and could be affordable only at the censorship level. At the same time, the existence of this vast number of applications these days might increase the false alarm rate which might cause normal traffic to be identified as anonymity traffic. This was a big motivation for seeking the possibility of improving flow analysis to give better results in less time.

9.1 Packet Behaviour in Anonymity Networks

The anonymity networks have their way in dealing with traffic within the network. The unit that is used on the Tor network is the cell. The size of the cell is 512 bytes. The Tor user (client) installed on the client's machine divides data sent by the client to the Tor network into fixed-size cells. When these cells arrive at the transport layer and beyond they are considered to be like any other normal traffic according to the protocol used. The cells will be packed inside packets and traverse like other packets on the network. There are research papers on studying the link between the packets and the cells. The point here is that the fixed-size cells have a kind of relationship with the number and size of the packets. Obfs3 is one of the obfuscation techniques used by the Tor network which change the data into unknown random strings. At the same time, Obfs3 does not change packet timing or volume. For example, downloading files while using Obfs3 does not change the increase in the traffic volume due to downloading files as compared with browsing only. Using Obfs3 is useful to hide the identity of the users or the communication parties. Consequently, the pattern in the size of the data could lead to a link between the data and the user.

On the other hand, how applications or protocols work in general has a sort of repetition in the data flow. For example, accessing a web server starts by sending a request to the web server then waiting for the server to reply while data travels back and forth to the web server. Whenever another access to the web server is taking place the process is repeated. This type of pattern exists to a certain level in many applications. On the anonymity network the pattern is used as well but it is more complex. The obfuscation techniques aim to make identification of such patterns a difficult task. In addition, on all anonymity networks the anonymity level is increased when more users are using it. The reason for that is the difficulty of finding patterns when the number of users is large. Also, on some of these anonymity networks users of the network relay data of other users on the networks in addition to their own data and messages related to managing the connection to the network.

On the anonymity networks the users' data contains mostly overhead due to encryption operations and managing the connection to the network. This requires that the users maintain a connection to the network where packets with volume and sequence keep going back and forth between the users and the network. No matter what obfuscation techniques are used to avoid inspecting packets, packet flows cannot be hidden. Scramblesuit alters the flow between the user and the Scramblesuit server. This change in the flow (as shown in Chapter 8) produces a different pattern from the original flow, but it is still identifiable.

Another aspect of packets' behaviour is the direction of dialog in the process of communication. For example, when watching a video stream the direction of the data is mostly from the server to the user with little from the user to the server. On PTP, this behaviour is different. The direction of the data takes a different form or a more balanced shape based on the application and the user's activity. The point here is that the application has its influence on the direction that the data will take back and forth.

9.2 Proposed Features

Assume that there are two communications taking place as shown on Figure 9.1 and Figure 9.2. Case 1 represents communication between user A and user B for application 1. Case 2 represents communication for application 2. In Case 1 user A sent 5 packets to user B which contain 10 bytes in total. User B sent 4 packets to user A with 6 bytes in total. The numbers used in this example are simplified for the purpose of explaining the features. In Case 2 (another application) user A sent 5 packets as well to user B with 10 bytes in total. User B sent back to user A 4 packets with 6 bytes in total. Based on these values, the total number of packets in both directions, the total bytes sent, the average payload size and the average inter arrival time for the two different applications are exactly the same. The average payload size in Case 1 (A) and Case 2 (A) is identical (equal to 2 bytes). The total bytes sent is identical as well (equal to 10). The average inter arrival time between the packets in Case 1 (A) is equal to 3. The average inter arrival time between the packets in Case 2 (A) is equal to 3 as well even though the timing in both cases is totally different.

Case 1						
A > B	Size	1	2	2	2	3
	Time	1	4	10	11	13
B > A	Size	1	2	1	2	
	Time	2	5	8	14	

Case 2						
A > B	Size	4	0	4	0	2
	Time	2	6	7	8	14
B > A	Size	1	2	1	2	
	Time	1	5	11	13	

Figure 9.1: Example of Packet Size and Inter Arrival Time Features

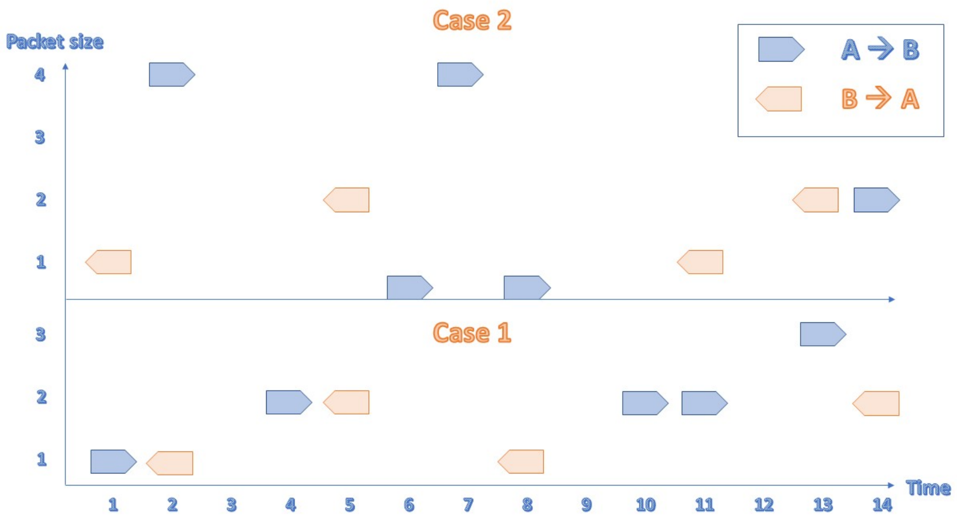


Figure 9.2: Packets Exchange in Case 1 and Case 2

The same applied to B in both cases where both have the same average inter arrival time which is 4. Consequently, in this particular situation, the average payload size, the number of packets, the average inter arrival time and the number of sent bytes are not enough to distinguish between Case 1 and Case 2. It might be hard in real life to have size, inter arrival time and the number of packets to be identical like this but the point is that even using these different important features might not give enough information to have a clear picture about two different applications.

Features generated by any flow exporter tool have different importance based on the types of applications or traffic under analysis. That means if the features are ranked top-down for one set of applications, then this rank is different for another set of applications. For example, in Chapter 8 (as shown on Section 8.1.2.2) the duration, the number of bytes sent and the maximum packet size were the most important features for describing the data under study. The importance of these features is determined based on a features ranking method called "Ranker" in WEKA. This method could use the information gain to decide the importance of the features. In addition, a decision tree could visualize the importance of the features based on the features information gain. When the idea of information gain is applied to the example given here and according to the value shown in Figure 9.1, then the information gain for the features (Average Inter Arrival Time, Average Payload Size, Total bytes sent and Number of packets) could not provide any difference for Case 1 and Case 2. Consequently, in this situation these features do not provide useful information for classifying the two applications. The features described below will address this difficult classification situation.

9.2.1 Maximum Packet Size

The highest value for the packet size in Case 1 (A) is 3 and 2 in Case 1 (B). In Case 2 (A) the maximum packet size is 4 and 2 in Case 2 (B). It can be seen that the maximum packet size will help to show the difference between Case 1 (A) and Case 2 (A) even when both have the same number of packets and the same average packet size. At the same time, however, the maximum packet size alone is not enough to distinguish between Case 1 (B) and Case 2 (B) where both have exactly the same size of packets.

9.2.2 Frequency of Maximum Packet Size

The maximum packet size is a useful feature but it can give more information when combined with the measurement of the influence of the maximum packet size. In Case 1 (A) the maximum packet size is 3 bytes but it appears only once in the communication. In Case 2 (A) the maximum packet size is 4 bytes and it appears twice. By adding the frequency of the maximum packet size, it could describe how strong the effect of this maximum packet size is on the communication.

9.2.3 Second Maximum Packet Size

In Case 1 (A) the highest packet size is 3 bytes and it appears only once, while the 2-byte packet size appears three times. Consequently, including both the highest packet size and the second highest packet size can give more information about the communication.

9.2.4 Second Maximum Packet Size Frequency

The highest packet size in Case 1 (A) equals 3 bytes and the second maximum packet size equals 2 bytes. However, the maximum packet size appears only once while the second maxim packet size appears three times. Consequently, adding the frequency of the second maximum packet size will show the repetition factor.

9.2.5 Packet Sequence

The 4-byte packet in Case 2 (A), which is the maximum packet size, appears twice but not in a sequential order. By contrast, the 2-byte packet, which is the maximum packet size in Case 1 (A), appears three times in a row. The maximum packet size or the second maximum packet size will not show this information. The maximum packet length or the average packet length will not see any difference whether the packets arrive in sequence or not. Consequently, the packet sequence here will describe how the packets arrive in the communication between A and B: it will give more value to the subsequent packets. Figure 9.3 and Figure 9.4 show how to present the packet sequence in both cases. The packets will have positive value (1) every time a packet goes from A to B in a cumulative way. If three packets go from A to B in sequence before any packets come from B, then this will add up to (3). Packets from B to A will take the packet sequence down one step, and so on. The packet sequence will be calculated as the summation of all cumulative values. The packet sequence for Case 1 will be 5 and 3 for Case 2. More on calculating the packet sequence can be found in Appendix A. The Pseudo code for calculating the Packet Sequence is shown in Algorithm 3.

Algorithm 3 : Calculation of the Packet Sequence.

```
inputs n: total number of packets in the direction of A to B and packets in the
direction of B to A.

if the first packet is from A to B then
    $SequenceValue_1 = 1$
else
    $SequenceValue_1 = -1$
end if

$PacketSequence = SequenceValue_1$

for i = 2 to n do
    if the current packet is from B to A then
        $SequenceValue_n = SequenceValue_{n-1} - 1$
    else
        $SequenceValue_n = SequenceValue_{n-1} + 1$
    end if

    $PacketSequence = PacketSequence + SequenceValue_n$

end for
```

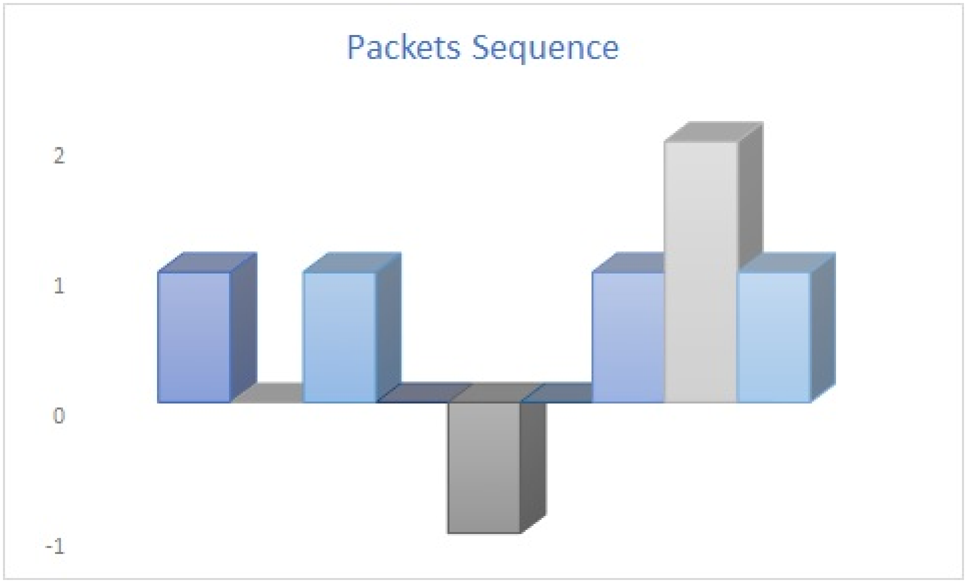


Figure 9.3: Case 1 Packet Sequence

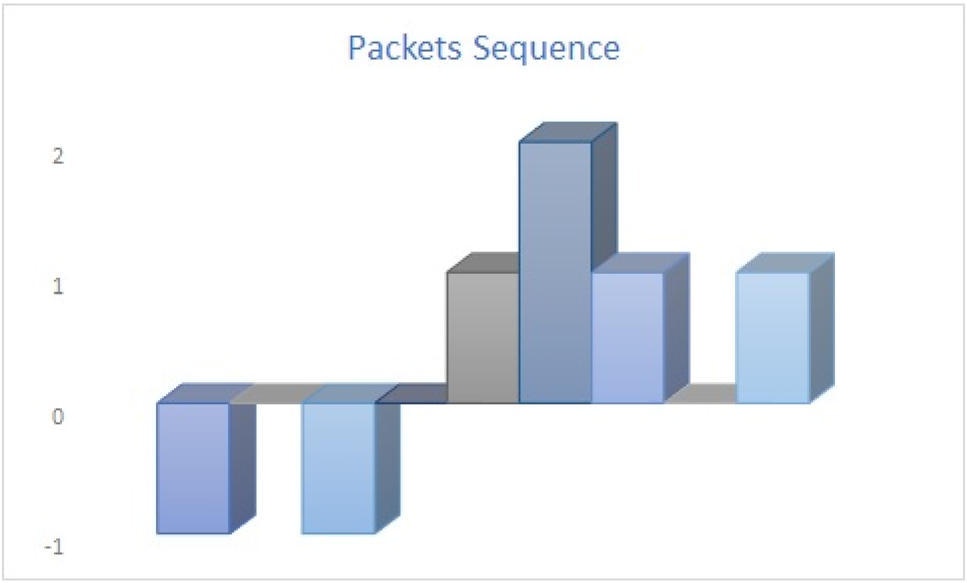


Figure 9.4: Case 2 Packet Sequence

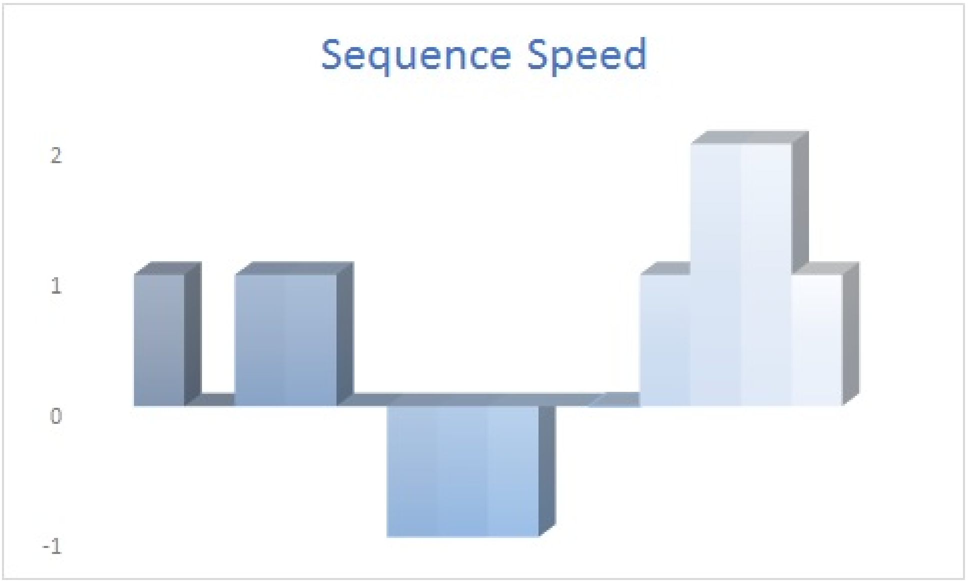


Figure 9.5: Sequence Speed for Case 1

9.2.6 Sequence Speed

The packet sequence considers the sequence of the packets between A and B, but it does not take into consideration how fast or slow the packets' flows stay in one direction before the sequence changes. If, for example, three packets arrive in sequence within three minutes, the sequence will have the same value even if these three packets arrive in sequence within 3 seconds. Figure 9.5 and Figure 9.6 show how to include the inter arrival time in the calculation to measure the sequence speed. In this way the time needed to change the direction from (A to B) to (B to A) or the opposite will be taken into consideration. The sequence speed reflects the change in direction with respect to the time taken to change the direction and for how long this change stays. Appendix A shows how to calculate the sequence speed for Case 1. The Pseudo code for calculating the Sequence Speed is shown in Algorithm 4.

9.2.7 Packet Momentum

Assume three packets arrive from A to B within 3 seconds with 2-, 3- and 4-byte packets in the first case and another three packets arrive from A to B within 3 seconds (same timing also) 6-, 8- and 9-byte packets in the second case. The sequence speed for both cases will be the same (when the inter arrival time is the same also).

Algorithm 4 : Calculation of the sequence speed.

```
inputs n: total number of packets in the direction of A to B and packets in the direction of B to A.
$IAT_n$: inter arrival time of packet n.
$T_n$: Time of packet n.

if the first packet is from A to B then
    $SequenceValue_1 = 1$
else
    $SequenceValue_1 = -1$
end if

$IAT_1 = T_1$
$SequenceSpeedValue_1 = SequenceValue_1 * IAT_1$
$SequenceSpeed = SequenceSpeedValue_1$

for i = 2 to n do
    if the current packet is from B to A then
        $SequenceValue_n = SequenceValue_{n-1} - 1$
    else
        $SequenceValue_n = SequenceValue_{n-1} + 1$
    end if

    $IAT_n = T_n - T_{n-1}$
    $SequenceSpeedValue_n = SequenceValue_n * IAT_n$
    $SequenceSpeed = SequenceSpeed + SequenceSpeedValue_n$

end for
```

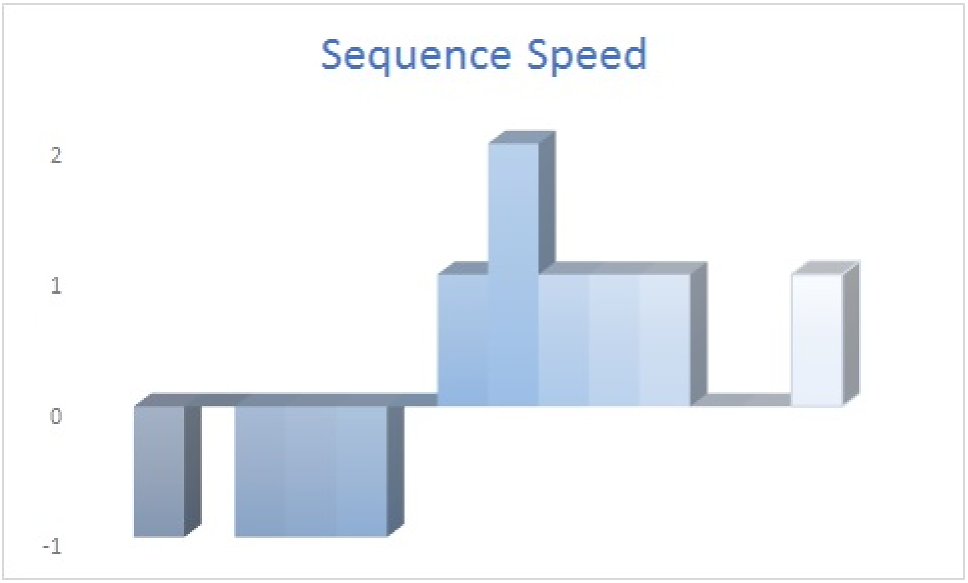


Figure 9.6: Sequence Speed for Case 2

The packet momentum is what takes into consideration the packet size in addition to the sequence and speed. Figure 9.7 and Figure 9.8 show the packet momentum for Case 1 and Case 2. The packet momentum distinguishes between packets that have the same inter arrival time and the same sequence. The packet momentum is shown in Algorithm 5. The calculation of the packet momentum for Case 1 is explained in Appendix A. The pseudo code for all the features of packet momentum is shown in Appendix B.

9.3 Traffic Analysis Using Packet Momentum

The strength of packet momentum analysis relies on two factors: a small number of features and a small number of packets required for the analysis. The number of features has an influence on the calculation cost: the higher the number, the higher the calculation cost. The number of packets needed to analyze the traffic affects the speed of classifying it. For example, the duration of the connection, one of the features employed in the flow analysis in Chapter 7 and 8, requires that the connection be terminated in order to calculate the duration of the connection. In the Tor pluggable transport analysis, the duration was one of the more important features in the analysis.

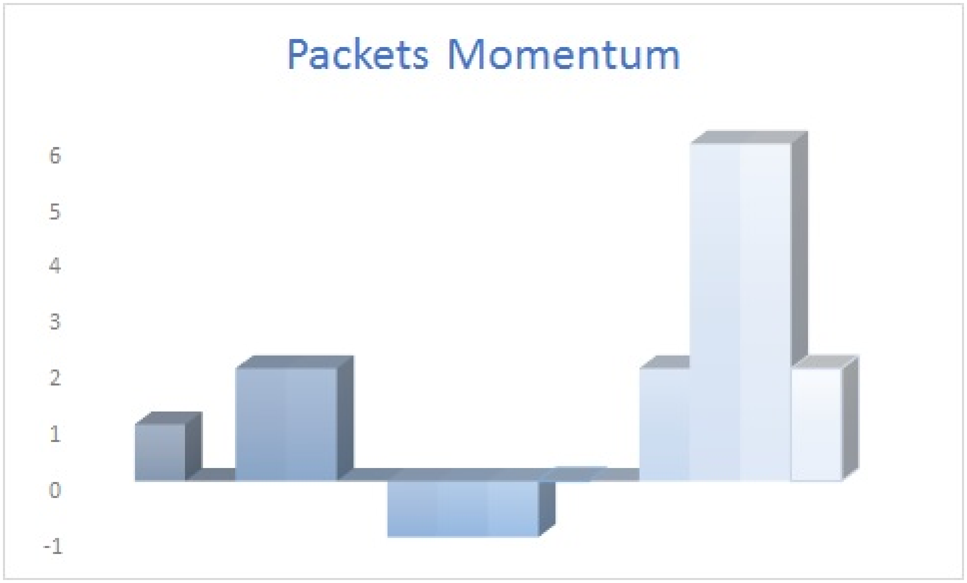


Figure 9.7: Packet Momentum for Case 1

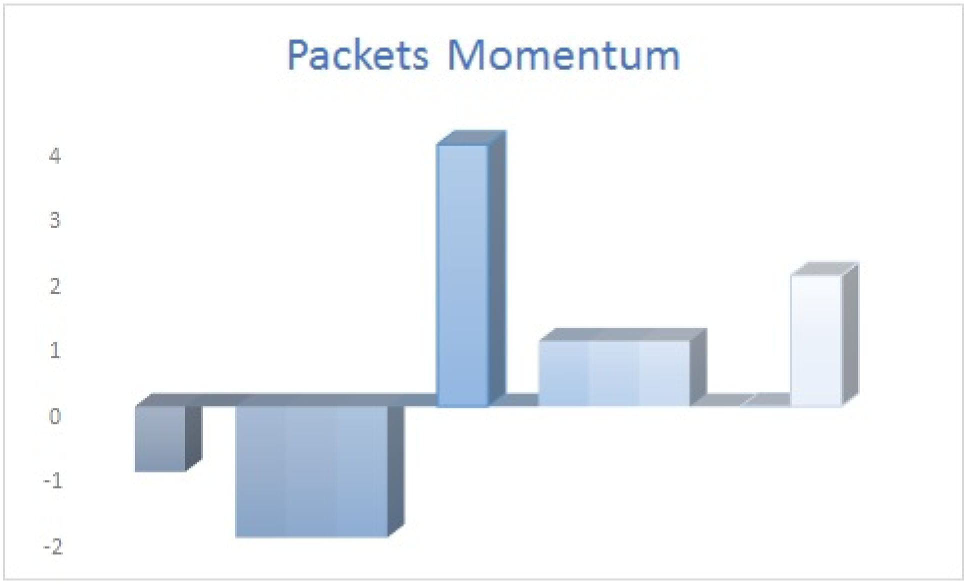


Figure 9.8: Packet Momentum for Case 2

Algorithm 5 : Calculation of the Packet Momentum.

```
inputs n: total number of packets in the direction of A to B and packets in the direction of B to A.
$IAT_n$: inter arrival time of packet n.
$T_n$: Time of packet n.
$S_n$: Size of packet n.
if the first packet is from A to B then
    $SequenceValue_1 = 1$
else
    $SequenceValue_1 = -1$
end if
$PacketSequence = SequenceValue_1$
$IAT_1 = T_1$
$SequenceSpeedValue_1 = SequenceValue_1 * IAT_1$
$SequenceSpeed = SequenceSpeedValue_1$
$PacketMomentumValue_1 = SequenceSpeedValue_1 * S_1$
$PacketMomentum = PacketMomentumValue_1$
for i = 2 to n do
    if the current packet is from B to A then
        $SequenceValue_n = SequenceValue_{n-1} - 1$
    else
        $SequenceValue_n = SequenceValue_{n-1} + 1$
    end if
    $IAT_n = T_n - T_{n-1}$
    $SequenceSpeedValue_n = SequenceValue_n * IAT_n$
    $SequenceSpeed = SequenceSpeed + SequenceSpeedValue_n$
    $PacketSequence = PacketsSequence + SequenceValue_n$
    $PacketMomentumValue_n = SequenceSpeedValue_n * S_n$
    $PacketMomentum = PacketMomentum + PacketMomentumValue_n$
end for
```

The duration of the connection might stay for a brief time or for a very long time. Accordingly, the number of packets in a connection between two parties varies between very low to very large. The packet momentum requires a smaller number of packets for the analysis regardless of the duration of the connection. More details on the number of packets employed in the packet momentum analysis are shown in Section 9.4. The analysis using packet momentum is described below.

9.3.1 Anonymity Network Identifications

A binary classification is used here to identify the anonymity network. The first class is the anonymity network which contains anonymity network traffic from Anon17 all combined into one class. This includes the obfuscated traffic of Tor and JonDonym. The second class is non-Anonymity traffic which is the LBNL/ICSI data set (the same as that employed in Chapter 8) where all traffic is labelled as non-anonymity. The LBNL/ICSI size is 11 GB distributed over several small PCAP files. In total, 211,370 flows were extracted from approximately 1.5 GB. The features employed here are the packet momentum features described previously. The number of packets extracted from each flow and used to calculate the features is 3. By using the C4.5 classifier and 10-fold cross-validation the accuracy was 98.75% as shown in Table 9.1.

	TP Rate	**FP Rate**	**TN Rate**	**FN Rate**
Anonymity Traffic	0.99	0.01	0.99	0.01
Non-Anonymity Traffic	0.99	0.01	0.99	0.01
Accuracy	**98.75%**			

Table 9.1: Results for binary classification using packet momentum.

Table 9.2 shows the accuracy of the classifier when using the packet momentum features to classify the anonymity networks (Tor, JonDonym and I2P) in addition to the obfuscated traffic from Tor and JonDonym. In this case, the anonymity networks and the obfuscated traffic are not grouped into one class but labelled separately. The background traffic is the same LBNL/ICSI data set. The accuracy was 98.73%.

9.3.2 Identification of Applications and Anonymity Networks

Table 9.3 shows the performance of the packet momentum when the background traffic is labelled according to the application or the protocol (same way as in Chapter

	TP Rate	**FP Rate**	**TN Rate**	**FN Rate**
Background Traffic	0.99	0.01	0.99	0.01
I2P	0.98	0	1	0.02
Tor	0.94	0	1	0.06
JonDonym	0.99	0	1	0.01
TCPIPFWD	0.98	0	1	0.02
SKYPEFWD	0.99	0	1	0.01
Flashproxy	0.98	0	1	0.02
FTE	1	0	1	0
Meek	1	0	1	0
Obfs3	0.97	0	1	0.03
Scramblesuit	0.92	0	1	0.08
Accuracy	**98.73%**			

Table 9.2: Results of packet momentum for the obfuscated traffic.

8). The anonymity network traffics and the obfuscated traffics also separated in independent classes. Accordingly, there are 22 classes used in this analysis. The result shows a 97.92% accuracy.

9.4 Packet Momentum Validation

One of the factors which influence the time required to build the training and testing models on a classifier is the number of features. In addition, the number of packets employed in the analysis affects the time required for building the models. Also, the number of packets will affect the calculation time for the features. Consequently, the lower the number of features and packets the better the efficiency of the classifier. The following examines these two factors.

9.4.1 Number of Packets

Table 9.4 shows the accuracy of the C4.5 10-fold cross-validation for the number of packet changes from 1 to 25 packets. The table includes as well the time needed to build the model, the number of leaves on the decision tree and the tree size. The data employed here is the same data used in Section 9.3.2 with a total of 22 classes representing the anonymity networks traffics, the obfuscated traffics and the LBNL/ICSI data set as the background traffic labelled as 12 different classes.

	TP Rate	**FP Rate**	**TN Rate**	**FN Rate**
HTTP	0.97	0.01	0.99	0.03
HTTPS	0.95	0	1	0.05
IMAPS	0.81	0	1	0.19
SNMP	1	0	1	0
NETBIOS-SSN	0.99	0	1	0.01
DNS	1	0	1	0
POP3	0.80	0	1	0.20
LPD	0.98	0	1	0.02
EPMAP	0.99	0	1	0.01
SMTP	0.99	0	1	0.02
SSH	0.43	0	1	0.57
OTHER	0.96	0.01	0.99	0.04
I2P	0.98	0	1	0.02
Tor	0.94	0	1	0.06
JonDonym	0.99	0	1	0.01
TCPIPFWD	0.99	0	1	0.01
SKYPEFWD	0.99	0	1	0.01
Flashproxy	0.98	0	1	0.02
FTE	1	0	1	0
Meek	1	0	1	0
Obfs3	0.97	0	1	0.03
Scramblesuit	0.92	0	1	0.08
Accuracy	**97.92%**			

Table 9.3: Results of applications and anonymity networks analysis.

The results show that packet momentum has the best accuracy with the lowest time to build the model when the number of packets is between 3 and 6. Below and above these two numbers, the accuracy will drop and the time will increase. When the number of packets was one the accuracy dropped to 45.78%. The reason is that the features of packet momentum will not be utilized. The second maximum packet size will always be zero no matter what applications or protocol are analyzed. The frequency of the maximum packet size will always be one for all the classes. When the number of packets is 2, the accuracy improved to 82.53%. Once the number of packets is 3 and above, the accuracy reaches 97%.

Number of Packets	**Accuracy (%)**	**Time (sec)**	**Leaves/Tree**
1	45.8	4.4	183/365
2	82.5	22.53	1568/3135
3	97.9	15.83	1649/3297
4	98.2	20.75	1497/2993
5	98.3	24.01	1524/3047
6	98.1	25.38	1742/3483
7	98	27.31	1763/3525
8	97.9	29.26	2109/4217
9	97.9	30.69	2102/4203
10	97.9	28.98	1956/3911
11	98	28.06	1812/3623
12	98	29.65	1808/3615
13	98	31.07	1902/3803
14	98	31.77	1894/3787
15	98	31.61	1895/3789
16	98	28.27	1868/3735
17	98	28.95	1900/3799
18	98	28.42	1918/3835
19	98	28.57	1947/3893
20	98	27.57	1914/3827
21	98	29.01	1904/3807
22	98	28.54	1982/3963
23	98	31.29	2036/4071
24	98	31.1	2026/4051
25	98	29.55	1999/3997

Table 9.4: Influence of number of packets on the packet momentum.

9.4.2 Number of Features

Based on the results of the number of packets in the previous section which shows that a number between 3 and 6 packets gives the better performance for packet momentum, the number of features will be analyzed for the same range of packets (3-6). WEKA's InfoGianAttributeEval will be used as the method for evaluating the attributes. The search method is the WEKA's Ranker. Table 9.5 shows the ranking of the features when the number of packets is between 3 and 6 packets.

Table 9.6 shows the accuracy and the time needed to build the model when the number of features is reduced. One feature will be removed at a time based on the ranking in Table 9.5. Starting from 11 features until only one feature remains, the

Number of Packets	**Features ranking**
3	5,1,10,11,8,4,9,6,7,3,2
4	5,1,10,11,4,6,8,2,9,7,3
5	1,5,10,2,6,11,4,8,9,3,7
6	5,1,10,6,2,11,4,8,9,7,3

Table 9.5: Features ranking for packets between 3 and 6.

accuracy and the time needed to build the model will be measured for 3, 4, 5 and 6. Mostly, removing a feature from the features list will reduce the time needed to build the model. At the same time the accuracy decreases for each feature removed from the features list. A high drop in the accuracy happens when the number of features is reduced from 3 to 2. The highest drop in the accuracy appears when reducing the number of features to only one feature.

No of Features	**3 Packets**		**4 Packets**		**5 Packets**		**6 Packets**	
	Acc (%)	**Time (sec)**	**Acc (%)**	**Time (sec)**	**Acc (%)**	**Time (sec)**	**Acc (%)**	**Time (sec)**
11	97.9	15.83	98	20.75	98.3	24.01	98.1	25.38
10	97.9	16.51	98	16.95	98.2	21.1	98	20.07
9	97.9	14.44	98	18.57	98.1	20.06	98	21.82
8	97.8	15.6	98	17.43	98.1	17.92	97.9	20.11
7	97.8	14.92	98	16.98	98	16.19	98	18.32
6	97.8	14.17	98	14.87	97.9	15.48	97.8	17.03
5	97.7	13.09	97.6	15.23	97.9	13.55	97.8	14.88
4	97.6	12.78	97.3	15.59	97.6	12.82	97.3	16.25
3	97.2	10.65	97.1	12.95	96.8	14.72	96.3	16.77
2	92.4	7.81	93.1	9.63	93	11.04	92.6	11.91
1	84.3	4.01	85.1	4.76	83.2	5.71	83.3	5.09

Table 9.6: Measurement of packet momentum performance for the number of packets vs the number of features.

9.5 Performance Under Different Classifiers

The C4.5 Decision Tree was the classifier employed in the aforementioned analyses in the previous sections using packet momentum. Table 9.7 shows the performance of packet momentum when using cross-validation for C4.5, Random Forests, Naive Bayes and Bayesian Network. The performance was measured for the 22 classes as in Section 9.3.2.

	C4.5	**Random Forest**	**Naive Bayes**	**Bayesian Network**
Accuracy (%)	97.92	98.25	39.2	90.58
Time (sec)	19.02	142.6	0.56	3.9

Table 9.7: Performance of Packet Momentum under different classifiers.

The performance of C4.5 (97.92 %) was increased to 98.25% when using Random Forests, while the time needed to build the model increased from 19.02 seconds to 142.6 seconds. Apparently, the increase in accuracy that Random Forests offers (0.33%) which is less than 1% does not substitute the increased time in building the model. Naive Bayes had the lowest time to build the model (0.56 seconds) but the performance suffered (39.2%). Bayesian Network has a balance performance between the time needed to build the model (3.9 seconds) and the accuracy (90.58%). Compared to Naive Bayes, Bayesian Network has much better performance but still, C4.5 is the best choice compared to time and accuracy.

A paired T-test [71] was run on the four classifiers in Table 9.7 to compare the performance (accuracy) of these classifiers when using the packet momentum for the 22 classes. C4.5 was used as a baseline scheme in a pair-wise comparison of the classifiers. 10-Fold Cross Validation was used for the four classifiers. Then, this procedure was repeated ten times which lead to the generation of 400 results. The test aims to provide evidence to reject the null hypothesis which means the accuracy of a classifier is statistically significantly better or worse than C4.5. As shown in Table 9.8 the confidence level in the test was 95% (0.05 significance level). The test shows that Random forest with a 98.2% accuracy was statistically significantly better than C4.5 with a 97.9% accuracy. Naive Bayes with a 39.2% accuracy was statistically significantly worse than C4.5. Bayesian Network with a 91% accuracy was statistically significantly worse than C4.5.

T-Test, Significance Level = 0.005, 10-Times 10-Fold Cross Validation			
Classifier	Random Forest	Naive Bayes	Bayesian Network
Statistically Significant	Better	worst	worst
Accuracy	98.2 %	39.2 %	91 %

Table 9.8: T-Test result for the accuracy of the C4.5 classifier compared to other classifiers.

9.6 Summary

This chapter proposed Packet Momentum: a set of features to identify multilayer-encryption anonymity networks. Packet Momentum aims to provide the suitable features that could provide sufficient information to identify multilayer-encryption anonymity networks efficiently. The proposed features in Packet Momentum are Maximum Packet Size, Frequency of Maximum Packet Size, Second Maximum Packet Size, Frequency of Second Maximum Packet Size, Packet Sequence, Sequence Speed and Packet Momentum. The Packet Momentum features were tested on identifying multilayer-encryption anonymity networks and showed high accuracy. Moreover, the results of using Packet Momentum on identifying applications running on the anonymity networks and on identifying obfuscated traffic used on anonymity networks showed that Packet Momentum is efficient for identifying such traffic as well.

Chapter 10

Conclusion

Multilayer-encryption anonymity networks provide privacy which has become a significant concern on today's Internet due to many attacks and privacy breaches. The anonymity and privacy these networks provide is a double-edged knife. Increasing attacks, threats and misuse of such valuable anonymity services trigger the need to identify such anonymity networks. Moreover, the implementation of the obfuscation techniques hardens the identification of such networks. Consequently, this research presents an analysis of multilayer-encryption anonymity networks and proposes a new approach for identifying them.

10.1 Dataset

Anon17 is an anonymity network data set which contains data from three anonymity networks: Tor, JonDonym and I2P. In addition to the traffic flows of these three anonymity networks, the data set includes applications traffic flows run on Tor and I2P. Furthermore, the data set contains traffic flows for the obfuscation techniques used on the Tor network: the pluggable transports. In addition, Anon17 includes data on the tunnels used on the I2P network. It would appear that Anon17 is the first publicly available anonymity network data set which covers three anonymity networks as well as obfuscation traffic.

10.2 Anonymity Measurement

Many systems provide anonymity for their users and most of these systems work on the separation between the users' identity and the final destination. The level of anonymity these services provide is affected by several factors, some of which are related to the design of the anonymity service itself. Others are related to how the system is used or the user's application/purpose in using the anonymity service. This research (i) proposes five factors which aim to measure anonymity level from the user's perspective; (ii) evaluates these factors for three anonymity services, namely Tor, JonDonym and I2P; and (iii) presents a mechanism to evaluate anonymity services based on the proposed factors and measure actual levels of anonymity. Understanding these factors and knowing how to address them is an important step in improving users' privacy. To this end, three popular anonymity systems, namely Tor, JonDonym and I2P, were used as case studies to analyze these factors. The analysis showed that even though these systems aim to provide anonymity to their users, user information is visible to the operators of the services. Furthermore, the infrastructure and the browser settings vary from one system to another. The setting is configured based on the developers'/administrators' evaluation of possible threats. The same threat might be considered high in one system but low in another. Based on the proposed factors, the anonymity of a given situation could be evaluated by a measurable mechanism. This evaluation could be used on any anonymity system using different scenarios.

10.3 Machine Learning Algorithms

Multilayer-encryption anonymity networks use multiple layers of encryption which makes inspecting the packets to identify such networks is not possible. In addition, some of these networks employ obfuscation techniques that make the identification task harder. One of the advantages of using Decision Tree is the interpretation of the result it gives through the visual representation of the Decision Tree. This interpretation of the solution is not possible when using many of the other machine learning algorithms. This supports the suitability of the Decision Tree for many encrypted traffic classification problems.

The number of used features is important to define the computational cost and time required in a classification task. The higher the number of features the higher the time to build the model. Decision Tree has the ability to select - based on the information gain - the most important features. Moreover, the lower the number of features features the less time to build the Decision Tree model. Reducing the number of features is helpful in a way that makes building the model and obtaining the result fast enough to make a decision while the connection is active. On the other hand, describing the data with a hight number of features could result in losing the generalization of the solution to the data at hand. While using a low number of features enables the learned solution to be more robust to the unseen/new data.

10.4 Traffic Flow Analysis of Anonymity Networks

Traffic flow analysis has demonstrated high accuracy in identifying multilayer-encryption anonymity networks. Experiments exercised on this research have shown the flow behaviour of applications when involving anonymity networks. Based on the design of the anonymity network, collecting the data and extracting the flows requires having some knowledge about the way they work. For example, the Tor network packs the traffic in cells and circuits while the I2P network uses tunnels. These differences in the design change the level of behaviour profiling that the traffic flow analysis can retrieve from the analysis.

Furthermore, traffic flow analysis has shown high accuracy in identifying obfuscated traffic. The Tor network has offered pluggable transports as the obfuscation technique. JonDonym has two forwarder techniques to relay connection to the JonDonym network. In Tor the pluggable transports with their different forms provide evasions or resistance to censorship. Obfsproxy is the framework used by these pluggable transports to obfuscate the user connection to the Tor network. This obfuscation concentrates mainly on hiding the contents that make the connections to the Tor network recognizable. Consequently, using deep packet inspection cannot detect them as with Tor. Pluggable transports obfuscated the Tor traffic successfully to look like random or different forms of traffic. At the same time this success at hiding the content is not optimum. The obfuscation in the pluggable transports changes the content shape distinctly from Tor which creates a fingerprint for the obfuscated pluggable transports. The results of this research show that pluggable transport flows have their own unique fingerprints which make them recognizable.

10.5 Efficiency and Accuracy Using Packet Momentum

Packet Momentum could use as few as three packets to identify multilayer-encryption anonymity networks with a high accuracy. The number of features in the Packet Moment approach is eleven. These features are fewer than the number of features employed previously in experiments on multilayer-encryption anonymity networks. The features have been inspired by the analysis and experiments of this research. For example, the traffic flow analysis of the Tor network highlights the fact that Tor uses fixed-size cells to communicate and carry data. This behaviour has its reflection in the packet sizes that Tor shows in the traffic analysis. Consequently, maximum packet size and its frequency are included in the Packet Momentum features pool. Furthermore, the usual way for the anonymity user to include obfuscated traffic in the connection to the anonymity network starts with the user sending a connection request to the obfuscation server and waiting for a response. By contrast, Flashproxy starts the connections to the user, not the other way around. This example of communication behaviour inspired the use of sequence and sequence speed. The features on Packet Momentum demonstrated a high accuracy on different data sets.

10.6 Future Work

For future work Anon17 will be expanded to include additional applications run on anonymity networks. Future research on anonymity measurement will continue to analyze other anonymity systems based on the proposed five factors and will evaluate them using the expanded quantification approach under adversarial conditions. In addition Packet Momentum will be analyzed on a larger scale and investigated for the implementation of this approach in a network tool used to identify anonymity networks. The Packet Momentum tool will be studied as well for identifying applications in conjunction with the anonymity networks.

Bibliography

[1] Dhiah el Diehn Abou-Tair, Lexi Pimenidis, Jens Schomburg, and Benedikt Westermann. Usability inspection of anonymity networks. In *Privacy, Security, Trust and the Management of e-Business, 2009. CONGRESS'09. World Congress on*, pages 100–109. IEEE, 2009.

[2] Alexa. http://www.alexa.com/topsites.

[3] Mashael AlSabah, Kevin Bauer, and Ian Goldberg. Enhancing Tor's performance using real-time traffic classification. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 73–84, New York, NY, USA, 2012. ACM.

[4] Hyun-Min An, Myung-Sup Kim, and Jae-Hyun Ham. Application traffic classification using statistic signature. In *Network Operations and Management Symposium (APNOMS), 2013 15th Asia-Pacific*, pages 1–6. IEEE, 2013.

[5] Anon17: Anonymity networks dataset. https://web.cs.dal.ca/~shahbar/data.html.

[6] John Barker, Peter Hannay, and Patryk Szewczyk. Using traffic analysis to identify the second generation onion router. In *Embedded and Ubiquitous Computing (EUC), 2011 IFIP 9th International Conference on*, pages 72–78. IEEE, 2011.

[7] Kevin S Bauer, Micah Sherr, and Dirk Grunwald. ExperimenTor: A testbed for safe and realistic Tor experimentation. In *CSET*, 2011.

[8] Laurent Bernaille and Renata Teixeira. Early recognition of encrypted applications. *Passive and Active Network Measurement*, pages 165–175, 2007.

[9] Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The disadvantages of free MIX routes and how to overcome them. In *Designing Privacy Enhancing Technologies*, pages 30–45. Springer, 2001.

[10] Alex Biryukov, Ivan Pustogarov, and Ralf-Philipp Weinmann. Trawling for Tor hidden services: Detection, measurement, deanonymization. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 80–94. IEEE, 2013.

[11] Christopher M. Bishop. *Pattern recognition and machine learning.* Springer, 2006.

[12] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[13] Dennis Brown. Resilient botnet command and control with Tor. https://www.defcon.org/images/defcon-18/dc-18-presentations/D.Brown/DEFCON-18-Brown-TorCnC.pdf, 2010.

[14] Matteo Casenove and Armando Miraglia. Botnet over Tor: The illusion of hiding. In *2014 6th International Conference On Cyber Conflict (CyCon 2014)*, pages 273–282, June 2014.

[15] Netflix hammers cross-border watchers and there may be no way out . http://www.cbc.ca/news/business/netflix-border-hopping-television-1.3805525.

[16] Abdelberi Chaabane, Pere Manils, and Mohamed Ali Kaafar. Digging into anonymous traffic: A deep analysis of the Tor anonymizing network. In *Network and System Security (NSS), 2010 4th International Conference on*, pages 167–174. IEEE, 2010.

[17] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[18] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology*, 1(1):65–75, 1988.

[19] Nicolas Christin. Traveling the Silk Road: A measurement analysis of a large anonymous online marketplace. In *Proceedings of the 22nd international conference on World Wide Web*, pages 213–224. ACM, 2013.

[20] Cisco Netflow. http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html.

[21] Jeremy Clark, Paul C Van Oorschot, and Carlisle Adams. Usability of anonymous web browsing: an examination of Tor interfaces and deployability. In *Proceedings of the 3rd symposium on Usable privacy and security*, pages 41–51. ACM, 2007.

[22] Deluge. http://deluge-torrent.org/.

[23] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In *International Workshop on Privacy Enhancing Technologies*, pages 54–68. Springer, 2002.

[24] Roger Dingledine and Nick Mathewson. Tor path specification. https://gitweb.torproject.org/torspec.git/tree/path-spec.txt.

[25] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[26] Diana S Dolliver. Evaluating drug trafficking on the Tor network: Silk Road 2, the sequel. *International Journal of Drug Policy*, 26(11):1113–1123, 2015.

[27] Diana S. Dolliver and Jennifer L. Kenney. Characteristics of drug vendors on the Tor network: A cryptomarket comparison. *Victims & Offenders*, 11(4):600–620, 2016.

[28] Kevin P Dyer, Scott E Coull, Thomas Ristenpart, and Thomas Shrimpton. Protocol misidentification made easy with format-transforming encryption. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 61–72. ACM, 2013.

[29] Peter Eckersley. How unique is your web browser? In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 1–18. Springer, 2010.

[30] I2P: Frequently asked questions. https://geti2p.net/en/faq#eepsite.

[31] Christoph Egger, Johannes Schlumberger, Christopher Kruegel, and Giovanni Vigna. Practical attacks against the I2P network. In *Proceedings of the 16th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2013)*, October 2013.

[32] Juan A Elices, Fernando Perez-Gonzalez, and Carmela Troncoso. Fingerprinting Tor's hidden service log files using a timing channel. In *Information Forensics and Security (WIFS), 2011 IEEE International Workshop on*, pages 1–6. IEEE, 2011.

[33] Nicholas P. Fandos. Harvard sophomore charged in bomb threat. http://www.thecrimson.com/article/2013/12/17/student-charged-bomb-threat/, December 2013.

[34] David Fifield, Nate Hardison, Jonathan Ellithorpe, Emily Stark, Dan Boneh, Roger Dingledine, and Phil Porras. Evading censorship with browser-based proxies. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 239–258. Springer, 2012.

[35] Peter Flach. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press, 2012.

[36] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2-3):131–163, 1997.

[37] Garlic routing. https://geti2p.net/en/docs/how/garlic-routing.

[38] Rentao Gu, Hongxiang Wang, and Yuefeng Ji. Early traffic identification using Bayesian networks. In *Network Infrastructure and Digital Content, 2010 2nd IEEE International Conference on*, pages 564–568. IEEE, 2010.

[39] Fariba Haddadi and A Nur Zincir-Heywood. Benchmarking the effect of flow exporters and protocol filters on botnet traffic classification. *IEEE Systems journal*, 10(4):1390–1401, 2016.

[40] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[41] I2Ps threat model: Harvesting attacks. https://geti2p.net/en/docs/how/threat-model#harvesting.

[42] Michael Herrmann and Christian Grothoff. Privacy implications of performance-based peer selection by onion routers: A real-world case study using I2P. In *Proceedings of the 11th Privacy Enhancing Technologies Symposium (PETS 2011)*, July 2011.

[43] Badr Hssina, Abdelkarim Merbouha, Hanane Ezzikouri, and Mohammed Erritali. A comparative study of decision tree ID3 and C4.5. *International Journal of Advanced Computer Science and Applications*, 4(2):13–19, 2014.

[44] HTTPS-everywhere extension. https://www.eff.org/https-everywhere.

[45] Chang Huijun, Shan Hong, and Zhu Hong. Early recognition of internet service flow. In *Wireless and Optical Communication Conference (WOCC), 2013 22nd*, pages 464–468. IEEE, 2013.

[46] I2P: Peer profiling and selection. https://geti2p.net/en/docs/how/peer-selection.

[47] I2P: Plugins. https://geti2p.net/en/docs/plugins.

[48] I2P: The network database. https://geti2p.net/en/docs/how/network-database.

[49] I2PSnark. https://geti2p.net/en/docs/how/tech-intro#app.i2psnark.

[50] iMacros. http://imacros.net/overview.

[51] InMon sFlow. http://www.inmon.com/technology/index.php.

[52] JAP: Data collection techniques. http://anon.inf.tu-dresden.de/help/jap_help/en/help/wwwprivacy_technik.html.

[53] jIRCii: The ultimate IRC client. http://www.oldschoolirc.com/.

[54] Anonymous surfing with JonDoFox. https://anonymous-proxy-servers.net/en/jondofox.html.

[55] Jondonym InfoService. https://anonymous-proxy-servers.net/en/help/infoservice.html.

[56] Juniper J-Flow. https://www.juniper.net/documentation/en_US/junose10.3/information-products/topic-collections/swconfig-ip-services/id-37225a.html.

[57] Teuvo Kohonen. *Self-organizing maps.* Springer Berlin Heidelberg, Berlin, Germany, 2001.

[58] Chenglong Li, Yibo Xue, Yingfei Dong, and Dongsheng Wang. Super nodes in Tor: Existence and security implication. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 217–226. ACM, 2011.

[59] Zhen Ling, Junzhou Luo, Kui Wu, and Xinwen Fu. Protocol-level hidden server discovery. In *INFOCOM, 2013 Proceedings IEEE*, pages 1043–1051. IEEE, 2013.

[60] Zhen Ling, Junzhou Luo, Wei Yu, and Xinwen Fu. Equal-sized cells mean equal-sized packets in Tor? In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–6. IEEE, 2011.

[61] Zhen Ling, Junzhou Luo, Wei Yu, Ming Yang, and Xinwen Fu. Extensive analysis and large-scale empirical evaluation of Tor bridge discovery. In *INFOCOM, 2012 Proceedings IEEE*, pages 2381–2389. IEEE, 2012.

[62] Peipeng Liu, Lihong Wang, Qingfeng Tan, Quangang Li, Xuebin Wang, and Jinqiao Shi. Empirical measurement and analysis of I2P routers. *Journal of Networks*, 9:2269–2278, September 2014.

[63] Maji. https://research.wand.net.nz/software/maji.php.

[64] Matlab. https://www.mathworks.com/products/matlab/.

[65] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer, 2002.

[66] Damon McCoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Shining light in dark places: Understanding the Tor network. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 63–76. Springer, 2008.

[67] Tom M Mitchell. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45(37):870–877, 1997.

[68] Keaton Mowery, Dillon Bogenreif, Scott Yilek, and Hovav Shacham. Fingerprinting information in JavaScript implementations. *Proceedings of W2SP*, 2:180–193, 2011.

[69] Keaton Mowery and Hovav Shacham. Pixel perfect: Fingerprinting canvas in HTML5. *Proceedings of W2SP*, pages 1–12, 2012.

[70] Steven J Murdoch. Quantifying and measuring anonymity. In *Data Privacy Management and Autonomous Spontaneous Security*, pages 3–13. Springer, 2014.

[71] Claude Nadeau and Yoshua Bengio. Inference for the generalization error. *Machine Learning*, 2001.

[72] Mary Natrella. *NIST/SEMATECH e-Handbook of Statistical Methods.* NIST/SEMATECH, July 2010.

[73] Evolving Proxy Detection as a Global Service. https://media.netflix.com/en/company-blog/evolving-proxy-detection-as-a-global-service.

[74] NIMS: Network information management and security group. https://projects.cs.dal.ca/projectx/.

[75] NoScript Firefox extension. https://noscript.net/.

[76] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*, pages 103–114. ACM, 2011.

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

[78] Project: AN.ON anonymity. http://anon.inf.tu-dresden.de/index.

[79] J Ross Quinlan. Discovering rules by induction from large collections of example. *Expert Systems in the Micro Electronics Age*, 1979.

[80] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[81] Thorsten Ries, Andriy Panchenko, and Thomas Engel. Comparison of low-latency anonymous communication systems: practical usage and performance. In *Proceedings of the Ninth Australasian Information Security Conference-Volume 116*, pages 77–86. Australian Computer Society, Inc., 2011.

[82] Karina Rigby. Anonymity on the internet must be protected. http://groups.csail.mit.edu/mac/classes/6.805/student-papers/fall95-papers/rigby-anonymity.html, 1995.

[83] Amirali Sanatinia and Guevara Noubir. OnionBots: Subverting privacy infrastructure for cyber attacks. In *Proceedings of the 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, DSN '15, pages 69–80, Washington, DC, USA, 2015. IEEE Computer Society.

[84] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In *International Workshop on Privacy Enhancing Technologies*, pages 41–53. Springer, 2002.

[85] Khalid Shahbar and A Nur Zincir-Heywood. Benchmarking two techniques for Tor classification: Flow level and circuit level classification. In *2014 IEEE Symposium on Computational Intelligence in Cyber Security (CICS)*, pages 1–8, Dec 2014.

[86] Khalid Shahbar and A Nur Zincir-Heywood. Traffic flow analysis of Tor pluggable transports. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 178–181, Nov 2015.

[87] Khalid Shahbar and A Nur Zincir-Heywood. Anon17: Network traffic dataset of anonymity services. Technical Report CS-2017-03, Dalhousie University of Halifax, February 2017.

[88] Khalid Shahbar and A. Nur Zincir-Heywood. Effects of shared bandwidth on anonymity of the I2P network users. In *38th IEEE Symposium on Security and Privacy Workshops, 2nd International Workshop on Traffic Measurements for Cybersecurity (WTMC 2017)*, May 2017.

[89] Softflowd. http://www.mindrot.org/projects/softflowd/.

[90] SOM toolbox. http://www.cis.hut.fi/somtoolbox/.

[91] Kyle Soska and Nicolas Christin. Measuring the longitudinal evolution of the online anonymous marketplace ecosystem. In *USENIX Security*, volume 15, 2015.

[92] STEM. https://stem.torproject.org/.

[93] Talieh Seyed Tabatabaei, Fakhri Karray, and Mohamed Kamel. Early internet traffic recognition based on machine learning methods. In *Electrical & Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on*, pages 1–5. IEEE, 2012.

[94] Dmitry Tarakanov. The inevitable move 64-bit ZeuS enhanced with Tor. https://securelist.com/blog/events/58184/the-inevitable-move-64-bit-zeus-enhanced-with-tor/, December 2013.

[95] Tcptrace. http://www.tcptrace.org/.

[96] Juan Pablo Timpanaro, Isabelle Chrisment, and Olivier Festor. Monitoring the I2P network, October 2011.

[97] Tor bridges. https://www.torproject.org/docs/bridges.html.en.

[98] The design and implementation of the Tor browser. https://www.torproject.org/projects/torbrowser/design/#idm29.

[99] Tor: Meek. https://trac.torproject.org/projects/tor/wiki/doc/meek.

[100] Tor Obfs3. https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/tree/doc/obfs3/obfs3-protocol-spec.txt.

[101] Tor partially blocked in China. https://blog.torproject.org/blog/tor-partially-blocked-china, October 2009.

[102] Tor pluggable transports. https://www.torproject.org/docs/pluggable-transports.html.en.

[103] Tor: How to Torify. https://trac.torproject.org/projects/tor/wiki/doc/TorifyHOWTO.

[104] Tor's protocol specifications. https://gitweb.torproject.org/torspec.git/tree/control-spec.txt.

[105] Gergely Tóth, Zoltán Hornák, and Ferenc Vajda. Measuring anonymity revisited. In *Proceedings of the Ninth Nordic Workshop on Secure IT Systems*, pages 85–90, 2004.

[106] Tranalyzer2. http://tranalyzer.com/.

[107] Tunnel implementation. https://geti2p.net/en/docs/naming.

[108] Unidirectional tunnels. https://geti2p.net/en/docs/tunnels/unidirectional.

[109] Cynthia Wagner, Gerard Wagener, Radu State, Alexandre Dulaunoy, and Thomas Engel. Breaking Tor anonymity with game theory and data mining. *Concurrency and Computation: Practice and Experience*, 24(10):1052–1065, 2012.

[110] Rolf Wendolsky, Dominik Herrmann, and Hannes Federrath. Performance comparison of low-latency anonymisation services from a user perspective. In *International Workshop on Privacy Enhancing Technologies*, pages 233–253. Springer, 2007.

[111] Benedikt Westermann and Dogan Kesdogan. Malice versus AN.ON: Possible risks of missing replay and integrity protection. In *International Conference on Financial Cryptography and Data Security*, pages 62–76. Springer, 2011.

[112] Tim Wilde. Great firewall Tor probing. https://gist.github.com/twilde/da3c7a9af01d74cd7de7, Jan 2012.

[113] Philipp Winter and Stefan Lindskog. How the great firewall of China is blocking Tor. In *FOCI*, 2012.

[114] Philipp Winter, Tobias Pulls, and Juergen Fuss. ScrambleSuit: A polymorphic network protocol to circumvent censorship. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, pages 213–224. ACM, 2013.

[115] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, and S Yu Philip. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008.

[116] YAF. http://tools.netsa.cert.org/yaf/index.html.

Appendices

Appendix A

Calculation of the Features on Packet Momentum

A.1 Calculation of Packet Sequence

As shown in Section 9.2.5, the packet sequence feature calculates how strong is the change in the packet direction between two communication parties, A and B. Packet Sequence takes into consideration the number of times the packets keep going in one direction. To explain how to calculate the Packet Sequence for Case 1 in Section 9.2, the packets can be arranged according to the direction, as shown in Table A.1. The time is not taken into consideration when calculating how long the packets stay in one direction. The time is used here only to find the change in the direction of the packets. The (+) sign means that the packet direction is from A to B. The opposite is true for the (-) sign. No packets arrived at 3, 6, 7, 9 or 12 seconds, so the table's cells are highlighted in gray to show that. Based on the sign which indicates the direction, one is added whenever there is a (+) sign and one is subtracted whenever there is a (-) sign. At the end all the values of the direction change are added together to find the packet sequence. Table A.2 shows the value of the direction each time a change in the direction is happening. Instead of using the time from 0 to 14 seconds, which is the time that the packet originally arrived at A or B, the change in direction will be used. At the first packet the sequence value in Table A.2 is set according to the direction of the packet (A to B). For the next packets, if the direction does not change, the sequence value is increased by one. When the direction changes (B to A), the sequence value is reduced by one, and so on. The packet sequence is the summation of all the sequence values. In this case, the packet sequence is 5.

Time	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Direction	+	-	■	+	-	■	■	-	■	+	+	■	+	-
Size	1	1	■	2	2	■	■	1	■	2	2	■	3	2

Table A.1: Direction of packets for Case 1.

Packet	**Sequence Value**	**Direction**	**Packet Sequence**
1	1	A to B	1
2	0	B to A	1 + 1 = 1
3	1	A to B	1 + 1 = 2
4	0	B to A	2 + 0 = 2
5	-1	B to A	2 + (-1) = 1
6	0	A to B	1 + 0 = 1
7	1	A to B	1 + 1 = 2
8	2	A to B	2 + 2 = 4
9	1	B to A	4 + 1 = **5**

Table A.2: Final calculations of packet sequence.

A.2 Calculation of Sequence Speed

To calculate the Sequence Speed for Case 1, it is necessary to add the inter arrival time from Table A.1 as shown in Table A.3:

Time	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Direction	+	-	■	+	-	■	■	-	■	+	+	■	+	-
IAT	1	1	■	2	1	■	■	3	■	2	1	■	2	1

Table A.3: Direction and Inter Arrival time for Case 1.

The direction in Table A.3 is the same as in Table A.1. The inter arrival time is the difference between the time of two consequent packets and it is independent of the direction. In the calculation of the sequence speed the change of direction will be measured with the inter arrival time. Then the summary of all these values will be added to find the sequence speed. Table A.4 shows the calculation of the sequence speed for Case 1. The column "Sequence Value" in the table represents the cumulative direction change for each instance of a packet arriving at that time. The column "Sequence Speed Value" is the multiplication of the IAT with the "Sequence Value". Finally the Sequence Speed is calculated by adding all the results in the "Sequence Speed Value" column. The sequence speed in Case 1 is 6.

A.3 Calculation of Packet Momentum

The packet momentum includes the size of the packets when evaluating the direction and time of the packets. The size is used to scale the amount of change up or down (in

Time	**Sequence Value**	**IAT**	**Sequence Speed Value**	**Sequence Speed**
1	1	1	1	1
2	0	1	0	1 + 0 = 1
3	▨	▨	▨	▨
4	1	2	2	1 + 2 = 3
5	0	1	0	3 + 0 = 3
6	▨	▨	▨	▨
7	▨	▨	▨	▨
8	-1	3	-3	3 - 3 = 0
9	▨	▨	▨	▨
10	0	2	0	0 + 0 = 0
11	1	1	1	0 + 1 = 1
12	▨	▨	▨	▨
13	2	2	4	1 + 4 = 5
14	1	1	1	5 + 1 = **6**

Table A.4: Calculation of the Sequence Speed for Case 1.

the direction of the communication between A and B, in this case). Table A.5 shows the size of the packets in Case 1 to be used in the packet momentum calculation. The table includes as well the Inter Arrival time and the direction of the packets taken from Table A.3.

Time	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Direction	+	-	▨	+	-	▨	▨	-	▨	+	+	▨	+	-
IAT	1	1	▨	2	1	▨	▨	3	▨	2	1	▨	2	1
Size	1	1	▨	2	2	▨	▨	1	▨	2	2	▨	3	2

Table A.5: Size, Time, Direction and Inter Arrival time for Case 1.

The calculation of the packet momentum requires knowing the size of the packets, the inter arrival time and the packet sequence. The latter has been calculated earlier and could be used to calculate the packet monument. Table A.6 shows how to calculate the packet momentum from the required values. The first two columns in the table are the same from the previous calculation of the sequence speed. The third column is the size of the packets at the time the packets arrive. The packet momentum value is the multiplication of the size, the inter arrival time and the sequence value. Finally, the packet momentum is calculated by the summation of all the column “packet momentum value”.

Time	**Sequence Value**	**IAT**	**Size**	**Packet Momentum Value**	**Packet Momentum**
1	1	1	1	1	1
2	0	1	1	0	1 + 0 = 1
3	▨	▨	▨	▨	▨
4	1	2	2	4	1 + 4 = 5
5	0	1	2	0	5 + 0 = 5
6	▨	▨	▨	▨	▨
7	▨	▨	▨	▨	▨
8	-1	3	1	-3	5 - 3 = 2
9	▨	▨	▨	▨	▨
10	0	2	2	0	2 + 0 = 2
11	1	1	2	2	2 + 2 = 4
12	▨	▨	▨	▨	▨
13	2	2	3	12	4 + 12 = 16
14	1	1	2	2	16 + 2 = **18**

Table A.6: Calculation of Packet Momentum for Case 1.

Appendix B

Packet Momentum Pseudo code

Algorithm 6 shows the Pseudo code of all the features of Packet Momentum.

Algorithm 6 : Pseudo Code of all Packet Momentum Features.

```
inputs n: total number of packets in the direction of A to B and packets in the direction of B to A.
$IAT_n$: inter arrival time of packet n.
$T_n$: Time of packet n.
$S_n$: Size of packet n.
$PSize_n$: Array of packet size for A and B.

for i = 1 to n do
  Find the maximum packet size in $PSize_n$
  Find the frequency of maximum packet size in $PSize_n$
  Find the second maximum packet size in $PSize_n$
  Find the frequency of second maximum packet size in $PSize_n$
end for

$maxPS_1 = the\, maximum\, packet\, size$
$FPS_1 = the\, frequency\, of\, maximum\, packet\, size$
$maxPS_2 = the\, second\, maximum\, packet\, size$
$FPS_2 = the\, frequency\, of the\, second\, maximum\, packet\, size$
```

```
if the first packet is from A to B then
    $SequenceValue_1 = 1$
else
    $SequenceValue_1 = -1$
end if

$PacketSequence = SequenceValue_1$
$IAT_1 = T_1$
$SequenceSpeedValue_1 = SequenceValue_1 * IAT_1$
$SequenceSpeed = SequenceSpeedValue_1$
$PacketMomentumValue_1 = SequenceSpeedValue_1 * S_1$
$PacketMomentum = PacketMomentumValue_1$

for i = 2 to n do

    if the current packet is from B to A then
        $SequenceValue_n = SequenceValue_{n-1} - 1$
    else
        $SequenceValue_n = SequenceValue_{n-1} + 1$
    end if

    $IAT_n = T_n - T_{n-1}$
    $SequenceSpeedValue_n = SequenceValue_n * IAT_n$
    $SequenceSpeed = SequenceSpeed + SequenceSpeedValue_n$
    $PacketSequence = PacketSequence + SequenceValue_n$
    $PacketMomentumValue_n = SequenceSpeedValue_n * S_n$
    $PacketMomentum = PacketMomentum + PacketMomentumValue_n$
end for
```

```
Return $maxPS_1$
Return $FPS_1$
Return $maxPS_2$
Return $FPS_2$
Return $PacketSequence$
Return $SequenceSpeed$
Return $PacketMomentum$
```