Classifying Tor Traffic Using Character Analysis

Pitpimon Choorod

Department of Computer and Information Sciences

University of Strathclyde, Glasgow

A Thesis Submitted in Fulfilment of the Requirements for the Degree of Doctor of Philosophy

January 31, 2024

Declaration

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Acknowledgements

I would like to express my heartfelt gratitude and profound appreciation to my former supervisor, *Dr. George Weir* for his excellently supportive supervision, constructive criticism, valuable suggestions, and endless patience. Without his energising enthusiasm and encouragement, this thesis would not have been completed. He was always generous with his time, effort, and insightful advice, and he helped me develop into a better critical-thinking researcher. I would also like to thank my current supervisor, *Prof. Anil Fernando*, for his invaluable suggestions and feedback.

I extend my heartfelt gratitude to the Viva committee, especially to *Joseph El Gemayel*, for his excellent coordination which ensured a seamless examination process. My sincere thanks go to the internal examiner, *Yashar Moshfeghi*, for his insightful feedback which significantly enhanced the comprehensiveness of my thesis. I am also deeply grateful to the external examiner, *Prof. Andreas Aßmuth*, for his invaluable guidance in meticulously directing my research, and his endless and tireless support in refining and correcting my errors. His mentorship has been instrumental in my moral and academic development. He is a great role model for me, especially in terms of work discipline and the ways of caring for students, which inspires me to follow his example in my upcoming academic career.

I would like to take this opportunity to thank my country and the Thai government, for providing me with sponsorship and the opportunity to pursue a doctorate degree.

I would like to thank the administrative and technical staff members of the Computer and Information Sciences (CIS) who have been so gracious in providing advice and assistance in their respective roles.

I would like to deeply thank my father, mother, and brother for their endless sacrifices and unwavering support, without them I would not have been able to pursue and achieve my dreams and complete my PhD degree. I'd like to dedicate this thesis to my cherished family. Since the COVID-19 pandemic, they have become even more supportive and motivating to me, allowing me to continue on this difficult path without giving up.

I also express my deep gratitude to several individuals who have been pivotal in my PhD journey and never left me behind. Firstly, a big thank you to *Dr. Sasin Janpuangtong* for your helpful advice. My thanks also go to *Nawaluck Sangsanit* for being a sympathetic listener and mental supporter during my toughest moments. I am grateful to *Marvin Wright* for being a reliable friend, always there when I needed him. I appreciate *Dr. Srisuda Chaikitkaew* for her kind wishes and genuine empathy during my difficult times. *Dr. Prapatsorn Wejprasit* deserves special mention for her invaluable and consistent support throughout this journey. Her sincere care and treatment will be forever in my deepest mind.

Importantly, special thanks are due to *Asst. Prof. Dr. Niramol Jantarachart*, who not only provided invaluable mental support with her comforting embraces, crucial for helping me stay relaxed and fearless, thereby maintaining a positive outlook and happiness throughout my journey, but also offered insightful guidance and feedback that were key in helping me cross the finish line. Additionally, I extend my thanks to everyone else who was behind the scenes, assisting with various study-related tasks. Finally, I must mention my little cat, *Ngaw-kung*, whose timely presence greatly helped to alleviate my distress every day. Having you in my life is truly a blessing.

Abstract

Tor is a privacy-preserving network that enables users to browse the Internet anonymously. Although the prospect of such anonymity is welcomed in many quarters, Tor can also be used for malicious purposes, prompting the need to monitor Tor network connections. Most traffic classification methods depend on flow-based features, due to traffic encryption. However, these features can be less reliable due to issues like asymmetric routing, and processing multiple packets can be time-intensive. In light of Tor's sophisticated multi-layered payload encryption compared with nonTor encryption, our research explored patterns in the encrypted data of both networks, challenging conventional encryption theory which assumes that ciphertexts should not be distinguishable from random strings of equal length.

Our novel approach leverages machine learning to differentiate Tor from nonTor traffic using only the encrypted payload. We focused on extracting statistical hex character-based features from their encrypted data. For consistent findings, we drew from two datasets: a public one, which was divided into eight application types for more granular insight and a private one. Both datasets covered Tor and nonTor traffic. We developed a custom Python script called Charcount to extract relevant data and features accurately. To verify our results' robustness, we utilized both Weka and scikit-learn for classification.

In our first line of research, we conducted hex character analysis on the encrypted payloads of both Tor and nonTor traffic using statistical testing. Our investigation revealed a significant differentiation rate between Tor and nonTor traffic of 95.42% for the public dataset and 100% for the private dataset.

The second phase of our study aimed to distinguish between Tor and nonTor traffic using machine learning, focusing on encrypted payload features that are independent of length. In our evaluations, the public dataset yielded an average accuracy of 93.56% when classified with the Decision Tree (DT) algorithm in scikit-learn, and 95.65% with the j48 algorithm in Weka. For the private dataset, the accuracies were 95.23% and 97.12%, respectively. Additionally, we found that the combination of WrapperSubsetEval+BestFirst with the J48 classifier both enhanced accuracy and optimized processing efficiency.

In conclusion, our study contributes to both demonstrating the distinction between Tor and nonTor traffic and achieving efficient classification of both types of traffic using features derived exclusively from a single encrypted payload packet. This work holds significant implications for cybersecurity and points towards further advancements in the field.

Contents

List of Figures

List of Tables

Acronyms

3DES or TDES Triple-DES

AES Advanced Encryption Standard

AI Artificial Intelligence

ANNs Artificial Neural Networks

AUC Area Under The ROC Curve

BoW Bag-of-Words

CART Classification and Regression Trees

CHARSTAT Character Statistics Analysis using Python

CNNs Convolutional Neural Networks

DES Data Encryption Standard

DL Deep Learning

DNNs Deep Neural Networks

DPI Deep Packet Inspection

DTs Decision Tree

EDH Ephemeral Diffie-Hellman

FP False Negative

FP False Positive

FPR False Positive Rate

FTP File Transfer Protocol

GUI Graphical User Interface

HMM Hidden Markov Model

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

ID3 Iterative Dichotomiser 3

IETF Internet Engineering Task Force

IP Internet Protocol

ISP Internet Service Provider

kNN k-Nearest-Neighbours

LSTM-RNNs Long Short-Term Memory Recurrent Neural Networks

ML Machine Learning

MLPs Multilayer perceptrons

MSS Maximum Segment Size

MTU Maximum Transmission Unit

NGOs Non-governmental organizations

NIST The National Institute for Standards and Technology

NLP Natural Language Processing

NLR The United States Naval Research Laboratory

NNs Neural Networks

NTC Network Traffic Classification

OS Operating System

P2P Peer-to-Peer

PCA Principal Component Analysis

QoS Quality of Service

RFs Random Forests

ROC Receiver Operating Characteristic

RSA Rivest-Shamir-Adleman

SA Sentiment Analysis

SFTP SSH File Transfer Protocol

SMTP Simple Mail Transfer Protocol

SPSS Statistical Package for the Social Sciences

SSH Secure Shell

SSL Secure Socket Layer

SVM Support Vector Machine

TCP Transmission Control Protocol

TCP/IP Transmission Control Protocol/Internet Protocol

TF-IDF Term Frequency-Inverse Document Frequency

TLS Transport Layer Security

TN True Negative

TP True Positive

TPR True Positive Rate

UDP User Datagram Protocol

VPN Virtual Private Network

WebRTC Web Real-Time Communication

WEKA Waikato Environment for Knowledge Analysis

"Make everything as simple as possible, but not simpler"

Albert Einstein (1879-1955)

Chapter 1

Introduction

This introductory chapter serves as an overview of our research, providing all the necessary information to help the reader comprehend the main aspects of this study. We begin by highlighting the importance of this research, followed by specifying the problem statement, which includes a discussion of potential challenges. Our study employs a systematic approach that involves formulating and testing hypotheses. Therefore, the research questions that guide this study are integral to formulating these hypotheses. To provide a comprehensive understanding of our research, we define the research objectives, which help to establish the study's scope and conceptual framework. At the end of the present chapter, a thesis outline gives a brief synopsis of each chapter, providing the reader with a clear understanding of the topics that will be addressed throughout the remainder of the thesis.

1.1 Background and Motivation

The World Wide Web has become an indispensable component of our daily lives, eliminating communication boundaries for people all over the world. The number of people accessing the web via various kinds of electronic devices is growing day by day. The web might be a perfect tool if users comprehend the potential dangers that can arise when browsing and use it prudently and smartly. When accessing the web through standard browsers such as Google Chrome and Internet Explorer, these tools

do not fully protect users' privacy. In the standard web browsing scenario, which can be termed as nonTor traffic, data transmission is more transparent. When employing security measures, nonTor traffic typically employs Transport Layer Security (TLS) for encrypting data packets during its transfer. The essential purpose of this encryption is to provide security and privacy for web-based transactions. Typically, when we browse the web, any authorities, such as the government, spies or our Internet Service Provider (ISP) can obtain information on our browsing habits. These entities may be motivated to profit from targeted adverts as well as online tracking, leading to potential cybersecurity threats. There are several techniques to protect user privacy while online, including connecting to a Virtual Private Network (VPN), using browsers in anonymous surfing modes (e.g., Incognito, in Chrome), using privacy-preserving search engines (e.g., DuckDuckGo or Swisscows), and utilising the well-known anonymity-focused Tor browser. This study focuses on characteristics of the Tor network that underpins the operation of Tor browsing.

Tor, originally developed by the United States Naval Research Lab, is a popular tool that provides privacy and anonymity for network users. Its primary purpose is to shield user identities from traffic monitoring and surveillance, making it a valuable resource for those seeking freedom in their online activities. In contrast to the single-layer TLS encryption typical of nonTor traffic, Tor makes use of multiple layers of encryption. This multi-layer approach, often termed 'onion routing', is designed to wrap data in several encryption layers corresponding to the randomly chosen relay nodes it passes through. While TLS in nonTor ensures data confidentiality for a direct communication path, Tor's use of TLS obscures both the data and the routing information, making it more complex and secure. However, the benefits of Tor also give rise to potential misuse by malicious actors. For those managing the technology, it becomes crucial to find a middle ground between allowing legitimate users to benefit from Tor's privacy features and preventing its exploitation for nefarious activities.

Numerous studies (Faizan & Khan, 2019; Liggett et al., 2020; Monk et al., 2018; Owen & Savage, 2015) have documented the wide range of illegal services facilitated by the Tor network, including drug trafficking, fraud, counterfeiting, weapons sales,

terrorism, child abuse, and sale of illegally obtained private and sensitive information. High-profile cybercriminal activities using the Tor network include the Silk Road darknet marketplace (Liggett et al., 2020) and Command and Control (C&C) communications for network exploit like Mevade Botnet (Hopper, 2014), ChewBacca malware (Mirea et al., 2018), and CryptoWall 2.0 ransomware (Yetter, 2015). These examples illustrate the exploitation of Tor's user anonymity for nefarious purposes, prompting law enforcement and governments to seek methods to block or monitor illicit Tor activity (Koch, 2019; Lee et al., 2016).

While Tor effectively conceals user identity, location, and activities, it cannot fully hide the network traffic generated during its usage. Traditional methods to prevent Tor usage, such as blocking the public IP addresses of Tor relays and directory authorities, have proven inadequate due to the use of Tor bridges and pluggable transports (Lee et al., 2016; Mrphs, 2016). Nevertheless, simply blocking Tor is not an ideal solution, as this would also restrict access for legitimate users who rely on Tor for privacy protection. As a response to these challenges, Network Traffic Classification (NTC) techniques have emerged as a potential solution for detecting and monitoring Tor traffic. These advanced approaches aim to identify, monitor, and potentially manage Tor traffic, enabling its use by legitimate users who require Tor's privacy-enhancing features.

NTC encompasses a range of techniques for analysing packets transmitted across a computer network. Simple classification methods rely on port numbers and presumed protocols (Reynolds & Postel, 1992), but these techniques are often unreliable due to the use of random ports by many applications. More advanced classification techniques, such as Payload-based or Deep Packet Inspection (DPI), examine characteristic signatures or patterns of strings found in the payload packets of specific applications. Despite providing accurate results in some contexts, DPI fails when applied to encrypted packets. Consequently, statistical-based classification methods that leverage Machine Learning (ML) techniques have gained popularity in recent years (Cuzzocrea et al., 2017; Dainotti et al., 2011; Lashkari et al., 2017; Mahdavi, Hassannejad, et al., 2018). Building upon this trend, most ML-based classification techniques rely on flow-based features arising from connection characteristics in the communication (A. Moore et al., 2013).

These methods have been applied to various objectives, including malware detection (Tran et al., 2020; Yeo et al., 2018), network intrusion detection (Gogoi et al., 2012; Pektaş & Acarman, 2019), botnet detection (W. Wang et al., 2020), and protocol-based classification (Bhargava et al., 2013; Schatzmann et al., 2010; Vinayakumar et al., 2017).

In this study, we aim to classify Tor traffic using ML while adopting a novel strategy that does not rely on traffic flow characteristics. Instead, our approach focuses on features derived solely from the encrypted payload. Although this may appear similar to DPI, our method operates effectively on encrypted packets without compromising the confidentiality of the packet payload. In fact, the features considered in this innovative approach are the undecrypted contents of the encrypted payload. This poses a challenge to the prevailing encryption theory, which posits that no information should be learnable from encrypted data. However, due to the differences in encryption mechanisms employed by Tor and nonTor networks, some distinctive characteristics may be inadvertently exposed and allow for effective classification. Capitalising on this potential vulnerability, our approach focuses on the following method. We measure frequencies of possible hex digits (0-9, a-f) found in the encrypted payload in the TCP or TLS/SSL layer. These counts are converted to ratios to ensure normalisation for different payload sizes. The resulting hex character statistics-based features are examined using statistical and ML approaches, to demonstrate their effectiveness as generic attributes for distinguishing between Tor and nonTor network traffic.

1.2 Thesis Statement

This thesis presents a novel approach to classifying Tor traffic using ML, focusing on features derived solely from the encrypted payload, rather than traditional traffic flow characteristics. By analysing the statistical frequencies of hex digits in the encrypted payload and employing advanced ML techniques, this research challenges the prevailing encryption theory that encrypted data should reveal no observable patterns. Our findings demonstrate the feasibility of distinguishing between Tor and nonTor network traffic based on unique characteristics in their encryption mechanisms, offering a significant contribution to the field of network traffic classification and cybersecurity.

1.3 Problem Statement

The benefit of maximising users' privacy through Tor incurs a negative impact in the prospect of potential abuse of this anonymity. Malicious actors may exploit Tor's anonymity features to engage in illegal activities, either by accessing regular websites or hidden services hosted on the Tor network, where a majority of illicit activities occur. Consequently, monitoring connections to the Tor network becomes essential, especially in cases where users may have malicious intent. Identifying illegal activities within Tor traffic poses a considerable challenge; however, tracking who is using Tor is more feasible. Several Tor traffic detection methods with high success rates have been reported but each comes with limitations. Tor packet delivery, which involves routing traffic through multiple proxy servers worldwide, is slower than regular traffic. As a result, many studies on Tor traffic classification rely heavily on flow-based features of traffic flow. However, these can be unreliable due to asymmetric routing, and the computation of multiple packets for feature extraction can introduce processing delays. Moreover, while Tor handshake traffic reveals distinct server names compared to nonTor traffic, this method relies on the presence of TLS/SSL certificates, which are only visible at the beginning of a connection (Lapshichyov & Makarevich, 2019).

To address these limitations, we propose a novel classification method for Tor traffic based on hex character statistics analysis. This approach is particularly intriguing because Tor achieves privacy for Internet browsing using multi-layer encryption, in contrast to the single-layer encryption employed by nonTor traffic. The application of multiple layers of encryption and distinct encryption mechanisms in Tor may influence the distribution of data within its payload, defying the traditional encryption theory that presumes encrypted data discloses no useful information for attackers. Despite this, our goal is to investigate whether these characteristics can be leveraged as distinguishing features for the automated identification of Tor traffic, utilising ML methods. Our proposed method employs packet-based features, requiring only a single packet, and demonstrates the ability to accurately distinguish Tor and nonTor network traffic types. By addressing the limitations of existing traffic classification techniques, our research

contributes significantly to the development of more effective methods to identify and monitor Tor traffic, and thereby, may lead to stronger Internet security.

1.4 Research Questions and Hypotheses

As previously mentioned, one goal of encryption is to ensure that ciphertext does not reveal information about the plaintext, even when an attacker has prior knowledge (Katz & Lindell, 2020). However, for the purpose of privacy, the encryption processes employed in Tor are more complex than those used in nonTor networks. These unique encryption techniques, along with other distinct networking properties of Tor, could potentially influence the distribution of data within its payload. This observation leads to the formulation of the first research question and the corresponding null hypothesis.

Q1: Can we distinguish Tor from nonTor traffic based on their encrypted payload? To address this question, we consider the following null hypothesis: H_{01} : There is no difference between Tor and nonTor traffic in terms of encrypted payloads.

Building on the first research question and hypothesis, we seek to explore the effectiveness of a proposed approach in accurately distinguishing Tor traffic. Traditional methods, especially those relying on flow-based features, need multiple packets from the start to the end of a flow to compute features. This dependency can sometimes cause unexpected timeouts, leading to process delays. In light of these challenges, our aim shifts towards providing a more data-efficient means of identifying Tor traffic. Consequently, we formulate the second research question and the corresponding null hypothesis.

Q2: Can we distinguish Tor from nonTor traffic using the encrypted payload in a data-efficient manner? In addressing this question, we consider the following null hypothesis: H_{02} : A single encrypted payload cannot be used to identify Tor traffic.

1.5 Objectives and Scopes of Research

The research background and problem statement set the direction for identifying and classifying Tor traffic, with an emphasis on investigating the unique characteristics of encrypted payloads. By utilising statistical analysis and ML techniques, our aim is to develop a novel, robust and effective method for distinguishing between Tor and nonTor traffic, potentially enhancing network security and aiding law enforcement efforts in monitoring Tor network activities. In light of these goals, we present our research objectives as follows.

Research Objectives:

1. Develop an automated method to extract features based on hex character statistics from both Tor and nonTor public and private datasets.

2. Study the differences in hex character statistics between Tor and nonTor encrypted payloads based on statistical methods for understanding their underlying statistical patterns

3. Evaluate the potential of classifying Tor and nonTor traffic based on features derived from hex character statistics using ML techniques.

4. Propose a novel payload size-independent approach for data efficient Tor and nonTor traffic classification.

Research Scopes:

1. To address the first objective, we deploy a custom-written script to extract encrypted payloads from raw traffic data, yielding four sets of features based on hex character statistics essential for further analysis.

2. To address the second objective, we conduct a comprehensive statistical analysis, encompassing both descriptive (visualisation) and inferential methods, to compare hex character statistics-based features between Tor and nonTor encrypted payloads, highlighting potential differences in hex character patterns.

3. To address the third objective, we experiment with various ML models, including Decision Tree (DTs)-based, Random Forests (RFs) and k-Nearest-Neighbours (kNN)-based algorithms, to assess their performance in classifying Tor and nonTor traffic using character analysis.

4. To address the fourth objective, we investigate the correlation between features and payload size to ensure the proposed methodology is robust against payload size variations.

1.6 List of Publications

1. Choorod, P., & Weir, G. (2021). Tor traffic classification based on encrypted payload characteristics. *2021 National Computing Colleges Conference (NCCC)*, 1–6.

2. Choorod, P., Weir, G., & Fernando, A. (2024). Classifying Tor Traffic Encrypted Payload using Machine Learning. *IEEE Access*, doi: 10.1109/ACCESS.2024.3356073..

1.7 Thesis Outline

This thesis presents a novel quantitative analysis grounded in hypothesis testing to illustrate the potential benefits of encrypted payload analysis for NTC. The following thesis chapters appear as follows:

- **Chapter 2** is divided into two parts. The first part provides an overview of the principles underpinning this thesis, equipping the reader with the necessary background to understand the remainder of the thesis. This includes the fundamentals of cryptography, protocols and their encryptions, an overview of the Tor network, character analysis, statistical analysis, and the ML approach. The second part of Chapter 2 reviews previous works in NTC, discussing various techniques, their benefits, and drawbacks. The chapter then narrows its focus to works most relevant to this thesis, specifically NTC in Tor.

- **Chapter 3** consists of two main sections. The first section presents the research methodology, including an overview, research framework, and descriptions of dataset collection, data preprocessing, statistical analysis, and ML approaches. The characteristics of our research tools and validation approach are detailed here. The second section describes the practical application of the methodology, covering data collection, raw data handling, and preparation of the system environment.

- **Chapter 4** presents a comprehensive statistical analysis of sample Tor and nonTor traffic, incorporating both descriptive and inferential statistics. This chapter offers an interpretation of the statistical results, highlights the significant findings, and discusses the implications of these outcomes in the context of the research problem.

- **Chapter 5** details the experiments conducted in Tor traffic classification, covering the selection and evaluation of various supervised classifiers. This chapter discusses the performance of each classifier, their advantages and limitations, and the impact of feature selection on the classification process. Additionally, it provides an interpretation of the classification results, highlighting the significant findings and their implications for the overall research objectives.

- **Chapter 6** examines the significant findings and their connections to the research questions and hypotheses. These findings offer crucial evidence in support of encryption theory, asserting that various network data types should not leak identifiable information and validating the classification tasks' success. This chapter provides a summary of the thesis findings, a discussion of the results, and an evaluation of the study's limitations. Finally, it highlights the contributions of this work and suggests directions for future research.

Chapter 2

Related Work and Literature Review

This chapter aims to provide the reader with a comprehensive account of the present research by addressing two major topics: an overview of the underlying technologies employed in this work and a survey of related work on NTC, with an emphasis on the Tor network.

Our study focuses primarily on the characteristics of encrypted packets in both Tor and nonTor networks, which heavily rely on encryption and encrypted protocols. We begin with an overview of encryption in the field of cryptography. A brief history of cryptography, an explanation of cryptographic principles, and an overview of various types of cryptography will also be presented. Encrypted packets are created according to specific encryption protocols, which we will further explore in this section. After discussing relevant aspects of encryption, we shift our focus to the Tor network, providing an overview of its functionality, usage, and importance. Following that, we delve into the technical aspects of the Tor network, including its construction, packet encryption, and successful delivery of data to the destination. In the subsequent section, we showcase the core methodology of this study which is character analysis. This has consistently proven useful in highlighting features in a wide variety of texts and our chosen approach to illuminating characteristics of encrypted Tor data packets.

Our proposed hypotheses associated with the Tor are addressed by two processes: the first involves statistical analysis, while the second explores the use of ML techniques. Accordingly, we will present these two analyses in this section. The latter part of this

chapter reviews related work on NTC, encompassing previous research on Tor and nonTor protocols. We will examine the methods employed, their strengths and weaknesses, and how these aspects contributed to the development of our approach.

2.1 Cryptography

Cryptography is ubiquitous in the digital world, pervading various aspects of our lives, from Internet applications to computational storage. It plays a crucial role in safeguarding sensitive data. Encryption is a specific technique within cryptography that aims to provide high levels of security, preventing information leakage from encrypted messages. Ideally, distinguishing between encrypted messages of identical length should be impossible, as per the fundamental principles of cryptographic design, which requires that no information about the original content should be revealed. This presumption represents a significant challenge in our research. Two of our null hypotheses reflect this view by stating that there should be no observable distinction between Tor and nonTor traffic in terms of encrypted payloads (H_{01}), and that a single encrypted payload should not be sufficient to identify Tor traffic (H_{02}). This section highlights the significance of robust cryptographic design and delves into other aspects of cryptography.

2.1.1 Overview

While some information is accessible to the public and not considered private, sensitive data such as passwords, personal information, and credit card numbers, or secret military communication, require confidentiality. In these cases, cryptography is used to disguise the original message and ensure secure transmission. Cryptography involves two main processes: encryption and decryption. Encryption utilises various techniques to transform an understandable message, called *plaintext*, into an incomprehensible message, called *ciphertext*. This transformation typically involves the use of a (secret) key, which is a critical component in ensuring that only authorised parties can access the original data. The key is used to encode the plaintext and, during decryption, the same or a corresponding key is used to decode the ciphertext back into the original plaintext,

ensuring the data remains secure from unauthorised access. Decryption is the reverse process of recovering the original message from the ciphertext, often utilising the same or a corresponding secret key. Figure 2.1 provides an example of the encryption and decryption processes.

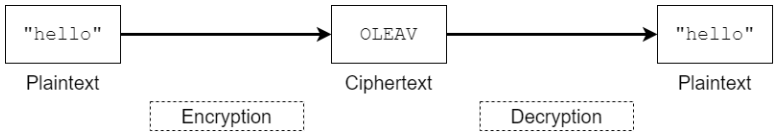


Figure 2.1: Encryption and decryption

Early evidence of data confidentiality dates back to ancient Egypt around 1900 BC, where a substitution method was used to replace unusual hieroglyphic symbols with customary ones, concealing the meaning of the writings (McDonald, 2015). Throughout history, many other cryptography techniques have been employed. However, this section will only cover a few early ciphers, summarised from Katz and Lindell (2020), along with their flaws before moving on to the development of modern cryptography as progress toward maximising security.

One of the earliest, simplest, and best-known ciphers is the *Caesar Cipher*, introduced by the ancient Roman emperor Julius Caesar to safeguard his military communications. The Caesar cipher is based on a substitution method by shifting the alphabet three positions forward. Caesar's ciphertexts are easily decipherable by sliding backwards three positions. To address the weakness of using the fixed key, the shift cipher introduced the use of a total of 26 potential keys in English letters, increasing the number of ciphertext possibilities. Unfortunately, a brute-force attack that tries every conceivable key in the encrypted string can break it. To overcome this weakness, the fixed shift cipher was improved to the mono-alphabetic substitution cipher, allowing for arbitrary one-to-one pairings, resulting in the key space size being 26! or around 2^{80}, making a brute-force attack impractical. Despite this improvement, statistical patterns of language can still be used to attack mono-alphabetic substitution ciphers using frequency analysis. To conquer such an attack, the poly-alphabetic shift cipher, also known as the *Vigenère cipher*, was introduced. It was known to be the strongest cipher at that time, but Charles Babbage, a British cryptographer, eventually broke it (Kahn, 1973; Schrödel, 2008).

The history of cryptography reveals that many adopted schemes have been compromised in time, whether quickly or after years of use. Interestingly, advancements in computing technology have made it easier to break codes, making it challenging to create unbreakable ciphers. Classical cryptography was considered more of an art than a science because it lacked complex techniques to keep messages secure, unlike modern cryptography, which uses sophisticated mathematical theories to improve encryption and decryption mechanisms. To meet the rigorous demands of modern cryptography, a cryptographic design must adhere to three principles: 1) Formal Definitions; 2) Precise Assumptions, and 3) Security Proofs (Katz & Lindell, 2020). These principles differentiate modern cryptography from its predecessors.

Principle 1 – Formal Definitions. Formal definitions serve as the foundation of modern cryptography design, facilitating precise explanations of potential threats and the security guarantees required to construct cryptographic schemes. A scheme is considered secure if it satisfies the definition; otherwise, it is deemed inadequate. Therefore, it is essential to establish a formal definition of a secure encryption scheme, as suggested by Katz and Lindell (2020, p.19):

> *Regardless of any information an attacker already has, a ciphertext should leak no additional information about the underlying plaintext.*

This definition can be represented by the mathematical expression below.

$$\Pr[C = c \mid M = m] = \Pr[C = c]$$

In this equation:

- Pr denotes the probability.
- M and C are random variables representing messages and ciphertexts, respectively.
- m is a specific value (message) that the random variable M can take on.
- c is a specific value (ciphertext) that the random variable C can take on.

The equation expresses that the probability of observing a specific ciphertext c given that the message random variable M takes the value m is the same as just the probability of observing that specific ciphertext c on its own. This encapsulates the essence of perfect secrecy, asserting that the ciphertext appears random regardless of the plaintext that generated it. An encryption scheme adhering to this principle demonstrates *perfect secrecy*. Another definition of perfect secrecy is based on the indistinguishability experiment, where an adversary intercepts a ciphertext and tries to determine which one of two known messages was encrypted. The encryption scheme is considered to provide *perfect indistinguishability* between messages if the probability of the adversary correctly guessing the encrypted message is exactly 50%. This is equivalent to a random guess, implying that the observed ciphertext provides no additional information to the adversary about which of the two messages was encrypted.

Principle 2 – Precise Assumptions. For a thorough validation of a cryptographic scheme's security, all assumptions must be explicitly stated. This facilitates meaningful comparisons between schemes based on distinct assumptions and allows for security proofs to be constructed.

Principle 3 – Proofs of Security. In contrast to the historical design-break-patch cycle of cryptography, contemporary cryptographic schemes rely on proofs of security that assure no vulnerabilities can be exploited. A proven secure scheme is one that adheres to precise cryptographic definitions and a specific set of explicitly stated assumptions.

Although the three-principle-based cryptographic approach provides a rigorous foundation for security, it does not necessarily guarantee real-world security (Katz & Lindell, 2020). In some measure, the present thesis seeks to exploit the limitation of such security schemes when applied to a specific aspect of computer networking.

2.1.2 Types of Encryption

The historical cryptography mentioned earlier relies on a single key for both encryption and decryption. When two parties wish to exchange a secret message, they must share the cryptographic key before communication commences. If a third party obtains the shared key, the encrypted message becomes vulnerable to exposure. To address this

issue, two separate keys are employed in the encryption and decryption processes. One key is designated for encryption, while the other is reserved for decryption. Generally, only the encryption key is made public, and the decryption key is privately held by the owner. Even if an adversary acquires the public key, the intercepted message remains secure as long as the private key is uncompromised. The first scenario, involving a single key, is referred to as *symmetric-key cryptography*, whereas the second scenario, which utilises two distinct keys, is known as *asymmetric-key cryptography* (Behrouz, 2022).

1) Symmetric-key cryptography: This approach utilises a single secret key for both encryption and decryption. Before initiating communication, the parties involved must exchange the shared secret key over a public communication channel. This key is securely stored and never shared again. The sender uses it to encrypt a message, converting it into ciphertext, which is then sent to the recipient. The recipient decrypts the message using the same key employed for encryption. This single-key usage scenario is also referred to as *private-key*, *shared-key*, or *secret-key cryptography* (Johnson, 2019).

Several well-known symmetric algorithms exist, such as Triple-DES (3DES or TDES), which employs a 56-, 112-, or 168-bit key. However, 3DES or TDES, which applies the outdated Data Encryption Standard (DES) three times to enhance security, is too slow. Consequently, it has been replaced by Advanced Encryption Standard (AES), which utilises a 128-bit block size and a 128-, 192-, or 256-bit key. According to various studies (Alenezi et al., 2020; Raigoza & Jituri, 2016), AES is regarded as the best encryption algorithm for both encryption and decryption. This evaluation is consistent with The National Institute for Standards and Technology (NIST)'s assessment, which took security, cost, and implementation criteria into account (Abdullah et al., 2017). Many widely used Internet applications, including WhatsApp and Facebook, as well as cryptographic protocols like TLS and Secure Shell (SSH), employ AES (Nistico et al., 2020).

2) Asymmetric-key encryption: This encryption method, also known as *public-key cryptography*, necessitates the use of two keys: a private key and a public key. The public key is distributed to anyone who wishes to encrypt data, while the private key is kept secret and used for decryption purposes. Major asymmetric encryption algorithms

include Rivest-Shamir-Adleman (RSA) (Kota & Aissi, 2022) and ElGamal (Rao, 2017; Tsiounis & Yung, 1998). In addition to encryption schemes, public-key cryptography has applications in network protocols, such as secret key exchange (e.g., (EC)DH) and digital signatures (e.g., (EC)DSA) (Heninger, 2022) for enhancing security tasks.

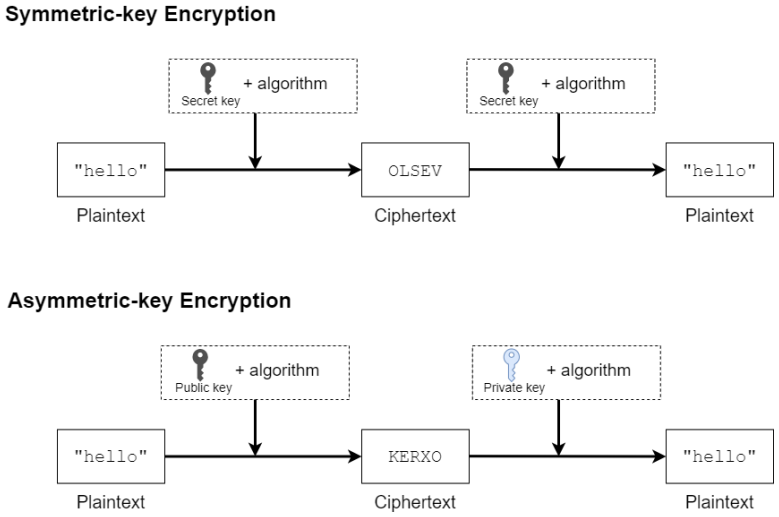


Figure 2.2: Symmetric and asymmetric cryptography

Figure 2.2 illustrates the two major types of cryptography: symmetric-key and asymmetric-key cryptography. Symmetric-key cryptography employs the same key for both encryption and decryption, making it quick and easy to implement. However, secure distribution of the key is necessary since anyone who intercepts it can decipher the encrypted message. On the other hand, asymmetric-key cryptography utilises separate keys for encryption and decryption, providing better security. However, it is more complex, time-consuming, and resource-intensive than symmetric-key cryptography.

As previously discussed, modern encryption schemes aim to achieve perfect indistinguishability, preventing adversaries from determining which of two messages was encrypted. However, in practice, entropy-based techniques can distinguish between encrypted and unencrypted messages (Tang et al., 2019; K. Zhou et al., 2020). Additionally, entropy is one of the parameters used to evaluate the robustness of cryptographic algorithms (Mousa et al., 2013; Mushtaq et al., 2017; Patil et al., 2016). Entropy measures the randomness or unpredictability of a message and can be used to assess

the security of encryption schemes. In this research, entropy will be utilised as one of the properties to investigate the characteristics of encrypted messages in two different network environments (Tor and nonTor), providing a comprehensive analysis of the security and performance of encryption schemes in real-world scenarios.

2.1.3 Entropy

Entropy, represented by H, measures the unpredictability or uncertainty associated with a random variable. Historically, Clausius introduced the concept in the physical sciences during the nineteenth century, applying it to describe the equilibrium of thermodynamic systems (Greven et al., 2014). This foundational idea was further expanded upon in 1948 by Claude Shannon, who is often referred to as "the father of information theory". He provided a formula for entropy, showing how it measures the uncertainty in information signals (Shannon, 1948). This concept forms the foundation for modern data compression and transmission techniques (Verdu, 1998). Moreover, Shannon emphasised its application in measuring the average information within letters of a text Shannon, 1951.

The mathematical formulation of Shannon's entropy for a discrete source with alphabet X is:

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{2.1}$$

where p_i denotes the probability of occurence of the i^{th} character.

Applications of entropy span diverse domains (Natal et al., 2021). In biology, it aids in discerning the disorder within symbolic DNA sequences (Das & Turkoglu, 2018). Linguists leverage entropy to probe into the richness of lexicons, and hydrologic engineers employ it to oversee various hydrological systems (Keum et al., 2017). In digital signalling, entropy measures the integrity of transmitted data. As an example, transmitting the binary sequence "11111111" and receiving "11110011" suggests bit-flips and potential data corruption during transmission (Lemons, 2013).

To illustrate the concept of entropy further, consider a bucket containing a mixture

of 100 green and red balls. The table below shows the entropy values for varying distributions of these balls:

Table 2.1: Entropy values based on different distributions of green and red balls in a sample set of 100 balls.

Distribution	H (bits)
10% Green, 90% Red	0.469
20% Green, 80% Red	0.722
30% Green, 70% Red	0.881
40% Green, 60% Red	0.971
50% Green, 50% Red	1
60% Green, 40% Red	0.971
70% Green, 30% Red	0.881
80% Green, 20% Red	0.722
90% Green, 10% Red	0.469

The table displays entropy values (H) for different green and red ball distributions. A 50:50 ratio has the highest entropy of 1 bit, indicating maximum unpredictability. As the distribution becomes more skewed (e.g., 90:10 or 10:90), the entropy decreases, showing a reduced level of unpredictability.

In cryptography, randomness is a crucial requirement, as information should not be vulnerable to adversary guessing. Entropy is utilised in the analysis of cryptographic constructions and key generation to measure the randomness or uncertainty of the characters appearing in the ciphertext, providing insight into the performance of cryptographic algorithms, as discussed in Patil et al. (2016) and Sanap and More (2021). More secure algorithms result in higher unpredictability or entropy, as they are less susceptible to guessing or frequency analysis. Similar to data encryption, data compression typically also leads to high entropy (Croll, 2013; Lyda & Hamrock, 2007; Mamun et al., 2015). The entropy in a ciphertext can be computed using Equation 3.4. In Chapter 4, entropy will be demonstrated as a means of estimating the randomness of encrypted payloads for two types of networks, Tor and nonTor, based on a single hex character.

A brief overview of cryptography has been presented, highlighting its primary objective of ensuring security and preserving confidentiality. This objective is evident in the widespread utilisation of cryptography for safeguarding data, with modern cryptography being integrated into various computer systems for numerous security applications. These include file encryption on disk, digital content protection (Diehl, 2012), blockchain technology (Yaga et al., 2019), secure payment card transactions, secure wireless traffic technologies via radio signals such as Wi-Fi, cellular and Bluetooth networks, encrypted web traffic over Hypertext Transfer Protocol Secure (HTTPS), and enhanced privacy for Internet users through the Tor network (Martin, 2017). The latter two examples of cryptographic applications, which are pertinent to this study, will be discussed in the next sections. This discussion will illustrate the critical role of cryptography in securely delivering data while also achieving additional confidentiality goals, such as authentication, integrity, and non-repudiation of origin.

2.2 Protocols and Their Encryptions

The research data in this study comprises a collection of network traffic traces, which are directly related to network protocols. Therefore, this section provides an overview of the Transmission Control Protocol/Internet Protocol (TCP/IP), which is the backbone of the Internet, and TLS, which enables secure communication of TCP/IP network traffic. A thorough understanding of these topics is essential for the proper execution of the data preprocessing step, ensuring the accuracy and validity of the conducted experiments.

2.2.1 TCP/IP Protocol Overview

The Internet is a global network that connects computers, and its fundamental purpose is communication. The underlying technology for Internet communication is the TCP/IP (Hunt, 2002), which is a reference model consisting of a suite of data communications protocols. A protocol is a standardised set of rules for exchanging data between electronic devices. The TCP/IP reference model governs how a particular computer is connected and transmits data via the Internet to other computers. The concept of TCP/IP

was developed in the 1970s by Vinton Cerf and Bob Kahn, and two years later, they published a paper (Vint & Kahn, 1974) outlining the Transmission Control Protocol (TCP) protocol, which provides all the transport and forwarding functions on the Internet (Leiner et al., 1997). The TCP/IP architecture is a system of connections between layers, each with its own set of protocols. The name TCP/IP is derived from the two primary protocols, TCP and Internet Protocol (IP), although many other protocols work in conjunction with them. The TCP/IP model was originally designed with four layers, but it is occasionally seen with a five-layer structure (with a physical layer below the link layer), which makes it simpler for computer scientists to analyse (Tanenbaum, Wetherall, et al., 2021). In this research, the focus is on the analysis of data operating at the top two layers, namely the application and transport layers. Therefore, the original four-layer TCP model, which encompasses these layers, will be discussed in this context.

The four TCP/IP layers, arranged from top to bottom, are named according to their functions as follows: Application, Transport, Network, and Link layers. This layering approach enables data to be transmitted down through the stack in a top-down fashion, with each layer performing distinct functions and communicating with the layers above and below it. The advantage of this layered structure is that any layer or service can be changed without affecting the others, making it easier to maintain and upgrade the network protocol (Fall & Stevens, 2012).To ensure proper delivery and communication, each layer of the stack adds control information called a *header* in front of the data to be transmitted. While the text-based headers of the Application Layer provide higher-level protocol-specific information, the fixed-size headers in the lower layers primarily guide the data's transmission across the network. Each layer considers the data sent from the adjacent layer as data and adds its own header to it. This process of adding a header at every layer is known as *encapsulation*. The architecture of the TCP/IP stack is depicted in Figure 2.3, along with the structure of additional data at each layer.

The following is an explanation of the primary functions of each TCP/IP layer, as well as the mechanism by which data is sent from the top layer to the bottom layer:

1. *Application Layer* is the topmost layer of the TCP/IP stack, where applications access network resources. This layer provides services to the end-users and enables

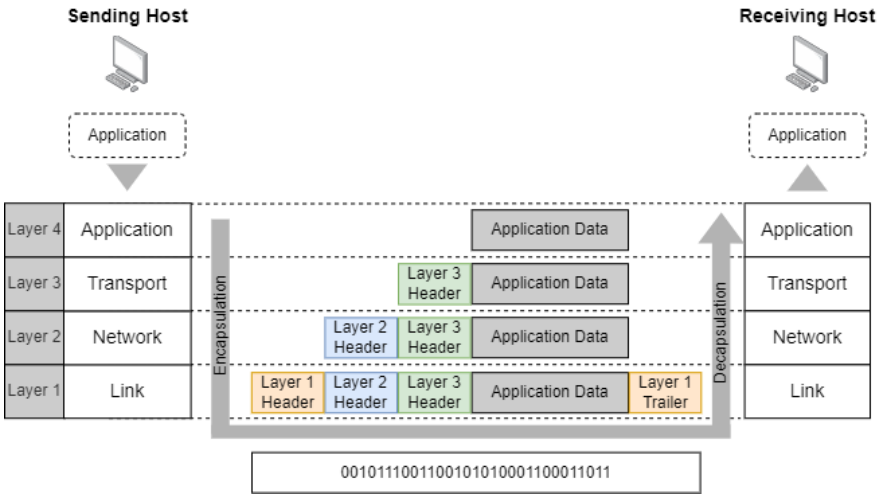


Figure 2.3: Data delivery from source to destination in the TCP/IP stack (adapted from Behrouz, 2022)

communication between different applications using standard protocols such as Hypertext Transfer Protocol (HTTP) for web browsing, File Transfer Protocol (FTP) for file transferring, Simple Mail Transfer Protocol (SMTP) for email, SSH for remote login, etc (Hunt, 2002). The protocols at this layer focus on how to represent, reconstruct, and interpret data. The actual data to be transmitted called the *application payload*, is a portion of a packet in network traffic and is the primary focus of this study. The payload structure will be discussed in the following section.

2. *Transport Layer.* The Transport Layer facilitates data transport through two crucial protocols, TCP and User Datagram Protocol (UDP). TCP is a connection-oriented protocol that ensures accurate and complete message delivery by providing a retransmission mechanism for missing data. In contrast, UDP is a simpler transport protocol that does not offer a retransmission mechanism for lost packets; it only guarantees data correctness, not completeness. Consequently, UDP prioritises transmission speed over data reliability. TCP is implemented when reliable message delivery is necessary, while UDP is employed when transmission delay is a more significant concern than occasional data loss (Hunt, 2002). For instance, email utilises TCP to ensure that recipients receive the same content as the sender.

On the other hand, streaming/real-time media protocols like Web Real-Time Communication (WebRTC) typically operate over UDP because a continuous data flow is preferred, and minor data loss has no significant impact on users. However, they can also be used with TCP if required (Santos-González et al., 2017).

At this layer, the initial encapsulation occurs, with a first *header* being added to the application payload (collectively referred to as *segments*). The header contains essential protocol information, such as source and destination ports, header length, flags, etc., which is necessary for the accurate control of packet delivery. Within the Transport Layer, one widely recognised parameter is *transport ports* or simply *ports*. Well-known ports were previously used to define specific applications and were considered crucial attributes for classifying application types. However, with the advent of dynamic ports, port-based classification has become obsolete (Nguyen & Armitage, 2008; Velan et al., 2015). In computer networks, a combination of the transport protocol, source and destination IP addresses, and source and destination ports form a 5-tuple. A series of packets sharing the same 5-tuple values is known as a *flow* (Nguyen & Armitage, 2008). Since it is presumed that each application possesses distinct statistical flow properties, such as packet sizes and time-based features (Lashkari et al., 2017), flow features have become vital attributes in NTC. This method follows the limitations of port-based and payload-based classification, enabling traffic classification without compromising users' privacy. However, the drawbacks of flow features have led to the development of our approach, which will be discussed in detail in Section 2.6.4.

3. *Network Layer.* The Network Layer is responsible for data addressing and routing, determining the optimal path within the network. IP is the prevailing protocol that manages packet delivery at this layer. Similar to the Transport Layer, the IP header is added to the Transport Layer's data (collectively referred to as *packets*). The most crucial information in the IP header includes the source and destination IP addresses, which identify the sending and receiving hosts.

4. *Link Layer.* Sometimes referred to as the *Network Access Layer.* This layer is

the lowest one and closest to the physical network. It provides an interface for connecting to network hardware. The widely used link layer protocol is Ethernet. The Ethernet protocol appends its header and an additional *trailer* to packets (colloquially known as *frames*) received from the Network Layer. The Ethernet frame then physically transmits data in the form of binary 1s and 0s, enabling a computer to understand and transmit it through physical media to the Ethernet network (Goralski, 2017).

The binary information from the sender's link layer is routed through several network devices until it reaches the receiver's link layer on the destination device. Before forwarding the data to the layers above, each layer removes its header, a process called *decapsulation*, leaving the data portion the same as that at the sender's corresponding logical level. The data delivery process is complete when the transmitted data reaches the same layer as the sender.

Network Packet Structure

As previously discussed, a network packet consists of two parts: the header and the payload. The header, located at the beginning of the packet, contains metadata in plaintext, which is essential for communication between the source and destination. The payload constitutes the actual data or application data of the entire transmitted message and can be in plaintext or ciphertext, depending on the need for secure data transmission. In the early days of the Internet, protocols transmitted data in plaintext, which was accessible to anyone who intercepted it. At that time, there were only a small number of people online, and mutual trust rendered encryption unnecessary. Later, as the Internet's popularity and range of applications grew, the need to secure online data communication became increasingly important. As a result, data encryption has evolved into a standard feature of data transmission.

Within each layer of the TCP/IP stack, there is a variety of alternative protocols for securing communications. Nonetheless, our focus will be on the initial encryption of application data that takes place at the Application Layer before being passed to the Transport Layer. Several encryption techniques are responsible for this process; however,

based on the datasets used in our analysis, we will only discuss the relevant encrypted protocols, such as TLS, SSH, and proprietary protocols, in the subsequent sections.

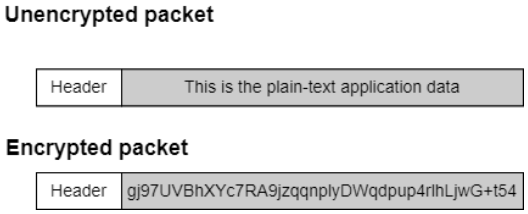


Figure 2.4: The structure of unencrypted and encrypted packets

Figure 2.4 illustrates the structure of both unencrypted and encrypted packets. Numerous Internet protocols, such as HTTP, SMTP, and FTP, utilise plaintext payload data. On the other hand, more recent secure protocols, including HTTPS, SSH, and other secure proprietary protocols, employ encrypted payload data. The content of the encrypted payload remains unreadable and is represented by network analyser tools in various formats such as binary, hex, and ASCII. When comparing these three representations, the binary format consists of only 0s and 1s, making it more difficult to read and write large values. In contrast, the hex format uses a base-16 numbering system with 16 characters (0-9, a-f), making it easier to read and write large values. Moreover, encrypted data often contains non-printable characters that do not map well to ASCII characters, complicating the analysis. In contrast, the hex representation can accommodate both printable and non-printable characters, allowing for a more accurate and comprehensive analysis of encrypted data. Consequently, this study analyses characters based on the hex format, as it is a suitable analysis method.

TCP Segmentation

To make network communication efficient, application data needs to be sent in smaller pieces. This is one of the responsibilities of the Transport Layer, called *segmentation*. Segmentation occurs with TCP but not with UDP. TCP divides the data received from the upper layers into transmittable segments, with each segment containing a sequence number, acknowledgement numbers, and other information that occupies the header information. These numbers allow the Transport Layer to accurately reassemble the message upon arrival at the destination, as well as identify and replace packets lost

during transmission. The size of these chunks is determined by the Maximum Segment Size (MSS), or the parameter of the maximum size of packets that can be sent over a network. Various factors, depending on the specific TCP stack implementation, affect MSS. TCP typically calculates the MSS based on the Maximum Transmission Unit (MTU) value managed at the Network Layer, but many implementations use segments of 512 or 536 bytes (Goralski, 2017). The process of breaking packets into smaller parts can also occur at the Network Layer and is known as IP *fragmentation*. The most common IP fragmentation takes place with UDP traffic but not with TCP traffic (X. Wang & Cronin, 2014). The reverse operation of segmentation and fragmentation on the receiving side is called TCP *reassembly* and IP reassembly, respectively.

This study makes use of the payload in the form of segments represented in hex format. As previously explained that the payload encrypted by various encryption protocols in the Application Layer is passed down to the Transport Layer. These encrypted payloads are then divided into smaller segments, as previously described, each with its own headers and payloads. It is these segments that are examined in this study.

2.2.2 TLS

The TLS protocol is essential for secure communication between applications over the Internet, providing end-to-end encryption. It offers authentication, confidentiality, and data integrity services, ensuring the security of data for all applications running above it. Initially designed to encrypt HTTP connections, TLS has since been extended to other Internet traffic types, including those used in the Tor network, which is the focus of this research. As TLS is implemented between the Application and Transport Layers, outgoing application data above the TLS layer is wrapped and encrypted using the TLS protocol, resulting in ciphertext generated by various TLS applications. This encrypted payload data will be analysed to determine the characteristics of randomised character distributions generated using different encryption schemes between Tor and nonTor networks.

TLS is a descendant protocol of Secure Socket Layer (SSL), which was first implemented by Netscape in 1994 (Ristic, 2015). The initial version of SSL, SSL 1.0, was

never released due to significant security bugs. SSL 2.0 was the first public release in 1995, followed by SSL 3.0, which was a complete redesign of the protocol and the final version of SSL. In 1996, the TLS protocol was introduced by the Internet Engineering Task Force (IETF) [1] as an improved version of SSL. Since then, TLS has undergone four releases: TLS 1.0 [RFC2246], 1.1 [RFC4346], 1.2 [RFC5246], and 1.3 [RFC8446]. The first two versions, TLS 1.0 (Dierks & Allen, 1999) and TLS 1.1 (Dierks & Rescorla, 2006), were vulnerable to dangerous flaws, leading to the release of TLS 1.2 (Dierks & Rescorla, 2008) in 2008. Currently, TLS 1.2 is the most widely used and considered reasonably secure. However, discovered attacks due to flaws in cryptographic ciphers and algorithms led to the significant improvement of TLS 1.3 (Rescorla, 2018) in 2018, which is the most recent and considered the strongest and safest version of the TLS protocol. At the time of writing, only TLS 1.2 and 1.3 are still active versions. Although the terms SSL and TLS are often interchanged or written as SSL/TLS due to their similar purposes, TLS is the more commonly used protocol, as SSL is no longer in use. Therefore, the term TLS will be used throughout this thesis.

To maximise security during deployment, TLS employs a combination of symmetric and asymmetric cryptography to maintain a fair balance between security and efficiency while ensuring secure data transfer. Asymmetric encryption occurs first during the TLS handshake process, where the client and server agree on new keys, called *session keys*, to be used for symmetric encryption in the TLS session communication. Once the connection is successfully established, TLS uses the pre-agreed symmetric or session keys to encrypt and transmit data.

As previously mentioned, TLS 1.2 and 1.3 are the currently utilised versions, and both appear in our datasets. The network traffic from the public source was captured in 2016 (Lashkari et al., 2017), while the private source was captured in 2020 (see Section 3.4.1). A thorough examination revealed that nonTor traffic of both datasets employs TLS 1.2 and 1.3, whereas Tor traffic of both datasets only uses TLS 1.2. This is consistent with the Tor Protocol Specification[2], which indicated that TLS support would be added to the Tor browser in 2020 (Dingledine & Mathewson, 2022). Consequently,

[1] https://www.ietf.org/

[2] https://github.com/torproject/torspec/blob/main/tor-spec.txt

TLS 1.2 and 1.3 will be addressed as the technology implemented in these datasets. The general processes of TLS will be discussed first, followed by a comparison of the two versions. We will then discuss the deficiencies of TLS 1.2, which led to the development of TLS 1.3.

TLS operates in two phases, each controlled by a separate protocol. The handshake protocol governs the first phase, which utilises asymmetric cryptography and digital certificates for authentication, while the record protocol controls the second phase, which employs symmetric cryptography for data encryption. The purpose of the first phase is to authenticate two parties using digital certificates so they can exchange a secret key to be used in symmetric encryption. The primary steps involve negotiating the protocol version and selecting a cryptographic algorithm defined in cipher suites. Once the agreed-upon algorithms are chosen, the parties authenticate each other using asymmetric cryptography and digital certificates, ensuring the authenticity of the parties involved. Following successful authentication, the two parties engage in a key exchange process, often using the Diffie-Hellman key exchange protocol, to collaboratively generate a shared secret key for session encryption without directly transmitting the key. This ensures that even if an eavesdropper listens to the entire handshake, they cannot derive the shared secret. With this shared secret established, the second phase can begin, focusing on encrypting bulk data. During this phase, transmissions from multiple data streams are encrypted collectively using the shared key. Upon receiving the encrypted message, the recipient checks for data modification using the Message Authentication Code (MAC) to ensure that the message has not been altered, thus maintaining data integrity. If no modifications are detected, the message is decrypted using the same symmetric secret key.

Both TLS 1.2 and 1.3 have two primary phases, but they differ in specifics. TLS 1.2 necessitates a full handshake, requiring two round trips to complete. In contrast, TLS 1.3 only requires one round trip, allowing the client to begin sending data immediately after the first round trip. This process is known as *TLS False Start* and reduces protocol round-trip latency. However, TLS 1.3 can further outperform with 0 RTT. Data from previously visited websites can be transmitted to the server along with the first message,

resulting in a round trip time of 0. TLS 1.3 employs 1 RTT or 0 RTT instead of 2 RTT, significantly increasing speed. The newer version not only enhances performance but also offers improved security. TLS 1.3 eliminates vulnerable ciphers such as CBC-mode and RC4, thus no longer supporting unnecessary or insecure ciphers. The Diffie-Hellman algorithm for shared secret key exchange, as used in TLS 1.2, is considered less secure than the Ephemeral Diffie-Hellman (EDH) key exchange protocol implemented in TLS 1.3, which generates a one-time key for the current network session. The key is discarded at the session's conclusion. If an attacker obtains both the private key and the encrypted message, the data remains secure. This feature, known as *perfect forward secrecy*, represents one of the most significant improvements in TLS 1.3.

Several important differences between the two versions of TLS are relevant to our work. For instance, the improved handshake process in TLS 1.3 results in fewer round trips and packet exchanges compared to TLS 1.2. Additionally, while both versions can utilize the Diffie-Hellman key exchange, TLS 1.3 mandates key exchange methods that provide perfect forward secrecy. In contrast, earlier configurations of TLS 1.2 allowed the use of RSA key transport, a mechanism that could compromise security if the server's private key was ever obtained. This shift ensures enhanced security in TLS 1.3, as the compromise of a long-term key cannot be used to decrypt past sessions. This understanding of the differences and their implications will be valuable during the packet investigation in Chapter 3.

2.2.3 Other Encrypted Protocols

Although most Internet applications use TLS for secure communication, other security protocols also provide robust safeguards for them. The use of encryption in other security protocols visible in the datasets, such as SSH and proprietary security protocols, will be discussed further below.

SSH

SSH (Barrett et al., 2009) is a secure remote administration protocol that enables users to execute commands on remote servers through the Internet. SSH operates at the

Application Layer and implements cryptographic techniques to secure communication and is typically executed over TCP (as defined in RFC 4251). Telnet, a formerly popular unencrypted remote connection service, has been entirely replaced by SSH. The SSH connection involves both symmetric and asymmetric encryption. The public and private key mechanisms are employed for key creation and exchange, while a shared key is used for data encryption within the SSH session. These processes bear resemblance to TLS, which is previously discussed in Section 2.2.2. While SSH and TLS share the objective of safeguarding user data from potential exposure, their communication fundamentals differ in certain aspects. For instance, SSH uses port 22, whereas TLS uses port 443. Additionally, SSH is utilised to execute remote commands on a server, while TLS is designed to establish secure information exchange. Unlike TLS, which implements a server certificate for client and server authentication, SSH uses a username and password to authenticate and create the connection.

Furthermore, SSH may also be utilised to encrypt data transferred through other network protocols such as FTP. By executing FTP over SSH, which is referred to as SSH File Transfer Protocol (SFTP), secure remote service can be enabled. This study analyses the encrypted segment payload of application data generated by the SFTP, which has been encrypted with the SSH protocol.

Proprietary Security Protocols

The protocols discussed above are commonly known as *standards* or *open protocols*, implying that they are open-source and can be used by anyone to build and produce their own products. The majority of applications use this type of protocol. However, some protocols are intended to be exclusive and cannot be used publicly. These protocols are referred to as proprietary protocols. They are developed and designed for a specific purpose by a single organisation. Typically, the owner enforces restrictions via patents and trade secrets and does not disclose the protocol's technical specifications. This lack of disclosure makes it challenging for outside parties to fully understand how these applications operate, and as a result, classifying their network traffic becomes difficult. As a proprietary protocol is implemented on the Application Layer, the encrypted

segment payload for analysis represents application data encrypted with proprietary security protocols. Two examples of proprietary protocols used in the datasets of this thesis are Spotify and Skype.

The reader has now gained a fundamental understanding of cryptography and network protocols, as well as how these two principles collaborate to provide Internet users with a sense of security. They also serve as a key component of the Tor network. The following sections will explain what the Tor network is, how it functions, and how it offers privacy. Knowledge of the Tor network will provide valuable insights into the characteristics of Tor and nonTor packets and their differences.

2.3 Tor

As the use of the Internet continues to grow, concerns regarding privacy have also increased. There are various methods available for Internet users to maintain their online privacy, including the use of VPN, proxy servers, mix networks, and onion routing (Dutta et al., 2022a; Li et al., 2013). Among these methods, Tor is one of the most widely adopted (Chakravarty et al., 2011; Platzer et al., 2020). and is renowned for its ease of use, requiring no technical knowledge, and being freely available for download (Shavers & Bair, 2016). This section provides an overview of Tor, discussing its nature, circuit-building process, and packet movement within the Tor network.

2.3.1 Tor Background

Tor is a popular tool for Internet users to maintain their privacy and anonymity. It operates as an overlay anonymous network situated in the Application Layer of the TCP/IP protocol stack. Tor is known as *The Onion Router* because it employs the *onion routing* technique to conceal user identities by encrypting packets multiple times and decrypting multi-encrypted layer packets in a manner akin to peeling off an onion's layers. Tor has grown in popularity, with over two million users and a daily traffic volume of above 200 Gbit/s (The Tor Project, 2021a). Tor is designed as a low-latency communication system to enable real-time use of interactive applications such as web

browsing, instant messaging, or SSH connections. This is in contrast to high-latency anonymity systems like Mixmaster and Mixminion, where packet transport can take approximately four hours on average, and sometimes up to several days (Annessi, 2014a; Hopper et al., 2010; Murdoch & Zieliński, 2007). In addition, Tor is appealing to users because it is open-source software, available for free download, and easy to use. Tor has grown in popularity, currently serving an estimated two million users and processing a daily traffic volume of 200 Gbit/s (The Tor Project, 2021a).

The origins of Tor can be traced back to the mid-1990s when mathematician Paul Syverson and computer scientists David Goldschlag and Mike Reed, who were working for the The United States Naval Research Laboratory (NLR), sought to develop a method for anonymous Internet browsing. They developed a prototype of onion routing, which combined encryption and private circuits. In the early 2000s, the onion routing concept was refined into a new, next-generation design and implementation, which evolved into the Tor project. The project was launched in 2002 and made publicly available in 2003 (Dutta et al., 2022b). The Tor development team released the formal design specification for the Tor network Generation 2 in 2004 (Dingledine et al., 2004). Since 2006, Tor has been maintained by a non-profit organisation known as the Tor Project[3].

Tor has become an essential tool for individuals who seek privacy in their online activities as their identities are completely concealed while accessing the Internet via the Tor network. General Internet users utilise Tor to prevent their Internet information from being traced by their ISP while they are online. Safe communication is necessary between reporters and whistleblowers, and Non-governmental organizations (NGOs) workers can establish remote connections to their organisation's websites more securely without revealing their true identities. Some activist groups use Tor to promote civil liberties online while shielding their identities from government monitoring of suspicious websites. However, the potential misuse of Tor by malefactors has turned it into a double-edged sword. Various illegal services, such as drugs, fraud, counterfeiting, weapons, terrorism, child abuse, and access to other illicit content, including stolen data, exist because of Tor (Faizan & Khan, 2019; Guitton, 2013; Monk et al., 2018; Owen & Savage, 2015).

[3]https://www.torproject.org/

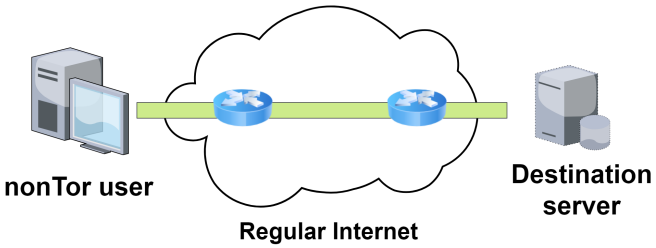
Cybercriminal activities enabled by the Tor network, such as the Silk Road darknet marketplace, the Mevade Botnet, the ChewBacca malware (Mirea et al., 2018), and Cryptowall 2.0 ransomware (Yetter, 2015) are notorious examples. Consequently, there is a need for law enforcement or governments to develop methods to block the illicit usage of the Tor network (Koch, 2019; Lee et al., 2016). In response to this, several research studies have attempted to explore methods for detecting Tor traffic, and these will be discussed in the Related Work section in 2.7.

In addition to the Tor browser, Tor provides a range of other services, such as Nyx for SSH connections, the Metrics Portal for Tor statistics, Tails for using Tor in amnesic and live mode, and onion services or hidden services for hosting .onion domain names. Despite its numerous advantages, a significant number of research has identified weaknesses in Tor, particularly its low latency, which can be passively attacked by monitoring the timing and volume of traffic entering and exiting the Tor network, leading to user de-anonymisation (Annessi, 2014b; Karunanayake et al., 2020; Sun et al., 2015). Although many studies have explored methods to compromise Tor, most of these approaches rely on identifying Tor traffic. To expose Tor users' online activities and identities, a substantial number of Tor nodes must be hosted and analysed by specialists, which can be a time-consuming endeavour. In certain instances, Tor users' identities were de-anonymised due to human errors, such as accidentally revealing their actual email addresses. It is worth noting that the Tor Project website specifies that the correct way to write Tor is with an uppercase 'T' followed by a lowercase 'o' and 'r,' rather than all uppercase letters like 'TOR' as is commonly misunderstood (The Tor Project, 2021b).

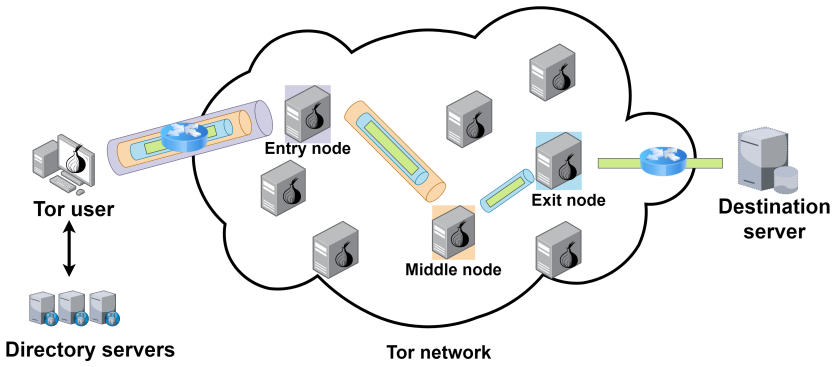
2.3.2 Tor Network

The Tor network is built on the concept of onion routing. The underlying principle of onion routing is that instead of a client communicating directly with a server, the connection is routed through several nodes. Figure 2.5 illustrates the Tor network compared to the conventional Internet. When a user accesses a website on the standard Internet, the browser typically establishes a direct connection to the destination, as

depicted in the packet form on the left side (a). In contrast, within the Tor network, packets are routed to the destination through multiple nodes, with each relay modifying the packets along the communication path.



(a) Normal Internet



(b) Tor Network

Figure 2.5: Comparison of connections in normal Internet and Tor network

The Tor network is comprised of numerous volunteer nodes worldwide that contribute bandwidth to form Tor virtual circuits. Approximately 7,000 active nodes contribute to the Tor network daily, as reported in The Tor Project (2021a). As more volunteers support Tor by running a Tor relay, the network becomes faster, more robust, more stable, and more secure for its users, since having more relays is generally advantageous. However, the current Tor network is relatively small compared to the number of people who need to use it. To help run a relay, individuals must have technical expertise and dedication. The Tor Project website[4] provides extensive information to guide relay operators in setting up their systems. Responsibilities include ensuring the relay runs on a server available 24 hours a day, seven days a week, and allocating at least 16 Mbit/s (Mbps) of upload and download bandwidth.

[4]https://community.torproject.org/relay/

Typically, Tor is made up of four main components;

1) Tor Client: The client launches the Tor browser on the local machine to connect to the Tor network.

2) Onion Routers (ORs): ORs (also known as routers, nodes, or relays) carry data between the client and the destination through a series of routers, which are three by default, consist of the *entry node*, *middle node* and *exit node*. At each router, packets are modified through encapsulation and decapsulation using the TLS protocol, which will be further explained in the subsequent section.

3) Destination: Generally, this refers to any servers that a Tor client requests access to.

4) Directory Servers: Also known as *Directory Authorities*, which deliver signed directory documents containing a list of signed server descriptors and onion router information, such as public keys and the status of each router. Consequently, clients can promptly receive real-time updates on the Tor network's status. Nine directory authorities are hard-coded into the Tor software, enabling clients to obtain information about onion routers to establish Tor circuits.

These components enable Tor communication operates smoothly. The process begins with Tor traffic being encrypted in layers (like layers of an onion) at the Tor client. This layered encryption traffic is then routed through a Tor circuit, which is created by randomly selecting three ORs (Onion Routers) by default, using information from the Tor directory service. It's worth noting that increasing the number of nodes may result in increased latency rather than improved anonymity. At each relay in the Tor circuit, one layer of encryption is peeled away, revealing the next relay's IP address, but not the final destination. This process continues until the traffic reaches the last node, the exit relay. Here, the final layer is decrypted exposing the destination ip address, and the original data is sent to its intended destination. The user's original IP address remains hidden, thereby preserving their anonymity. Upon reaching the destination, a response is triggered by the reverse process and returned through the same Tor circuit until the requested data is received and decrypted at the originating Tor client. The steps of Tor traffic communication will be discussed in more detail in the following section.

2.3.3 Tor Traffic Communication

Tor uses a different approach to traditional communication methods. Rather than transmitting traffic in its original packet format, Tor dispatches data in fixed-sized cells, enhancing anonymity. For data transmission within the Tor network, circuits must be established first. Setting up these circuits involves various encryption and decryption operations. Tor uses both symmetric and asymmetric encryption techniques, including public-key ciphers, to maintain security and privacy. This is similar to the TLS protocol, but with more complex operations.

Tor Cells: The Tor network structures its traffic into fixed-size cells, each containing 512 bytes, with a header and a payload. The encryption of Tor traffic does rely on the TLS protocol for the initial connection and the basic communication. Beyond that, Tor adds an onion-like layering of encryption, especially when establishing circuits. Tor cells can be classified into two categories: control cells and relay cells. Figure 2.6 illustrates the structure of both types of Tor cells. The headers of all cells are not onion-encrypted, allowing subsequent Tor routers to process them; however, the remaining cell components are encrypted in the characteristic onion-like fashion.

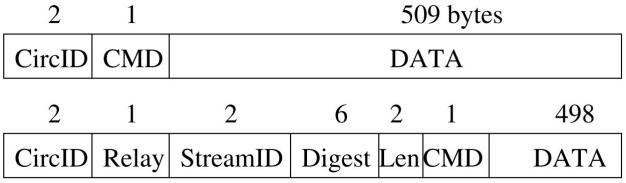


Figure 2.6: The structure of control (above) and relay cells (below) (Dingledine et al., 2004; Karunanayake et al., 2021; Saputra et al., 2016)

Each Tor control and relay cell contains a 2-byte circuit identifier (CircID) in its header, denoting the circuit to which the cells belong. This design is essential since the Tor network multiplexes multiple TCP streams across each circuit to enhance efficiency and confidentiality. Furthermore, a 1-byte command (CMD) is included to identify the cell type and its corresponding function. Control cell commands encompass padding (to pad packets into fixed sizes), create (to establish a connection), created (to signify a connection has been formed), and destroy (to dismantle a circuit). In contrast, relay

cells consist of an 11-byte header in addition to a control cell, encompassing a stream identifier (StreamID), a cryptographic integrity verification mechanism, and the length of the relay payload (Len). Relay commands include relay data, relay begin, relay end, relay teardown, relay connected, relay extend, relay extended, relay truncate, relay truncated, relay sendme, and relay drop.

Tor network data is typically encapsulated into uniform 512-byte cells. When traffic congestion occurs, cells may be amalgamated or divided into multiple MTU-sized packets and a singular non-MTU-sized packet. Consequently, the size of IP packets transmitted through the Tor network varies and appears random over time, diverging from fixed or MTU sizes (Ling et al., 2012).

Tor Circuit Construction: For communication within the Tor network, the construction of a Tor circuit is a prerequisite, as illustrated in Figure 2.7. This process is initiated when a user, Alice, requests a web page using the Tor Browser. While Alice interacts with the browser interface, the underlying Tor software manages the intricacies of circuit creation.

Initially, the Tor client requests a list of relays from the directory servers. The software then automatically constructs the Tor circuit by randomly selecting three Tor routers, sending a control 'create' cell to the first relay (OR1) in the circuit. In response, OR1 dispatches a control 'created' cell back, indicating the establishment of the initial relay connection.

To extend the circuit further, the Tor client sends a relay 'extend' cell to OR1 with the address of the second OR (OR2). OR1 then forwards a control 'create' cell to OR2. Upon receiving a control 'created' cell from OR2, OR1 packages it into a relay 'extended' cell, returning it to the Tor client. This sequence ensures the integration of two ORs into the circuit. Typically, the Tor software adds three ORs to the circuit, so the procedure is repeated until the third relay is incorporated, designating it as the circuit's exit node.

Once the circuit is established, a TCP stream is launched. The Tor client sends a relay 'Begin' cell through the circuit up to the final relay. At the exit node, a three-way handshake is initiated with the intended destination web server. Upon completing the TCP handshake, the exit node sends a relay 'Connected' cell through the circuit. This

fully constructed circuit then manages the transmission of data requests and responses. When the communication session is over, the Tor client sends a control 'destroy' cell to each OR in the circuit, instructing each relay to terminate the connection.

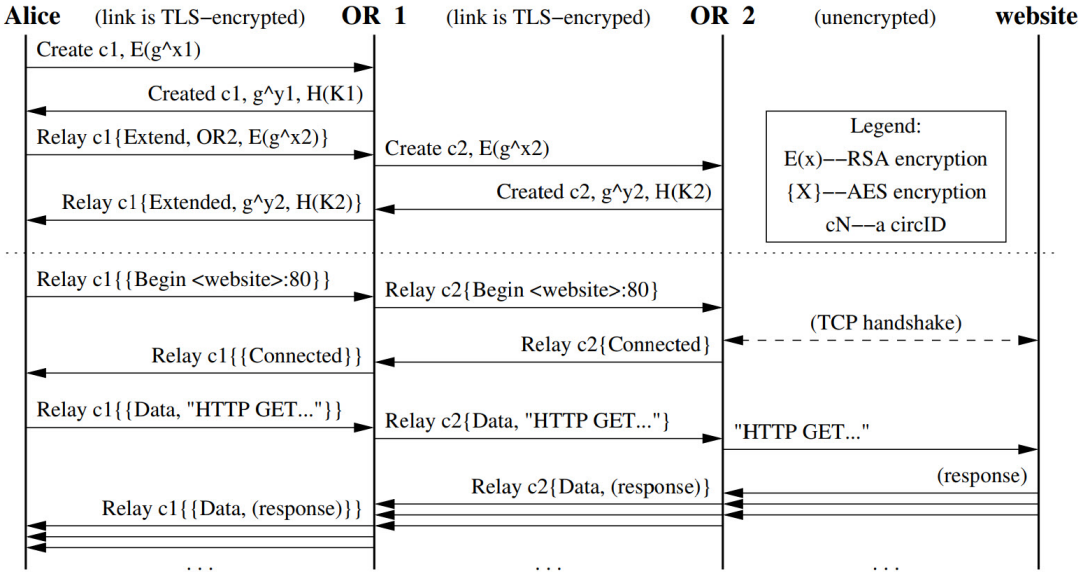


Figure 2.7: Constructing a Tor circuit using two relays by browsing a web page (Dingledine et al., 2004; Saputra et al., 2016)

Tor Traffic Delivering: In a normal network where a secure protocol is used, data encryption happens only once at the sending machine, and the packets are securely routed to the best path. However, the Tor network's data delivery mechanism is different. It involves a circuit creation process (discussed above) and several encryption and decryption repetitions (discussed below).

Figure 2.8 illustrates the delivery of encrypted packets in the Tor network. In the Tor circuit, the Tor client possesses three pre-established keys corresponding to the selected relays. First, the IP address of the Exit relay C is incorporated into the innermost layer, encrypted with key C. Subsequently, the IP address of the Middle relay B is added, and the middle layer is encrypted using key B. Finally, the outermost layer is encrypted with key A, following the addition of the IP address of Entry relay A. The multilayer encryption is completed and transmitted through the established circuit.

Upon reception by Entry relay A, the outermost layer is decrypted using the previously

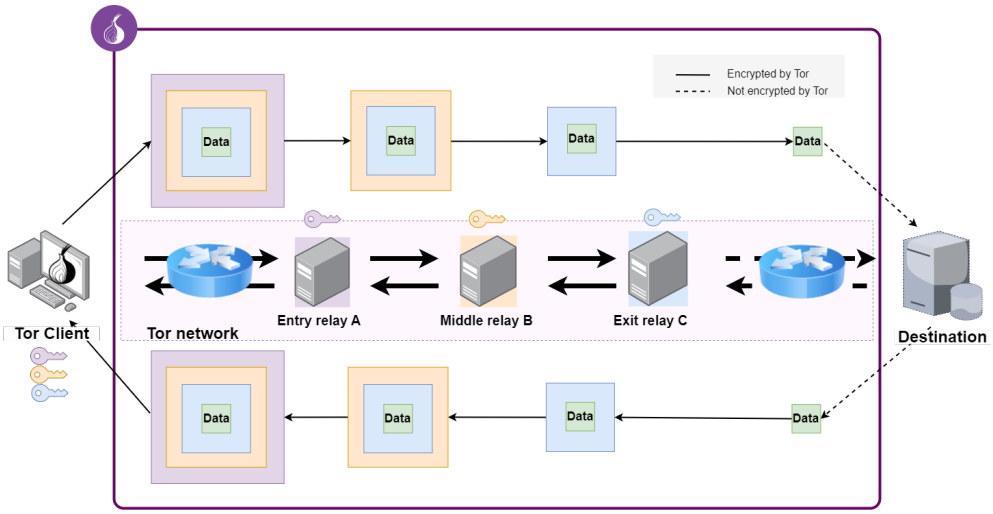
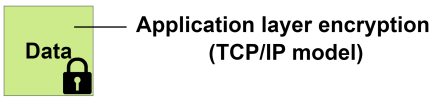


Figure 2.8: Encrypted packets delivering in Tor network

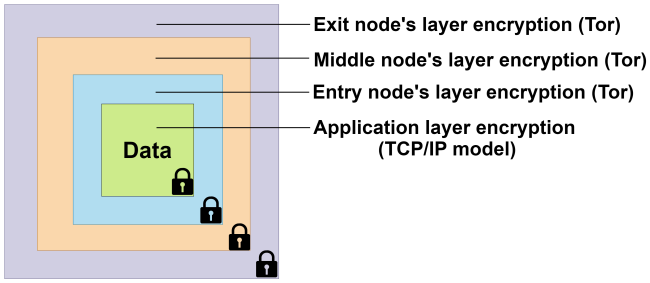
agreed-upon key with the Tor client, unveiling the next hop, or Middle relay B's IP address. Relay A subsequently forwards the remaining packet to Middle relay B. Similarly, the outermost layer is decrypted using the key previously established with the Tor client, revealing the next hop, or Exit relay C's IP address. Relay B then forwards the remaining packet to Exit relay C. The Exit relay C repeats the process, decrypting the outermost layer with the agreed-upon key, revealing the original packet as it appeared at the Tor client before onion-like encryption, and finally transmitting it to the destination. This final decryption at the exit node is not facilitated by Tor, as Tor does not provide end-to-end encryption. Consequently, if a user seeks enhanced security for their data after exiting the Tor network, employing encryption protocols such as HTTPS for the Tor client is essential.

At the destination, the process is repeated in reverse order, utilising the same circuit relays. At each node, response packets are individually encrypted with their respective keys. When a packet reaches the Tor client, all encrypted layers are decrypted using the shared keys of A, B, and C, exposing the original packet from the destination. This Tor-based traffic communication ensures that each router communicates solely with its adjacent nodes. For instance, the entry node is only aware of its communication with the Tor client and the intermediate node, remaining oblivious to other nodes (e.g., the exit node and the destination) along the message's path.

2.3.4 Tor Encrypted Payload



(a) Normal Internet



(b) Tor Network

Figure 2.9: Comparison of encrypted payload structures in nonTor (a) and Tor networks (b)

In Figure 2.9, a comparison is presented between the number of encryption layers used for outgoing and incoming packets in the nonTor and Tor networks. The figure illustrates that packets at the Tor client are encrypted three times, resulting in three encrypted layers, while packets at a nonTor client (using a secure protocol) are encrypted once, resulting in a single encrypted layer. The number of encryption layers and variables used in the encryption algorithms and key generation processes of Tor and nonTor can differ, leading to variations in the characteristics of the ciphertext. These distinctions may provide valuable insights into the security of encryption schemes used in various network environments.

In summary, Tor employs the TLS protocol to secure its communication. It uses asymmetric cryptography, especially the Diffie Hellman and RSA algorithms, for public key exchange. Historically, 1024-bit keys were used, but such key lengths are now viewed

as insecure. Today, it's recommended to use a security parameter of 2048 bits or more for RSA[5]. For bulk data transmission, Tor utilises symmetric cryptography, specifically 128-bit AES in Counter (CTR) mode, to encrypt and decrypt cells. This implies that any traffic originating from a Tor client carries a triple-layered AES encryption, with each layer specifically corresponding to the entry, middle, and exit relays in the Tor circuit. Furthermore, Tor employs several other cryptographic schemes for various operations, including path selection and congestion handling. While these schemes are not the focus of this research, further information can be found in AlSabah and Goldberg (2016), Dingledine et al. (2004), and Martin (2017) and the *Tor Protocol Specification*[6].

We have presented background information on computer security-related topics, which should help readers gain a solid understanding of these subjects and comprehend the subsequent steps of our study. Our focus now shifts to character analysis, which is the primary methodology of our research. We have chosen this approach based on its extensive research and proven accomplishments, and we will provide further justification for our decision.

2.4 Character Analysis

Character analysis constitutes the fundamental concept of the methodology employed in this study, originating from textual analysis. The advent of the Internet has led to exponential growth in online data, including emails, social media posts, publications, and more. Textual analysis is instrumental in scrutinising texts and facilitating users in efficiently obtaining relevant information. To enable machine processing for knowledge and insights extraction, the unstructured data within these documents must be converted into structured numerical data. Techniques such as Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) (Kowsari et al., 2019) aid in transforming text sentences into numeric vectors. These techniques are grounded in a statistical approach that calculates word occurrences within the text. While BoW constitutes a collection of frequency vectors, the TF-IDF model encompasses the significance of words

[5]Recommendations on key lengths can be found at https://www.keylength.com/
[6]https://github.com/torproject/torspec/blob/main/tor-spec.txt

based on their relative importance. A term is considered more significant if it appears frequently in a target document and infrequently in other documents (Dang et al., 2020). Word frequency-based analysis is an essential component of Sentiment Analysis (SA), one of the most prevalent Natural Language Processing (NLP) applications. Although statistical computational approaches have demonstrated success in text classification (Falck et al., 2020; Luo, 2021; Singh et al., 2021), they have yet to be explored within the context of computer networks.

This study uses character statistics analysis to examine the characteristics of encrypted payloads based on individual hex characters. Similar to the key steps in textual analysis, such as data preprocessing to eliminate irrelevant characters and performing character statistics analysis to obtain insightful information, our approach involves data preprocessing to discard unnecessary data and extract the features from each hex character of raw payload data. Statistical techniques and ML are then employed to analyse these features and report insights on our approach.

2.5 Statistical Analysis

Statistical analysis is a process that involves the analysis, interpretation, and presentation of large data sets to extract useful quantitative information concerning the sample characteristics (Ott & Longnecker, 2015; Pagano, 2012). There are two types of statistical analysis: *descriptive statistics*, which are used to describe the characteristics of a set of observations, and *inferential statistics*, which are used to make inferences about a broader population using data from a sample. Our study employs both types of statistical analysis to support our research hypothesis.

Since there are various statistical techniques available for data analysis, it is essential to choose the relevant techniques for our study based on the levels of measurement (Fisher & Marshall, 2009). The levels of measurement classify the values assigned to variables in relation to each other and are broadly categorised into three groups. The *nominal level* is a variable measurement based on the names of the categories, such as sex (male = 1, female = 2). The *ordinal level* is a variable measurement of variables that cannot be measured directly, such as satisfaction level (1 = very unsatisfied, 5 =

very satisfied). The *continuous level* is a measurement of infinite scales, such as weight and temperature. Our study's data is derived from the calculation of hex characters within the ciphertext. As such, it falls under the continuous level of measurement. We will analyse this quantitative data using descriptive statistics metrics and generalise the results using inferential statistics, as further explained below.

2.5.1 Descriptive Statistics

Descriptive statistics employ various metrics to summarise the characteristics of a dataset collected from either a sample or an entire population. Descriptive statistics typically involve three primary measurements: distribution, which evaluates the frequency of sets of data values; central tendency (e.g., mean), which estimates the midpoint or centre of sets of data values; and variability (e.g., min, max and SD), which quantifies the spread or dispersion of data. The quantitative data in our study consists of continuous variables and will be measured using data frequency, mean, min, max, and SD. We will present all descriptive statistics numerically and visually in graphs to facilitate comparisons. This comparative summary will provide a preliminary characterisation of the two groups of data from two different networks.

2.5.2 Inferential statistics

Inferential statistics are used to draw conclusions about an entire population based on a sample data set. Hypothesis testing is a common application of inferential statistics (Allua & Thompson, 2009). Two types of inferential statistics are parametric and nonparametric tests, which are based on the frequency histogram graph plotting. A parametric inference test is appropriate when the data distribution is normal or has a symmetrical bell shape. Conversely, a nonparametric test is employed for data with a non-normal distribution. Several methods can be used to test for the normality of continuous data. The Kolmogorov–Smirnov test and the Shapiro–Wilk test are two well-known methods for testing normality. Statistical software such as SPSS[7] and PSPP[8]

[7] https://www.ibm.com/products/spss-statistics
[8] https://www.gnu.org/software/pspp/

offer built-in functions for easy data normality testing. The Shapiro–Wilk test is best suited for small sample sizes ($n < 50$), whereas the Kolmogorov–Smirnov test is more appropriate for larger sample sizes ($n > 50$) (Mishra et al., 2019).

Given that our observational data is assumed to deviate from a normal distribution and the data points are independent of one another, the appropriate statistical test to examine the null hypothesis is the 'Mann-Whitney U Test', also known as the Mann-Whitney-Wilcoxon (MWW) or Wilcoxon rank-sum test. This nonparametric test is employed to compare the attributes of two groups of independent samples. Statistical software packages such as SPSS and PSPP offer built-in functions to facilitate the execution of the Mann-Whitney U Test. The test yields a p-value ranging from 0 to 1. If the p-value is less than 0.05, the difference is considered statistically significant, providing strong evidence against the null hypothesis, as there is less than a 5% probability of it being accurate. Consequently, the null hypothesis is rejected, and the alternative hypothesis is accepted. In contrast, if the p-value exceeds 0.05, the interpretation would be the opposite, supporting the null hypothesis instead. Equation 2.2 describes the Mann-Whitney U statistics by the following, for each group (Nachar et al., 2008):

$$\begin{aligned} U_1 &= n_1 n_2 + \frac{n_1(n_1+1)}{2} - \sum R_1 \\ U_2 &= n_1 n_2 + \frac{n_2(n_2+1)}{2} - \sum R_2 \end{aligned} \tag{2.2}$$

where n_1 represents the size of the first sample, n_2 denotes the size of the second sample, $\sum R_1$ refers to the sum of the ranks of the first sample when the samples are arranged in order, and $\sum R_2$ corresponds to the sum of the ranks of the second sample when the samples are sorted in order.

Our first analysis, which is based on statistical tests, is the initial step in addressing the first research question: "Can we differentiate Tor from nonTor encrypted traffic based on their encrypted payload?" Chapters 3 and 4 will detail how this statistical analysis can aid in answering this question. In the following section, we will explore ML, which is related to the second analysis and is also an essential tool in answering the

second research question: "Can we efficiently distinguish Tor traffic using the proposed approach?" ML is a broad area comprising many disciplines that can assist scientists automatically in solving a wide range of real-world problems. Therefore, we can leverage its applications to help clarify any ambiguities. Chapters 3 and 5 will also discuss how ML can be involved in addressing this research question.

2.6 Machine Learning

ML approaches have become increasingly popular for automating computational tasks, simplifying large and complex problems, and enabling faster and more efficient completion of manual jobs. The appeal of ML lies in its ability to learn or be trained from meaningful data and produce outputs for particular tasks with fewer lines of code than manual programming would require. This section provides an overview of ML, its applications, and its relevance to the research questions addressed in this study.

2.6.1 Overview

Early Artificial Intelligence (AI) techniques were primarily based on rule-based systems, operating according to a predefined set of instructions. For example, email spam filters sort emails based on blacklisted terms. However, these systems have significant disadvantages when handling complex tasks, such as distinguishing between cat and dog images or adapting to evolving spam strategies. When a new problem arises or an existing one significantly changes, rule-based systems often require extensive manual updates, which are both time-consuming and inefficient. To address these limitations, ML, a subfield of AI, has emerged, offering a more flexible and efficient approach. ML systems learn from examples rather than manually coding every possible rule. For instance, an ML system can identify spam by studying examples of flagged emails, or differentiate between cats and dogs by learning from labelled images. The ability to automate the learning of repetitive tasks is one of the most important characteristics of ML. Two well-known ML definitions are provided below.

First, Arthur Samuel, a pioneer in the field of ML, described it in 1959 as *"Field of study that gives computers the ability to learn without being explicitly programmed."* (Bhavsar et al., 2017; Samuel, 1959).

Second, a more contemporary definition from Tom Mitchell, author of a prominent ML book, describes it in 1997: *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."* (Mitchell, 1997).

These definitions share the notion of allowing computer systems to perform tasks beyond traditional programming by intelligent self-learning from their surroundings. In other words, ML refers to the ability of computer systems to improve their performance on a specific task by learning from experience, without being explicitly programmed. This involves the use of algorithms that enable computers to identify patterns and make predictions based on the data they have been exposed to, and subsequently adapt to new information to improve their performance. ML integrates various scientific domains, including statistics, mathematics, and computer science, to achieve remarkable success in a wide range of different applications, such as speech recognition, image processing, NLP, autonomous vehicles, and many more. It has become a rapidly growing field of research and development that is not only transforming traditional computer-related areas but also extending its impact to various other disciplines, including biology, chemistry, physics, medicine, and agriculture, while also addressing complex problems in cryptography.

The application of ML in cryptography facilitates cryptanalysis of cryptosystems, such as cryptanalysis, password strength evaluation, side-channel attacks, encrypted traffic classification, etc. (Benamira et al., 2021; Gjorgjievska Perusheska et al., 2021). In cryptanalysis, ML can be used to identify the classic cipher algorithm used to generate a given ciphertext message. A study by Krishna (2019) utilised Support Vector Machine (SVM) and Hidden Markov Model (HMM) to classify ciphertext consisting only of alphabetic characters and assuming the plaintext is English. The authors showed that HMM was more effective and that experiments with different numbers of hidden states of

HMM could determine the unknown key length. The models could also be used to score modern ciphers. In password security, Melicher et al. (2016) proposed a novel method for predicting the guessability of passwords using a Neural Networks (NNs)-based model. The authors trained a Deep Neural Networks (DNNs) with a dataset of over 50 million passwords to predict which passwords are likely to be guessed by adversaries. The model was faster, leaner, and more accurate in predicting password guessability than other tools. ML techniques can also be applied to break cryptosystems by performing specific side-channel attacks (Hettwer et al., 2020; Maghrebi et al., 2016; Timon, 2019). A study by Timon (2019) presented a new method called Differential Deep Learning Analysis (DDLA) for performing side-channel attacks using Deep Learning (DL) techniques in a Non-Profiled context. By combining key guesses with DL metrics, the authors showed that it is possible to recover information about a secret key. The method can outperform classic Non-Profiled attacks and can also break masked implementations in black-box without pre-processing or assumptions.

Additionally, the use of ML is particularly important in the field of classifying encrypted traffic, which is the main focus of this study. In recent years, the encryption of Internet traffic has become increasingly prevalent, which has made it more difficult for network administrators and security researchers to monitor and analyse network traffic. In order to address this problem, researchers have turned to ML techniques to develop models that can accurately classify encrypted traffic based on its characteristics. This approach has been the focus of numerous studies, as demonstrated by several surveys (Cao et al., 2014; H. Liu & Lang, 2019; R. Liu & Yu, 2020; Velan et al., 2015). These studies share the goal of using ML techniques to efficiently identify and classify targeted encrypted traffic using designed features. For example, Draper-Gil et al. (2016) introduced a classification technique for distinguishing encrypted and VPN traffic based on time-related features. Their study utilised two prevalent ML algorithms, C4.5 and kNN, which yielded similar results with over 80% accuracy, although C4.5 performed slightly better. Another study by Lotfollahi et al. (2020) proposed a Deep Packet scheme capable of identifying encrypted traffic and differentiating between VPN and nonVPN network traffic. They asserted that their DL-based approach, relying solely on packet

length features, surpasses all other proposed classification methods on the VPN-nonVPN dataset belonging to UNB ISCX[9]. These studies demonstrate the potential of ML in encrypted traffic classification, which is becoming increasingly important in the era of pervasive encryption.

As outlined in Section 2.1, the two fundamental characteristics of encryption schemes, *practical secrecy*[10] and *practical indistinguishable* encourage researchers in encrypted traffic detection to focus on flow-based. However, there are limitations to the flow-based approach in encrypted traffic detection due to several issues, which will be discussed in detail in Section 2.6.4. These limitations motivate us to explore a novel approach based on character statistics analysis of encrypted traffic. In the upcoming section, we will discuss why deep learning, despite its popularity and effectiveness in various domains, may not be the optimal choice for our study. Instead, we will argue that ML techniques offer a more suitable and targeted approach for addressing the specific challenges and objectives of our research.

Deep Learning

Prior to the emergence of advanced ML schemes, traditional ML was widely employed. In recent years, DL, a subset of ML, has demonstrated remarkable promise in intelligently solving complex problems through a specific type of Artificial Neural Networks (ANNs) known as DNNs. The DNNs architectures greatly enhance the capabilities of shallow learning, which is commonly referred to as conventional ML (Janiesch et al., 2021; H. Liu & Lang, 2019). This advanced approach is widely known as DL. With the increasing availability of big data, DL algorithms have gained significant popularity due to their ability to leverage large volumes of data for improved performance (Q. Zhang et al., 2018). Based on the findings of Aggarwal et al. (2018) and Mathew et al. (2020), it is typically observed that DL models tend to exhibit improved performance with more data, while ML models often reach a plateau after a certain point. It is worth noting that both DL and ML can utilise handcrafted features (Liang et al., 2017; Shaheen et al., 2016; Shinde & Shah, 2018). However, DL has the distinct advantage of automatically

[9]https://www.unb.ca/cic/datasets/index.html

[10]While the concept of perfect secrecy represents an ideal goal in cryptography, achieving it in practice is not always feasible due to various constraints in the context of computer networks. Consequently, the term *practical secrecy* and *practical indistinguishable* are more appropriate for real-world applications.

learning the significance of extracted features from data, in contrast to ML, which relies solely on the provided handcrafted features.

DL has the potential to achieve high predictive accuracy using ANNs, or NNs in short, which consists of computational units called *neurons* that mimic the biological neural network of the human brain to intelligently solve problems (Aggarwal et al., 2018). These computational units are interconnected by weights and arranged in multiple processing layers, such as input, hidden, and output layers, for model learning (Sarker, 2021). Neural networks with more than one hidden layer are called DNNs, and the models created with DNNs are referred to as DL (Weidman, 2019). The most common DL algorithms include Multilayer perceptrons (MLPs), Convolutional Neural Networks (CNNs) or ConvNet, and Long Short-Term Memory Recurrent Neural Networks (LSTM-RNNs) (Sarker, 2021).

Despite the remarkable achievements of DL, it is not suitable for all scenarios. Inappropriate implementation of DL can lead to challenges due to its limitations, such as being resource-intensive and time-consuming (Mueller & Massaron, 2019; Müller & Guido, 2018). The arguments against using DL in this study are as follows: 1) DL's non-feature engineering nature makes it a black box model, rendering the results difficult to interpret (H. Liu & Lang, 2019; Sheu, 2020). In some cases, a white box model is required for interpretability (Loyola-Gonzalez, 2019). Our approach, based on character statistics analysis of Tor and nonTor packets, necessitates this. 2) DL excels at analysing unstructured data, such as images or videos, which is beneficial for applications like self-driving cars, image/speech recognition, and NLP (Mathew et al., 2020). Our data is organised in tabular form, referred to as structured data, which is better suited for traditional ML algorithms (H. Liu & Lang, 2019). 3) It is true that handcrafted features can also work effectively with DL, but doing so would necessitate high-end computing resources and increased processing time, making it unsuitable for real-time traffic analysis compared to shallow models that can perform faster classification (H. Liu & Lang, 2019). For these reasons, DL is not the appropriate method for Tor traffic classification in this study. With this understanding, we now turn our attention to the more general aspects of ML. In the following sections, we will discuss various ML techniques, algorithms, and

their applications, particularly focusing on those relevant to the task of classifying Tor traffic.

2.6.2 Machine Learning Types

The primary objective of the ML approach is to enable computers to learn autonomously using ML algorithms or models. Based on the characteristics of the input data used for model training, ML is classified into four types: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning (Sarker, 2021).

1) Supervised Learning: Supervised learning consists of two phases: learning and prediction. In the learning phase, a labelled dataset containing input and desired output is employed to train the ML algorithm, assuming a pattern of relationship exists between them. The algorithm will make predictions once the final model is developed. The two major types of supervised learning are *regression*, where the target variable is a continuous value, and *classification*, where the target variable is a discrete value (Sarker, 2021). Sentiment analysis in text classification, as described in Section 2.4, is an example of supervised learning. In this research, Tor traffic detection is considered a classification problem since it involves separating encrypted payloads based on whether they belong to the Tor or nonTor network. Popular supervised ML algorithms include kNN, Linear models, Naive Bayes classifiers, DTs, RFs, SVM, and MLPs, which is a fully connected multi-layer NNs (Müller & Guido, 2018).

2) Unsupervised Learning: Unsupervised learning is an ML technique applied to unlabelled data, as opposed to supervised learning. Model training and correct answers are not available for unsupervised learning. Instead, unsupervised learning algorithms are used to discover correlations and relationships by analysing the information provided in the dataset. Anomaly detection in NTC is a common application of unsupervised learning. It involves identifying unusual or abnormal patterns in data, often without prior knowledge of what those patterns may look like, making it suitable for analysing unlabelled data (Eskin et al., 2002; Syarif et al., 2012). In such cases, unsupervised learning algorithms can reveal the underlying structure of the data and detect deviations from the norm, effectively identifying anomalies even in the absence of

labelled information. Examples of unsupervised learning algorithms include K-Means, Hierarchical clustering, and Principal Component Analysis (PCA).

3) Semi-supervised Learning: Semi-supervised learning is a combination of supervised and unsupervised learning in which the model is trained using both labelled and unlabelled data. Semi-supervised learning approaches are employed in real-world situations when labelled data is inadequate. The classification accuracy can be improved if the unlabelled data provides additional useful information for prediction. Examples of this type of learning algorithm include Tri-training, used for classifying Tor anonymous traffic at the application level (Lingyu et al., 2017), and self-trained algorithms used for NLP (Tanha et al., 2017; B. Wang et al., 2008).

4) Reinforcement Learning: This type of learning differs from the previous three types as no training datasets are required. Reinforcement learning enables an agent to learn through trial and error in an interactive environment, using feedback based on previous experiences. Examples of reinforcement learning problems include playing computer games, complex decision-making problems, and reward systems (Sutton & Barto, 2018).

2.6.3 Supervised Learning Process

Supervised learning is a technique used to solve classification problems, aiming to enable machines to learn from predefined classes. The general processes of the supervised learning model are depicted in Figure 2.10.

The supervised learning process has two phases: learning and prediction. In the learning phase, the first step is data preprocessing, which is crucial for successful ML as the classifier must identify useful patterns from the input data. This step involves eliminating irrelevant data, such as duplicates and outliers, followed by feature engineering. Feature engineering involves transforming raw input data into a set of features that can be used to distinguish different classes of data. In some cases, selecting the most important features can also be done to reduce computational complexity and prevent overfitting. Once the features have been extracted, the model is trained using supervised learning algorithms to generate a classification model. This involves selecting

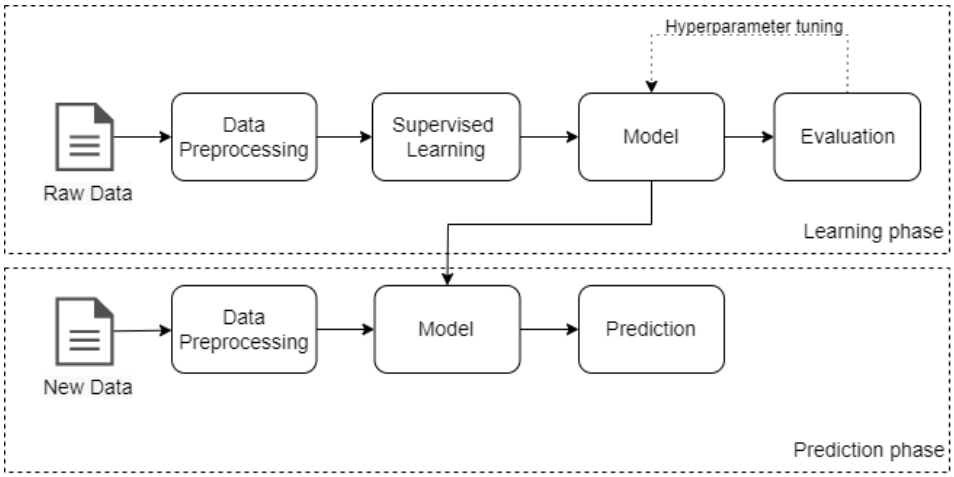


Figure 2.10: Diagram of the supervised learning process (adapted from B. Liu, 2011)

an appropriate model and optimising its parameters to achieve the best performance on the training data. Hyperparameter tuning may be required to enhance the model's performance, and this tuning process may be repeated until the best final model is obtained. The final step in the learning phase is to evaluate the model's testing results. This involves testing the trained model on a set of data that was not used during the training process to measure the model's performance and assess its generalisation ability on the test set. In the prediction phase, the finalised model can be used in real-world tasks to predict classes of new, unseen data. The performance of the model can be assessed by comparing its predictions to the true labels of the test data. If the model's performance is not satisfactory, it may need to be retrained or updated with new data to improve its accuracy on future predictions.

2.6.4 Feature Engineering

A feature, also referred to as an attribute, input, or variable, is a distinctive and quantitative characteristic of an observed activity or object. Features serve as essential input variables for ML algorithms to perform classification tasks. Feature engineering is the process of transforming raw data into a well-structured set of features that can enhance the performance and training of ML models (Dong & Liu, 2018). This section focuses on two critical steps in feature engineering pertinent to this study: feature

extraction and feature selection.

Feature Extraction: Feature extraction is a crucial step in feature engineering, as it involves identifying and extracting relevant features from raw data. This process aims to capture significant information, reduce data dimensionality, and retain essential properties. Depending on the nature of the data and the specific problem being addressed, various techniques can be employed for feature extraction, including Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and statistical or mathematical transformations. However, for domain-specific problems, such as the classification of Tor and nonTor traffic in our approach, manual or domain-specific feature extraction is often more suitable. This approach relies on domain knowledge and expert insights to derive meaningful features that are relevant to the specific problem and allow for more accurate classification. This is because manual feature extraction allows for the inclusion of relevant information that may not be captured by automatic feature extraction methods.

In the context of Tor traffic classification using ML approaches, previous studies have primarily focused on time-based features derived from traffic flow (Cuzzocrea et al., 2017; Lashkari et al., 2017; Shahbar & Zincir-Heywood, 2015). This emphasis is due to the fact that Tor packet delivery requires passing through multiple proxy servers worldwide, resulting in slower traffic compared to regular traffic. However, the asymmetric nature of Internet routing, in which an ISP's router may only capture one direction of a flow, can influence time-related features (Crotti et al., 2009; Salman et al., 2020). Network conditions such as latency, jitter, and congestion can cause inconsistencies in packet timings, which can also reduce the effectiveness of these features. Furthermore, although packet size is a commonly used feature due to Tor's fixed cell-size, packet lengths can be altered when traffic passes through a tunnel or proxy (Rezaei & Liu, 2019). In addition to network-related issues, flow-based analysis has its own set of disadvantages, particularly when calculating features from multiple packets. These disadvantages include the strain on computational resources, potential inaccuracies due to inconsistent packet characteristics or network anomalies, privacy concerns, and significant network overhead when analysing large volumes of traffic. These limitations highlight the need for alternative approaches that are more efficient, accurate, and privacy-preserving in

the context of Tor traffic classification.

Consequently, we propose a novel approach for feature extraction that emphasises character analysis of encrypted payloads. This concept arises from the fact that Tor, like other nonTor encryption protocols, uses TLS for encryption, but its routing and encapsulation methods are distinct, as presented in Section 2.3.3. This distinction could potentially enable a novel approach to classify Tor traffic. Our method analyses occurrences of hex characters (0-9 and A-F) in ciphertexts, calculating their relative frequencies or proportions. These character statistics serve as features for the ML model, representing the ciphertext's composition based on hex character distribution. The goal is to capture subtle differences in hex character distribution between encrypted traffic types on Tor and nonTor networks for effective classification. Our proposed features, derived from individual packets, exploit a single packet in classifying network traffic, offering a significant advantage over multi-packet flow-based features and providing a more efficient method.

It is important to note that character analysis of encrypted payloads can be challenging due to the nature of encrypted data. Encryption algorithms, including TLS, are designed to produce ciphertext that appears random, thus making it difficult to distinguish between different types of encrypted traffic by analysing the character patterns. Nevertheless, as discussed in Section 2.1, the cryptographic principles, which offer rigorous trust in security within a controlled and idealised environment, may not necessarily guarantee security in the real world (Katz & Lindell, 2020), particularly in the context of computer networks. Various factors during data transmission in a heterogeneous computer network environment, such as data fragmentation and the use of different encryption mechanisms, can impact their encrypted data, potentially exposing meaningful patterns that could be exploited.

Feature Selection: Following feature extraction, feature selection is the subsequent step in feature engineering. The primary objective of feature selection is to identify the most informative and relevant features from the extracted set, thereby reducing the number of features used in the ML model. This step is crucial in mitigating overfitting by reducing the complexity of the model and removing irrelevant or redundant

features. Overfitting occurs when a model learns the noise or random fluctuations in the training data instead of the underlying patterns, leading to poor generalisation performance on new, unseen data. Feature selection not only helps in improving the model's generalisation performance but also enhances its interpretability and reduces the training time. Feature selection techniques can be broadly classified into three categories: filter methods, wrapper methods, and embedded methods. Filter methods evaluate features independently based on their relevance to the target variable, wrapper methods assess feature subsets by evaluating the model's performance, and embedded methods incorporate feature selection as part of the learning algorithm itself.

2.6.5 Classification Algorithms

Selecting appropriate algorithms for specific tasks is a challenging effort. For our binary classification task, we considered simple yet effective supervised algorithms that have demonstrated success in prior literature. One such algorithm is the DTs, which is widely used in various disciplines such as text analysis (Saad & Yang, 2019; Singh et al., 2021) and NTC (Song & Ying, 2015). DTs are known for their simplicity, lack of ambiguity, and resilience even in the face of missing values. However, one major disadvantage of DTs is the risk of overfitting, especially when dealing with small datasets. To mitigate this issue, the RFs algorithm can be used to enhance model performance and prevent overfitting (Shaik & Srinivasan, 2019; Ziegler & König, 2014). According to Aggarwal (2018), RFs are among the best-performing classifiers. In addition to DT-based and RFs algorithms, we chose kNN as a competitive algorithm. This is due to kNN simplicity, effectiveness, ease of understanding, and high performance, which make it a popular choice in many domains, including text mining to automatically classify text documents into categories (Bijalwan et al., 2014; Menzies et al., 2018; Moldagulova & Sulaiman, 2017; Radovanović & Ivanović, 2008) and NTC (Deng et al., 2016; Moldagulova & Sulaiman, 2017). The success of the kNN classifier in these applications inspired us to adopt it for our classification task alongside the other chosen algorithms.

Decision Trees

DTs are a popular predictive algorithm in supervised learning for solving regression and classification problems. They are non-parametric and effective at dealing with huge, complex datasets (Song & Ying, 2015). DTs builds a model by predicting the value of a target variable using basic decision rules derived from data attributes. The DTs algorithm employs a repeated divide-and-conquer approach to divide a node into two or more sub-nodes. This approach creates an inverted tree-like network in which the root node is the topmost attribute, branches are decision rules, child nodes are attributes, and leaf nodes are decision outcomes or classes. A classification tree is built in a top-down approach via the recursive splitting of nodes into sub-nodes. The DTs algorithm utilises a recursive divide-and-conquer strategy to partition a node into two or more sub-nodes. This method constructs an inverted tree-like structure with the root node representing the topmost attribute, branches representing decision rules, child nodes corresponding to attributes, and leaf nodes signifying decision outcomes or classes. A classification tree is built using a top-down approach, where nodes are recursively split into sub-nodes.

Several prevalent splitting criteria are employed to determine which feature to split in DTs algorithms, including information gain, Gini index, and gain ratio. Information gain is grounded in Shannon Claude's information theory and entropy concept (Shannon, 1948). Although information gain serves as an effective splitting criterion, it can exhibit bias when dealing with a large number of features. To mitigate this bias, the gain ratio, which normalises information gain, is utilised as an alternative (Breiman et al., 2017). The Gini index evaluates the impurity of a node by assessing the disparity between the probability distributions of the target feature values, thereby contributing to the decision-making process in splitting nodes.

Figure 2.11 depicts a simple decision tree that determines whether Saturday mornings are favourable for playing tennis based on three factors: outlook, humidity, and wind conditions.

In ML, some prominent DTs-based algorithms include Iterative Dichotomiser 3 (ID3), C4.5, and CART. The ID3 algorithm, introduced by Quinlan (1986), is considered a relatively simple approach. However, it possesses several drawbacks, such as not guar-

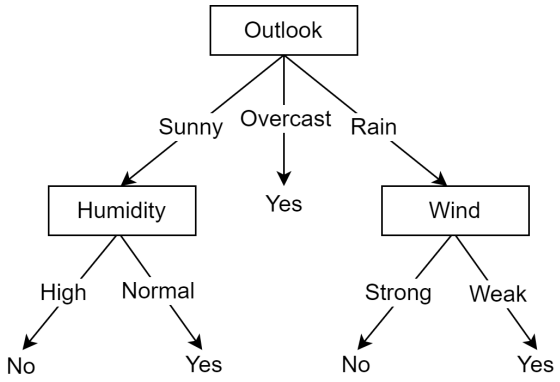


Figure 2.11: A decision tree (adapted from Mitchell, 1997)

anteeing an optimal solution, encountering data overfitting issues, and being designed exclusively for nominal attributes (Rokach & Maimon, 2015). To address ID3's limitations, Quinlan (2014) introduced C4.5, an extended version of ID3. C4.5 incorporates pruning to manage large trees and handles numeric attributes and missing values. C5.0, an enhanced and commercial version of C4.5, consumes less memory and computational time (Rokach & Maimon, 2015). Classification and Regression Trees (CART), developed by Breiman et al. (2017), can generate both regression and classification trees. and CART adopts the Gini index. Additionally, while ID3 and C4.5 support top-down tree construction with multiway splits, CART employs binary decision tree splits (Shamrat et al., 2021). Due to its limitations, ID3 may still be used for educational purposes but is less suitable for use in modern research or applications. It is often replaced by more advanced algorithms like C4.5 and CART, which address the limitations of ID3 and provide improved performance in various scenarios. J48, the Java implementation of the C4.5 algorithm, is employed in WEKA, while the CART algorithm is available in the Scikit-learn DTs library for Python 3. Given that the use of a single decision tree can result in overfitting and poor generalisation performance, the combination of multiple DTs to form a RFs can help address these issues and improve overall model performance (Müller & Guido, 2018).

Random Forests

RFs (Breiman, 2001) is an ensemble method that combines multiple DTs to create a more robust and accurate model. Ensemble methods leverage the strengths of two or more base models to yield a powerful, combined model. RFs models, which are ensembles of DTs, are considered among the most effective ML models for both classification and regression tasks (Müller & Guido, 2018).

As suggested by its name, a random forest is an ensemble of individual decision trees. The RFs method improves upon the performance of a single DTs by constructing multiple trees, each trained on a different random subset of the data. This is achieved through a process called *bootstrapping*. In addition, a random selection of features is used to form each decision tree model, introducing diversity among the trees. This combination of bootstrapping and random feature selection helps mitigate overfitting, which is a common issue with single decision trees. The process of combining results from multiple models is known as aggregation. In the context of RFs, the combination of bootstrapping and aggregation is referred to as *bagging* (short for *bootstrap aggregating*). Bagging helps create diverse decision trees and reduce overfitting, thus enhancing the performance of the RF model. For each input instance, the RF model makes predictions using all the individual trees. In a classification problem, the final prediction is based on the majority vote of all trees. In a regression problem, the prediction is based on the average of all decision tree predictions. By aggregating the predictions of multiple trees, the random forest model can achieve better generalisation and accuracy compared to a single decision tree. This makes the RFs model an outstanding choice for many ML tasks. Figure 2.12 provides a graphical illustration of a RFs model.

The use of bootstrap aggregation in a RFs offers a substantial advantage over a single decision tree, as it helps reduce variance and mitigate overfitting. Moreover, RFs is capable of handling a large number of input attributes while maintaining efficient processing (Rokach & Maimon, 2015). However, one drawback of the RFs approach is its reduced interpretability compared to a single decision tree. Due to the complex nature of an ensemble of trees in a forest, it becomes challenging to study individual trees, rendering RFs a black box model (Guidotti et al., 2018). On the other hand, the

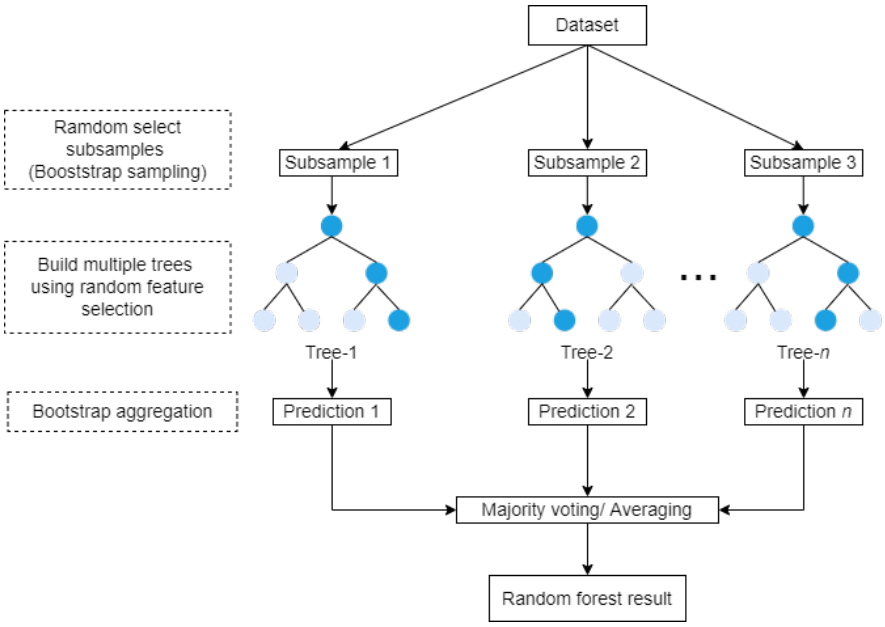


Figure 2.12: The principle of the RFs algorithm prediction (adapted from Y. Liu et al., 2012)

kNN algorithm is notable for its transparency, which means it is easy to understand. Along with its simplicity and effectiveness, kNN has been successfully applied in various research areas. Therefore, it presents a promising alternative to the RFs and DTs models in our study.

k-Nearest Neighbors

The kNN algorithm is a widely used non-parametric learning algorithm in ML and data mining applications. Despite its reputation as one of the simplest ML algorithms, it has proven to be effective in many various cases, including image recognition, text classification, recommendation systems, and others (Guo et al., 2003). It can solve both classification and regression problems and is based on the assumption that related things tend to cluster together (Allahyari et al., 2017). As the name implies, the kNN algorithm identifies the k closest neighbours to a given data point or object, and then classifies the object based on the majority label of those neighbours. In cases where multiple neighbours share the same distance to the object being classified, they may have different labels, thereby an odd value of k is typically chosen to ensure a unanimous

decision. To calculate the distance between two points or nearest neighbours, the kNN classifier commonly employs the Euclidean distance function. This function measures the straight-line distance between two points in a Euclidean space. It is important to note that the choice of distance metric can have a significant impact on the performance of the kNN algorithm, and different metrics may be more appropriate for different types of data. Therefore, it is essential to consider various factors such as data type, data distribution, and the nature of the problem when selecting an appropriate distance metric.

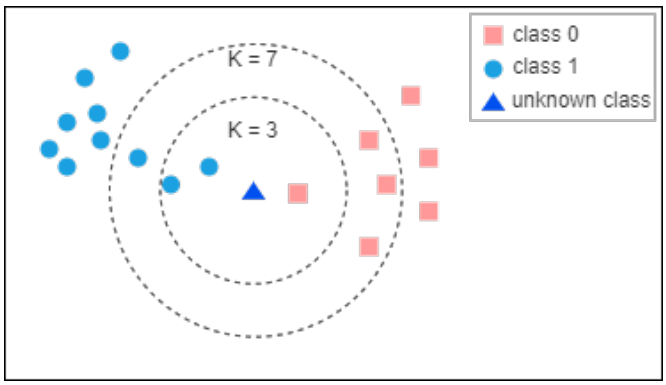


Figure 2.13: kNN method for binary classification with $k = 3$ and $k = 7$

Figure 2.13 illustrates a simplified explanation of the binary classification problem using the kNN method. The objects in the figure, represented as dots and squares, can be considered as labelled objects, while the new triangle represents an unlabeled object x. To predict the class of x, the distances between x and the labelled samples in the training dataset are calculated. The k samples with the smallest distances to x are selected, and their class labels are used to vote for the class of x. For example, if $k = 3$, x belongs to the dot class, whereas if $k = 7$, x is assigned to the square class. It is important to note that the predicted class may vary depending on the chosen value of k.

Choosing the optimal value of k is still a challenging task in the kNN method. To address this issue, several studies have proposed improved versions of the kNN method that aims to find the best k value for a more efficient model (Guo et al., 2003; S. Zhang et al., 2017, 2018). Although there are many methods for selecting the optimal k value, one common approach is to test the model's performance using different k values and

select the one that yields the best results (Guo et al., 2003).

2.6.6 Model Evaluation

Model evaluation is a critical step in assessing the effectiveness of a trained and tested model. The process involves dividing the dataset into appropriate portions for model building and testing. Two common methods for splitting data for model evaluation are discussed below.

1) Percentage split: This method is typically used for large datasets. The dataset is split into two portions for training and testing based on a percentage split. The choice of the split ratio affects the performance of the model. In practice, a commonly used ratio is 66% for training and 34% for testing. However, the exact ratio can vary depending on the specific problem and dataset being used. After the split, the model is trained using the training subset and evaluated using the test subset. This technique is simple and straightforward, but it suffers from the drawback of not testing all available data, which can result in the model experiencing overfitting, i.e., low bias and high variance. To mitigate this overfitting issue, multiple models can be fitted using cross-validation (Robertazzi & Shi, 2020).

2) Cross-validation: Cross-validation is a widely used resampling technique for training and testing predictive models. It evaluates the model on an independent test set, similar to the percentage split method. However, cross-validation assesses all split test subsets. To perform cross-validation, the number of folds (N) must be determined. Commonly, N is set to 5 or 10 (Kuhn, Johnson, et al., 2013). A larger N results in reduced bias but increased computational time. Notably, N is often set to 10, as numerous comprehensive tests have demonstrated this to be the optimal parameter. The cross-validation procedure then proceeds by dividing the dataset into N equally sized, random folds (referred to as stratified cross-validation). For each of the N folds, N-1 subsets are used for training, while the remaining subset is employed for testing the trained model in rotation over N iterations. The objective of stratified cross-validation is to minimise potential variance and bias by thoroughly training and testing all available data.

In conclusion, for small or simple datasets, the percentage split may provide sufficiently reliable performance estimates, and the added complexity of cross-validation may not be necessary. Conversely, for large or complex datasets, cross-validation can provide more robust performance estimates and help prevent overfitting. However, the choice of the evaluation method in practice depends on the specific problem, dataset size and structure, computational resources, and other factors. There is no strict rule or threshold that applies to all cases. As a result of the time-consuming issue and limited computational resources in this study, cross-validation has been utilised for smaller datasets containing around 10,000 instances or fewer. On the other hand, for larger datasets exceeding approximately 10,000 instances, a percentage split is more appropriate.

2.6.7 Performance Metrics

In the domain of ML, evaluating the performance of a model is crucial for understanding its effectiveness and accuracy in addressing specific tasks. Selecting appropriate evaluation metrics allows researchers and practitioners to compare different models and identify areas for improvement. Various metrics are available to assess the performance of models in response to specific problems. The choice of evaluation metric primarily depends on the type of model used, such as regression or classification models. For classification problems, which are the focus of this study, the following measures are commonly employed to report model performance:

Confusion Matrix: A confusion matrix is an $N \times N$ table that summarises the performance measurement of a model on a test dataset, where N is the number of classes in a classification problem. The matrix consists of the number of predicted classes and the number of actual classes, providing insight into the errors and types of errors generated by the classifier. In the case of a binary classification problem, as considered in this study, a 2×2 confusion matrix is illustrated in Figure 2.14.

The components of the confusion matrix are as follows:

- True Positive (TP): The sample value is true and the model prediction is positive. In this case, the instance is classified as Tor and the prediction is also Tor (correct

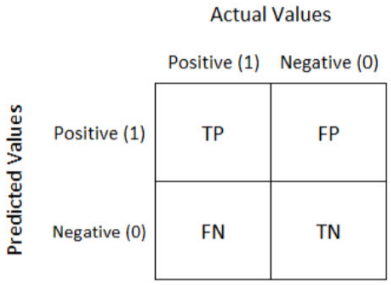


Figure 2.14: 2×2 Confusion Matrix

prediction).

- True Negative (TN): The sample value is true and the model prediction is negative. In this case, the instance is classified as nonTor and the prediction is also nonTor (correct prediction).

- False Positive (FP): The sample value is false and the model prediction is positive. In this case, the instance is classified as Tor but the prediction is nonTor (incorrect prediction).

- False Negative (FP): The sample value is false and the model prediction is negative. In this case, the instance is classified as nonTor but the prediction is Tor (incorrect prediction).

Additional performance metrics can be calculated using the values obtained from the confusion matrix table, as described in the following equations:

Accuracy: Accuracy is a measure of a model's overall efficiency, calculated as the ratio of correct predictions to the total number of predictions. It is a useful starting metric; however, it may not be appropriate for significantly imbalanced classes (Chawla, 2009). The accuracy score can be calculated using Equation 2.3. To express the result as a percentage, multiply it by 100. A higher accuracy, closer to 100%, indicates a better model. In our case, accuracy measures how well the model can correctly classify Tor and nonTor traffic.

$$ACC = \frac{TP + TN}{(TP + TN + FP + FN)} \tag{2.3}$$

Precision: Precision measures the accuracy of positive predictions. The precision evaluates the proportion of true positive predictions (correctly identified Tor traffic) relative to the sum of true positives and false positives (instances incorrectly identified as Tor traffic). In other words, in our case, precision measures the model's ability to correctly classify Tor traffic out of all Tor traffic predictions. The precision score can be calculated using Equation 2.4. Precision values range from 0 to 1, with values closer to 1 indicating a better model.

$$PR = \frac{TP}{(TP + FP)} \tag{2.4}$$

Recall: Recall, also known as True Positive Rate (TPR) or sensitivity, measures the accuracy of correctly identifying relevant data. The recall is the proportion of true positive predictions (correctly identified Tor traffic) relative to the sum of true positives and false negatives (instances that are actually Tor traffic but are incorrectly identified as nonTor traffic). In other words, recall measures the model's ability to correctly classify Tor traffic out of all actual Tor traffic. The recall score can be calculated using Equation 2.5. Recall values range from 0 to 1, with values closer to 1 indicating a better model.

$$REC = \frac{TP}{(TP + FN)} \tag{2.5}$$

F1 score: The F1 score, also known as the F-score or F-measure, is the harmonic mean of precision and recall, providing a single metric that balances both aspects of the model's performance. In the context of classifying Tor and nonTor traffic, the F1 score can be interpreted as a measure of the model's ability to accurately identify Tor traffic while minimising both false positives (incorrectly identified Tor traffic) and false negatives (actual Tor traffic misclassified as nonTor traffic). The F1 score is calculated using Equation 2.6. Both TP and FP are considered, making them suitable metrics for unbalanced datasets. A high F1 score indicates that the model is effective with a good balance between precision and recall.

$$F1 = \frac{2 * (REC * PR)}{(REC + PR)} \tag{2.6}$$

ROC/AUC score: The Receiver Operating Characteristic (ROC) and Area Under The ROC Curve (AUC) are commonly used performance metrics for binary classification problems. The ROC curve is a graphical representation of the relationship between the TPR, also known as recall, and the False Positive Rate (FPR), where FPR = $\frac{FP}{(FP+TN)}$. The TPR is plotted on the y-axis, and the FPR is plotted on the x-axis, showing the trade-off between sensitivity and specificity at various classification thresholds. The AUC is a scalar value representing the area under the ROC curve, which measures the classifier's ability to distinguish between positive and negative samples. The AUC value ranges from 0 to 1, where higher AUC values indicate better performance. An AUC value of 0.5 suggests that the classifier is no better than a random chance at distinguishing between positive and negative samples, while an AUC of 1.0 indicates perfect classification. Although it has been suggested that the ROC/AUC metric is particularly suitable for imbalanced datasets, as it evaluates a classifier's performance without making assumptions about class distribution (Gómez et al., 2019; H. Zhang et al., 2012), the ROC/AUC metric can still prove valuable for evaluating classifiers on balanced datasets. This is because it provides additional insights into the classifier's performance across various thresholds. While accuracy offers a single value representing the overall correct prediction rate, it may not fully capture a classifier's performance concerning FP and FP. In contrast, the ROC/AUC metric accounts for the trade-off between sensitivity and specificity at different thresholds, thus facilitating a more comprehensive understanding of the classifier's performance. Consequently, this study incorporates ROC/AUC scores in the evaluation of the models.

A comprehensive understanding of the background knowledge associated with this research has been provided through the review of the fundamental technologies underpinning the research methodology. This review should provide readers with a clear understanding of the topic. Nevertheless, it is also crucial to examine related research or previous studies focused on NTC, particularly in Tor traffic, ranging from the past to the present. This section will discuss the methods, benefits, and drawbacks of these prior works, which have inspired the development of our novel approach.

2.7 Related Work on Network Traffic Classification

NTC encompasses a variety of techniques aimed at classifying unknown network packets traversing a computer network into known traffic classes, such as browsing, email, VoIP, etc. NTC serves several purposes for ISP and Network Administrators, including QoS management (e.g., critical applications like video conferencing receive priority over non-critical ones like email, resulting in a better user experience), network security (e.g., malware and intrusion detection), and more. NTC can generally be divided into three main methods: Port-based, Payload-based, and ML-based.

2.7.1 Port-based Method

The most basic technique for NTC relies on inspecting the 'well-known' ports indicated in packet headers, such as ports 20 and 21 for FTP, port 22 for SSH, port 25 for SMTP, and port 80 for HTTP (Reynolds & Postel, 1992). Traffic can be classified based on the port number and the presumed associated protocol. This method boasts the advantages of simplicity, speed, and effectiveness. However, since the use of assigned ports is not mandatory, many contemporary applications employ random ports to obfuscate TCP/UDP headers and evade detection. Consequently, the port-based classification approach may no longer be effective, as demonstrated by studies such as Karagiannis et al. (2005). Several studies (Karagiannis, Broido, Brownlee, et al., 2004; Karagiannis, Broido, Faloutsos, & Claffy, 2004; Karagiannis et al., 2003; A. W. Moore & Papagiannaki, 2005) have substantiated that using port numbers to identify application traffic has become increasingly unsuccessful.

Madhukar and Williamson (2006) conducted a comparison of three techniques for classifying Peer-to-Peer (P2P) applications: port-based classification, Application Layer signatures, and Transport Layer analysis. Due to P2P applications employing various obfuscation techniques, robust and effective methods for identifying P2P traffic are essential. The performance of each technique was analysed using sample traffic traces collected from the University of Calgary's Internet connection over a two-year period. The results demonstrated that port-based analysis is inefficient, as it can only identify between

30% to 70% of current Internet traffic. Although application signatures are reliable, they may be unattainable due to legal or technical constraints. Transport Layer analysis, a flow-based method focusing on connection behaviour patterns, appears promising as it offers an efficient means of measuring P2P traffic. To overcome the limitations of the port-based method, other classification techniques that utilise payload-based approaches are preferred.

2.7.2 Payload-based Method

The payload-based method, also known as DPI, relies on examining characteristic signatures or patterns of strings found in the contents of application packets or payloads. Although this technique often yields highly accurate results for classifying network traffic, it also encounters several drawbacks, such as high computational costs for pattern matching in payloads, potential confidentiality violations, and ineffectiveness when inspecting encrypted packets (Bakhshi & Ghita, 2016; Shafiq et al., 2016). Encrypted traffic has long posed a significant challenge for DPI. To address this, Sherry et al. (2015) introduced *BlindBox*, a DPI technique capable of inspecting encrypted payloads without decryption, thereby alleviating privacy concerns. Their approach represents the first instance of combining encryption benefits with DPI middlebox functionality. The term 'BlindBox' refers to the middlebox's inability to access the content of the traffic. However, their proposed middlebox is only applicable to HTTPS traffic.

2.7.3 Machine Learning-based Method

The increasing prevalence of encrypted traffic has led to a growing interest in ML and statistical-based classification methods for NTC. Numerous studies have demonstrated the effectiveness of using ML techniques for classifying network traffic, including tasks such as malware detection (Tran et al., 2020; Yeo et al., 2018), network intrusion detection (Gogoi et al., 2012; Pektaş & Acarman, 2019), (W. Wang et al., 2020), protocol-based classification (Bhargava et al., 2013; Schatzmann et al., 2010; Vinayakumar et al., 2017).

As discussed in Section 2.2.1, ML-based NTC typically utilises flow-based features derived from the Transport Layer (Layer 3 in the TCP/IP model) because they provide

valuable insights into the behaviour and characteristics of network traffic without requiring access to the content of the packets. Flow-based features can include metadata such as connection duration, packet size, packet inter-arrival time, and the number of exchanged packets between sender and receiver. This approach aids in classifying traffic according to Application Layer protocols by analysing patterns and relationships in the Transport Layer data. Additionally, the Internet Layer (Layer 2 in the TCP/IP model) plays a supplementary role in protocol identification. The IP header, part of the Internet Layer, contains a 'Protocol' field that explicitly specifies the next encapsulated protocol, such as TCP, UDP, or ICMP. This information allows for easier identification of higher-layer protocols. These ML-based NTC methods, which leverage flow-based features from the Transport Layer and supportive information from the Internet Layer, have been successfully applied to various classification tasks, including Tor traffic classification, the focus of this study, which will be discussed in detail in the following sections.

2.7.4 Network Traffic Classification on Tor

As mentioned earlier, NTC is a crucial task for detecting Tor traffic because Tor traffic is often disguised to look like other types of network traffic, making it difficult to identify without sophisticated analysis tools. Researchers have made efforts to study and develop approaches for detecting network traffic generated from the Tor network, which offers several benefits, including: (i) providing insight into Tor traffic fingerprints to test the system's robustness (Shahbar & Zincir-Heywood, 2015), which could help minimise the detectability of Tor traffic and subsequently enhance user privacy (Barker et al., 2011; Granerud, 2010); (ii) facilitating cyber surveillance for authorities, as Tor's anonymity attracts criminals for illicit activities (Cuzzocrea et al., 2017), including investigating network traffic sent to and from the darknet, an anonymous network frequently used for malicious activities (Lingyu et al., 2017; Niranjana et al., 2020); and (iii) improving the Tor user experience by prioritising different types of Tor traffic generated from various applications (AlSabah et al., 2012) and revealing nonTor activities within the Tor network that could affect users' privacy (Hodo et al., 2017).

Based on a coarse classification of traffic analysis methodologies, active and passive

network monitoring are commonly employed to attack Tor networks. Active network monitoring refers to a type of monitoring technique where an observer interacts (e.g., modifying or injecting packets) with the network to gather information. In the context of Tor, active network monitoring encompasses various techniques, including the alteration of packet sizes by inserting a signal into the targeted Tor traffic between a Tor client and an exit relay. The altered traffic can then be observed and detected through this embedded signal (Fu et al., 2009; Ling et al., 2011, 2012). Other approaches to active monitoring include using decoy traffic for malicious Tor exit nodes, modifying user traffic parameters on the server-side and seeking similar modifications on the client-side via statistical correlation (Chakravarty et al., 2014). Although this network monitoring method efficiently and quickly verifies interaction relationships among users, it requires donating a number of computers as Tor relay nodes to control Tor traffic communication (Ling et al., 2012). Passive traffic analysis, on the other hand, passively monitors network traffic without modifying any packets. This approach involves collecting every packet traversing the network card to analyse and classify network traffic using various methods. This approach has been the subject of extensive research, which can be summarised as follows:

Some researchers use DPI techniques to identify traffic by examining details in associated TLS certificates, such as hostname, TLS handshake, process, certificate size, and port number. Since Tor uses different TLS characteristics from nonTor traffic, these features can discriminate Tor usage (Granerud, 2010; Lapshichyov & Makarevich, 2019; Mayank & Singh, 2017; Saputra et al., 2016). Chinese researchers (Bai et al., 2008) were among the first to identify Tor packets based on Tor traffic fingerprint characteristics, using combined DPI and packet frequency analysis methods to fingerprint Tor and Web-Mix network traffic. Their paper explains that Tor and Web-Mix have similar network topologies and unique IP packet characteristics. Both networks use unencrypted and encrypted traffic to exchange data depending on the communication pair. As a result, two procedures are considered to identify network traffic: First, packets are sent in an unencrypted format during the early stages of Tor and Web-Mix network establishment, and specific characteristics of payload strings are examined. Second, when

the traffic relay starts, packets are encrypted, and the matching length technique for Tor and dummy packet frequency analysis in Web-Mix are used. These two procedures yield a traffic fingerprint recognition rate of over 95% for both networks. However, these methods are slow and require considerable processing power. To overcome these drawbacks, ML approaches have been explored. These techniques are sometimes referred to as statistical analysis, as they rely on the analysis of statistical attributes or features such as packet size, packet length, flow segment size, round trip time, duration, and more. These statistical features enable the machine to learn and improve its performance through experience. Based on past research, the following three groups of features have been used as input to train supervised or semi-supervised algorithms, arranged by the difficulty of obtaining features and the means of data collection:

- **Packet-based** features are derived from packet size, frequency of packet transmission, and metadata from packet headers. Barker et al. (2011) and Lingyu et al. (2017) use these characteristics to classify Tor packets.

- **Flow-based** features are obtained from network flows in the form of packets that all contain the same value from five parameters (Source IP, Destination IP, Source Port, Destination Port, and Protocol). Some examples of flow-based features include the number of packets in the flow, the total size of the flow in bytes, the length of individual packets, the forward and backward times between packets, the duration of an active or idle flow, the size and number of bidirectional packets per second, and the duration of the flow. Most studies (Cuzzocrea et al., 2017; Hodo et al., 2017; Lashkari et al., 2017; Mercaldo & Martinelli, 2017; Shahbar & Zincir-Heywood, 2014; Soleimani et al., 2018) utilise these features from packet flows for Tor traffic classification.

- **Circuit-based** features refer to the time associated with Tor circuit construction and operation, including Tor cell transmission (e.g., circuit lifetime, data transmission rate, cell inter-arrival times (AlSabah et al., 2012), cells per circuit lifetime, number of uplink cells, and downlink cells to uplink cells time rate (Shahbar & Zincir-Heywood, 2014)).

Our study centres on utilising supervised learning to develop a model capable of distinguishing between Tor and nonTor traffic. In this regard, we will discuss several investigations that utilise supervised algorithms, as well as a few studies that employ DL.

Supervised Learning Approach

Barker et al. (2011) aimed to develop a simple passive recognition technique for detecting Tor traffic. They established a private Tor network with three directory servers and fifteen relays to simulate real-world traffic. They captured three sets of packets from 170 simulations over a seven-week period: regular HTTPS for two weeks, regular HTTP passing a private Tor network for two weeks, and HTTPS passing a private Tor network for three weeks. They used only packet sizes as attributes for classification. Their classification results showed that RFs and J48 were able to classify all three types of traffic with an accuracy of over 90%. In contrast, Adaboost failed to detect HTTPS over Tor. The packet-based features were also used in Lingyu et al. (2017). Only four key features, including packet length entropy, 600-byte packet frequency, zero data packet frequency in the first ten packets, and the mean of packet interval, were required to identify Tor traffic. When trained with the improved decision tree algorithm, the accuracy was 94%.

In the literature, two scenarios are most commonly addressed when using ML to classify Tor traffic, namely (a) detection of Tor traffic and (b) categorisation of Tor-based applications. In Lashkari et al. (2017)'s paper introduced the UNB-CIC dataset, which has been widely used in Tor classification research. The authors employed traffic flow features and implemented four classifiers to achieve high performance in both scenarios. For scenario (a), they obtained 96.4% and 97% for the weighted average of recall and precision of Tor and nonTor, respectively, using the C4.5 algorithm. For scenario (b), they achieved 84.3% and 83.8% for the same metrics using the RF algorithm. Similarly, Cuzzocrea et al. (2017) utilised the same dataset and flow-based features but used different classifiers. They found that JRip provided the best performance in scenario (a) with 100% for the weighted average of recall and precision, while J48 gave the best

performance in scenario (b) with 99.8%.

Another study that utilised the same dataset aimed to enhance the efficiency of the Tor network by detecting anomalous traffic or nonTor traffic that could compromise users' privacy. Hodo et al. (2017) employed a learning system to create a profile of regular traffic and defined any behaviour that deviates from the profile as outlier traffic. Their results, based on two algorithms, showed that a CFS-ANN hybrid classifier outperformed SVM in detecting nonTor traffic, achieving an overall accuracy of 99.8% and 94%, respectively.

Wang and colleagues L. Wang et al. (2020) conducted a recent research study that utilised network flow characteristics to distinguish Tor traffic. Their work builds upon previous studies that differentiate between various network traffic types on personal computer (PC) platforms. The authors extended this approach to compare the effectiveness of traffic classification across two platforms: PC and mobile. With the increasing prevalence of smartphone usage, this comparison is becoming more relevant. The study's findings demonstrate that the proposed method, which employs both time-related and non-time-related features, achieves near-perfect accuracy in identifying Tor traffic on both platforms. When utilising both types of features, the J48 supervised learning algorithm outperformed jRip, Naive Bayes, and Bayesian Networks. Platform-specific analyses showed that time-based features were more effective on PCs, while non-time-based features performed better on mobile devices. This difference is likely due to the less stable and more intricate nature of network traffic transmitted over mobile networks compared to wired PC connections.

In addition to utilising features derived from flow packets to classify Tor traffic, some researchers take into account the attributes extracted from the Tor network circuit. For instance, AlSabah et al. (2012) observed that different applications result in varying packet behaviours at the circuit (circuit lifetime and amount of data transferred by the circuit) and cell (cell inter-arrival times and the number of cells sent) level. In order to classify encrypted packets belonging to Tor circuits, they employed two forms of traffic classification. Offline classification only considers circuit logs in Tor entry guards, while online classification is performed during circuit operation and utilises both cell and circuit properties for a real-time classifier. Consequently, the effectiveness of the

classification is critical. Using Bayes Nets and Decision Tree, they achieved an accuracy of 97.8% and 91% for online and offline classification, respectively. Their proposed ML implementation, in conjunction with a defined Quality of Service (QoS) rule, can improve Tor network performance, aligning with their objective.

In their study, Shahbar and Zincir-Heywood (2014) expanded on previous works aimed at classifying Tor by incorporating flow and circuit-based features. Their objective was to compare the classification effectiveness at both feature levels. They hypothesised that the encrypted cells in the circuit have distinct behaviours and characteristics that are dependent on the type of traffic, to data traversing the Tor network. When trained with C4.5 on 70% of examples consisting of four features, namely cells per circuit lifetime, uplink cells, the rate of downlink cells to uplink cells, and the EWMA of the recently sent cell, the accuracy approached 100% at the circuit level. On the other hand, 112 flow features obtained from Tranalyzers and Tcptrace were learned by Bayes Net with a 10-CV, and they achieved 100% accuracy at the flow level. Notably, circuit-level features can only be extracted at the relay node, whereas flow-level features can be collected passively between the Tor client and Tor's relay.

Since the public availability of IP addresses for Tor relays has made it simple to restrict access to the Tor network by banning these relays' IP addresses. Additionally, DPI methods can be effective in detecting Tor traffic at the TLS protocol level by inspecting specific strings in the TLS certificate fields. To overcome this, Tor developers introduced bridges and pluggable transports to help obfuscate Tor traffic and bypass censorship or detection mechanisms. These techniques make Tor traffic appear similar to other common nonTor protocols, making it harder for DPI to identify the traffic as originating from the Tor network. Also, bridges are not publicly listed, preventing the blocking of Tor by concealing the IP addresses of the bridge nodes. However, research has successfully identified this 'pretend-to-be-nonTor' traffic. For instance, Shahbar and Zincir-Heywood (2015) used flows extracted by Tranalyzer and trained with C4.5 to identify flows from five different pluggable transports (Flashproxy, Scramblesuit, FTE, Meek and Obfs3) of Tor bridges. Their overall correctly classified result is 97%, demonstrating that pluggable transport flow behaviours can be discriminated from other

network traffic as they possess unique fingerprints. Similarly, Soleimani et al. (2018) demonstrated that three commonly used Tor pluggable transports (Obfs3, Obfs4, and ScrambleSuit) can be detected using their ML methodology utilising a set of bidirectional flow statistics, such as total flow volume, mean packet length, and standard deviation of packet length that is unaffected by the contents of the flow. The authors only examine the first 10 to 50 packets, which are involved in the establishing activity and have a distinct connection pattern that helps identify the flow. Four supervised learning algorithms, including Adaboost, RFs, SVM, and C4.5, were utilised to classify distinct traffic types along a 70 to 30% split, resulting in optimal F-measure and AUC values. The authors concluded that traffic recognition produces better results as the number of analysed packets increases, particularly when more than 30 packets are used in the training phase.

Furthermore, due to the well-known fact that the Tor network's hidden services are a shelter for criminal activities, there have been efforts to not only classify regular Tor traffic and Tor bridge traffic but also to detect Tor hidden services. Kwon et al. (2015) proposed an attack on Tor's hidden services by examining their fingerprinting. They claim to be the first passive detection system that successfully identifies traffic from hidden servers amidst regular Tor traffic. Flow traffic attributes, such as the total transmission size and time, the number of incoming and outgoing packets, the location of each outgoing cell, and the number of consecutive cells of the same type, were used as inputs for their three supervised learning algorithms. They discovered that kNN outperformed C4.5 and CART for classifying hidden services.

Deep Learning Approach

As discussed in the previous section, the prominent feature of DL in effectively modelling data without requiring handcrafted features has led to its increasing popularity in NTC. However, despite its success in classifying network traffic, there has been relatively little research on the application of DL in the area of Tor network classification. In this context, two studies that focused on the classification of Tor traffic are presented below for instances.

One study that closely aligns with our approach to utilising payload-based features is that of Kim and Anpalagan (2018). The UNB-CIC dataset, which is renowned in the Tor classification field, was employed in the study to enable comparison with the results obtained by the dataset creators, Lashkari et al. (2017). This study proposes an approach for classifying encrypted Tor traffic using a 1D-CNN model with raw packet headers as input. Specifically, the first 54 bytes of TCP packets, including the TCP/IP header and Ethernet II header, are utilised as input. These hex values are then converted to decimal values. The proposed CNN model performs packet classification without any manual feature extraction or engineering. The results of the study demonstrated that the proposed CNN model outperformed the C4.5 algorithm in terms of precision and recall in scenario A, which involved classifying Tor and nonTor traffic. Specifically, the CNN achieved perfect precision and recall, whereas the C4.5 algorithm achieved 0.95 and 0.93, respectively. In scenario B, which involved classifying application-level Tor traffic, the CNN model also outperformed the C4.5 algorithm for all applications. The CNN achieved a range of 0.86-0.99 in precision and 0.84-1.0 in recall, whereas the C4.5 algorithm ranged from 0.62-0.98 in precision and 0.48-0.97 in recall. These results suggest that the proposed approach can effectively classify Tor traffic without the need for manual feature extraction or engineering, which is a significant advantage of DL.

It is widely acknowledged that DL is well known for its ability to extract features automatically, without the need for human intervention. However, it is also possible to incorporate handcrafted features into deep learning models, as demonstrated in the study by Sarkar et al. (2020). The authors presented a DNNs for detecting Tor traffic using DL, which was tested on the UNB-CIC Tor network dataset. However, the study employed human-crafted features based on time-based features provided by the dataset owner. The authors used feature selection methods such as Symmetric Uncertainty (SU) and Correlation-Based Filter Selection (CFS) to train their DNNs models using 25 out of the 28 available features. In scenario A, the best result achieved an accuracy of 99.89% with DNNs, while in scenario B, the accuracy was 95.60%. These results were significantly better than those achieved using C4.5, which was employed in the dataset creator's work.

While many studies have shown that flow-based features are effective for Tor traffic classification, there are some limitations to this approach as discussed in Section 2.6.4. The latter two studies have demonstrated that DL methods have shown superior performance compared to traditional classification approaches, particularly by utilising automated feature extraction capabilities (Sarkar et al., 2020). However, our work requires manual feature extraction, and there are limitations associated with using DL, as discussed in Section 2.6.1. These limitations justify our use of traditional ML methods, applying character statistical-based features to achieve our objectives.

2.8 Summary

In the first part of this chapter, we provided an overview of various technologies essential to appreciate the background and setting of our research. We introduced cryptography principles, different Internet protocols and their encryption, and discussed the Tor network architecture and encryption techniques that ensure anonymity for its users. We also compared Tor and nonTor packets and explored character analysis, a fundamental concept in our study's methodology. Additionally, we presented statistical analysis and essential concepts of ML relevant to our research.

The second part of this chapter focused on related work in NTC, specifically for Tor traffic. We presented diverse approaches used to classify network traffic, along with their advantages and disadvantages. We also reviewed recent studies that demonstrated the potential of ML and DL approaches in achieving high accuracy in Tor traffic classification.

The following chapter gives a detailed account of the methodology adopted in our research, building on the background and related work presented in this chapter.

Chapter 3

Research Methodology and Experimental Setup

This chapter is divided into two main sections: research methodology and experimental setup. The methodology section outlines the research approach and framework, including data requirements, proposed features, data collection, data preprocessing, statistical analysis, and ML techniques. Additionally, in methodology, we discuss the tools used and measures taken to ensure the validity and reliability of each stage in the study framework. The experimental setup section focuses on the implementation of the methodology, which includes collecting the data needed for analysis, handling the raw data, and preparing the system's environment to produce accurate findings.

3.1 Methodological Approach

A research methodology is established at the outset of research, addresses research problems and draws new conclusions by systematically working toward a predetermined goal. Our research primarily focuses on characterising Tor traffic through a quantitative approach that involves observing character statistics in encrypted payloads. This approach is appropriate because integrating quantitative data into research can help generate authentic and relevant research findings by statistically testing and estimating the outcomes (Fletcher, 2017). Our chosen methodological approach is appropriate for

Tor traffic classification based on the statistics of character features. The attributes used were derived from the hex characters that appear in the encrypted payloads of Tor and nonTor packets, yielding numerical values. For example, quantitative data is derived from Tor and nonTor network attributes to compare descriptive statistics, which is then presented graphically to help readers understand the results. Quantitative data can also be used for the classification phase since our experiments require numerical data that can be evaluated and validated relative to our research objectives. The methodology adopted in this study is in line with the work by Wiek and Lang (2016), which stated that the quantitative methodological approach is more efficient for research since it emphasises measurement of objectives such as statistical, numerical, and graphical analysis.

In addition to selecting an appropriate methodological approach, it is crucial to determine a research design that provides clear direction for procedures in a research study (Creswell & Creswell, 2018). Our study involves quantitative data for statistical and ML analysis, which falls under the category of experimental research. This approach involves manipulating the independent variable and observing the resulting changes in the dependent variable. Our study aims to investigate the characteristics of encrypted payloads between Tor and nonTor networks, as outlined in null hypothesis H_{01}. Therefore, the independent variable is the network type (Tor or nonTor), and the dependent variable is the characteristics of the encrypted payloads, such as the hex character frequency, hex character frequency ratio, total characters, and entropy. Null hypothesis H_{02} seeks to determine if a single encrypted payload is sufficient to identify Tor traffic. In this case, the independent variable is the classifiers used to identify Tor traffic when using a single payload, and the dependent variable is the efficiency of these classifiers.

Based on the methodological approach and experimental research of this study, we have developed a research methodology that focuses on testing research hypotheses using the scientific method. Fundamentally, the scientific method (Dewey, 1933; Hoy & Adams, 2015; Kerlinger, 1966) involves four key steps for acquiring knowledge based on empirical evidence: defining a problem, formulating a hypothesis, justifying the hypothesis, and observing and designing an experiment for testing the hypothesis. Adopting the scientific method, the following steps were included in the proposed research methodology: the

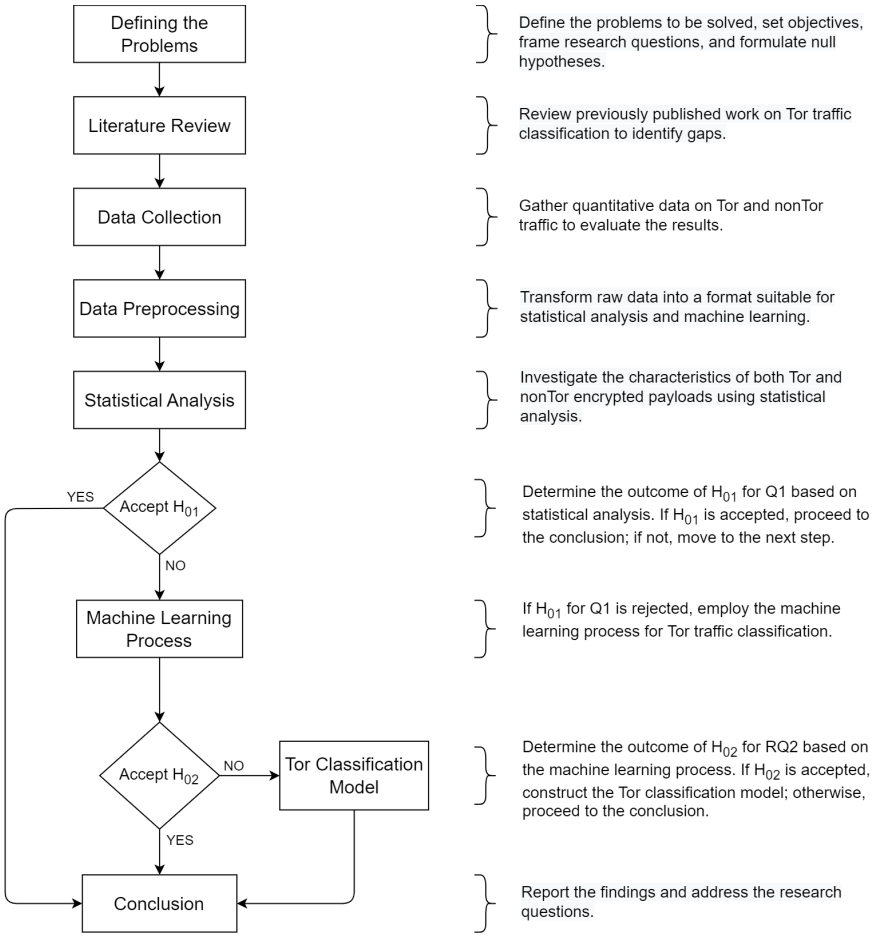


Figure 3.1: Proposed methodology of the thesis (Adapted from Dewey (1933) and Kerlinger (1966))

first step is to define the research problem, objectives, research questions, and hypotheses. The *second step* is to review previous work on the identified problem on Tor network traffic classification. The *third step* is to collect data. The *fourth step* is to convert raw data into well-formed data that can be used for statistical analysis and as input to ML models. The *fifth step* examines the characteristics of encrypted payloads from Tor and nonTor networks to test the null hypothesis of the first research question. If the null hypothesis is not rejected, the findings are concluded. Otherwise, the ML process continues to investigate the Tor traffic classification. The classification results are used to evaluate the null hypothesis of the second research question. If the hypothesis is not rejected, the findings are concluded. Otherwise, a Tor traffic classifier is produced, which

is in the *sixth step*. Finally, in the *seventh step* the findings of the study are reported, and the research questions are answered.

A comprehensive understanding of our quantitative research design and the proposed methodology, which follows a logical sequence, has been described. This understanding will aid in conducting the practical tasks of the study in a scientific manner, ultimately fulfilling the research objectives. The following section discusses the data requirements that need to be examined first in order to collect research data appropriately. This examination is essential to ensure that we collect the right type of data, with appropriate measures and methods, to answer our research questions and test our hypotheses effectively.

3.2 Data Requirements

Data is a critical component of research, as it forms the foundation for generating research findings (Hofmann, 2013). Before initiating the data collection process, it is essential to identify and understand the data requirements of the study. For our study, which aims to classify network traffic, we require that the collected network data encompass secure network protocols running over the transport layer on the TCP/IP stack, resulting in encrypted packets. Meeting these data requirements will enable us to ensure that the research data collected is relevant, accurate, and sufficient to conduct analyses that effectively address our research questions and objectives.

In networking, data collection is typically classified into two types based on the captured packets approach: *active* and *passive* (D. Zhou et al., 2018). The active approach involves modifying packets by inserting data into traffic and observing the responses for network traffic monitoring purposes. In contrast, passive data collection involves passively monitoring network traffic without modifying any packets, capturing all flow packets passing through the network card using network traffic monitoring tools to analyse and classify the network traffic. For our study, we used passive data collection to capture and analyse network traffic data. The analysis of Tor traffic classification necessitates the collection of network traces from multiple applications operating on both

Tor and nonTor networks. These traces represent encrypted payload data derived from various security protocols. Passive traffic capture is employed to collect the data, which can be conveniently executed on the client-side without interfering with the Tor nodes. To ensure consistent results and accurately validate the assumptions, the collected data from a diverse range of applications on both networks must be sufficiently dense and robust.

In academic research, publicly available datasets, also known as *secondary datasets*, are generally preferred as they save time and money and are often of high quality. However, when specific criteria for the research cannot be met by any available data, researchers obtain self-gathered datasets, also known as *primary datasets*. In our study, we focus on analysing the statistics of characters that appear in encrypted payloads. As discussed in Section 2.1.1, it is acknowledged that modern encryption algorithms are highly effective (Katz & Lindell, 2020), and should make it infeasible to determine traffic type solely by inspecting encrypted payloads. Consequently, character statistics-based analysis is likely to fail with encryptions, as ciphertext strings cannot be interpreted. In light of this assumption, we chose also to collect a self-gathered dataset to ensure independence and consistency within the heterogeneous networking environment. It is important to note that our study aims to demonstrate the character statistics-based approach applied with modern encryption to investigate the characteristics of Tor and regular nonTor traffic, rather than revealing message information from the ciphertext. To summarise, our study employs two types of dataset sources. The first is a publicly available dataset. The second is a privately gathered dataset that supports the findings of the publicly available dataset. The process of collecting network traces for the private dataset will be elaborated in Section 3.5.1.

3.3 Proposed Features

As discussed in Section 2.2.1, network analyser software typically presents ciphertext in three data formats: binary, ASCII, and hex. The binary format, consisting of only 0s and 1s, becomes lengthy and thus more challenging to visually interpret for larger numbers. ASCII characters comprise a set of 256 characters, including both printable

and non-printable ones. The broad range of characters in ASCII, along with the inclusion of non-printable control characters, adds layers of complexity to the analysis process. For instance, the wide range of characters makes it difficult to quickly identify patterns, and the presence of non-printable characters can make visual inspection challenging. In contrast, the hex format uses a base-16 number system that includes 16 characters (0-9, a-f). It offers a more compact representation of data, which can make reading and interpreting larger values more intuitive. This format can represent both printable and non-printable ASCII characters. In light of these considerations, our analysis emphasises single-hex characters over the two-hex characters commonly displayed in network analysers. Similar to the advantages observed in feature selection that reducing features can improve the model efficacy, a smaller feature set can lead to improved analytical efficiency. Simplifying pattern identification might enhance both the overall effectiveness and the speed of the operational process. As a result, our approach relies upon statistics of 1-hex values, as presented below:

(1) A set of hex frequency (*F_0-F_f*): This feature measures the frequency of each hex character (0-9, a-f) in the encrypted payloads. Though encryption aims to make the content of the payload unreadable, the frequency of each character can still be observed. The analysis of the frequency of each hex character in the encrypted payloads can reveal patterns and characteristics of the encrypted data. By examining the frequency of each character, it is possible to identify which characters occur more frequently and which are rare. This information can be used to identify specific patterns and anomalies in the data that may be useful in classification or other analysis purposes. For example, it may be possible to identify specific types of network traffic based on the frequency of certain characters in their encrypted payloads. Additionally, by comparing the frequency of characters in encrypted payloads across different networks or applications, it may be possible to identify differences or similarities in their encryption methods.

(2) A ratio set of hex frequency(*R_0-R_f*): This feature measures the ratio of each hex character's frequency (0-9, a-f) to the total frequency of all hex characters in the encrypted payload. It aims to normalise the frequency distribution by accounting for potential variations in payload sizes. This method ensures length normalisation and

independence, reducing biases or inaccuracies that may occur when analysing encrypted payloads based solely on absolute packet length.

(3) The total number of characters (T): This feature measures the total number of characters in the encrypted payloads, which corresponds to the payload length, by counting individual hex characters in the ciphertext. This metric can be used to identify patterns in the data and to compare the size of payloads across different networks or applications.

(4) Entropy (E): This feature measures the randomness of the encrypted payload by calculating the Shannon entropy of the payload. It serves as a metric to quantify the randomness or unpredictability of data. In the context of encrypted payloads, higher entropy signifies a more random distribution of hex, which is anticipated in well-encrypted data and generally preferred for secure communication, as discussed in Section 2.1.3. Ideally, encrypted data should display a uniform distribution of values (Gancarczyk et al., 2011).

3.4 Research Methods

In addition to the methodological approach, a research design must be determined to provide clear direction for procedures in a research study (Creswell & Creswell, 2018). A research design involves the planning of systematic research operations to efficiently address problems while adhering to the research objectives or hypotheses. It contains several procedures that serve as a blueprint for the researcher (Dulock, 1993; Yin, 2009) to construct the research framework action plan for the investigation. The research methodology of this study consists of a series of steps that must be executed to achieve the research objectives. The procedures and instruments employed in each step of this research are detailed in the following section.

A research framework represents the underlying structure or model of a research plan by outlining completed procedures throughout the study. It serves as a guide to narrow down the scope of research. Figure 3.2 depicts the framework for our research. Our research framework consists of four major phases. The *first phase* involves the procedure for collecting datasets. Our study employs both primary and secondary datasets, which

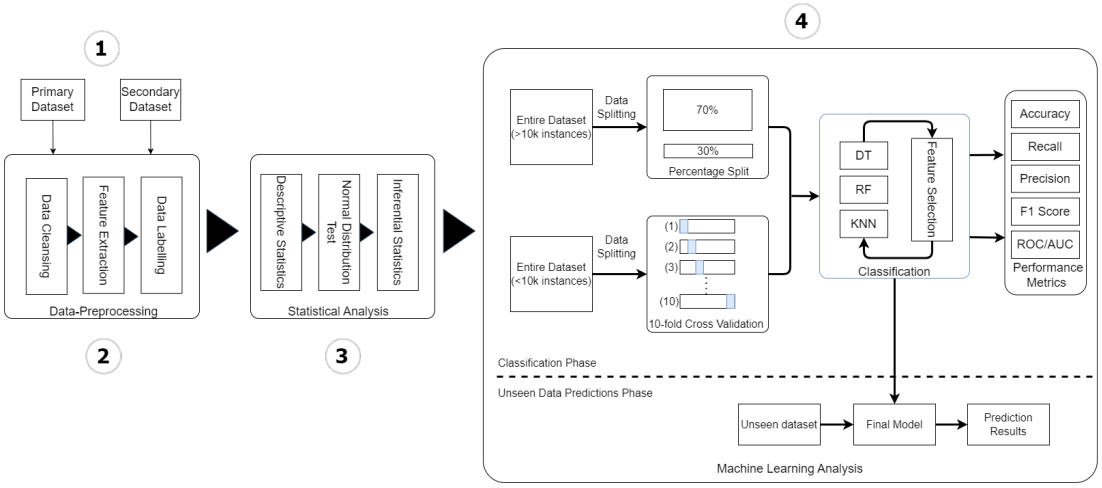


Figure 3.2: Research framework of the thesis

will be implemented and analysed using the same procedures. For the primary dataset, publicly available data is obtained. For the secondary dataset, which requires collecting data privately, a network analyser tool, called Wireshark, is used to collect network traces from both Tor and nonTor networks. Details for the acquisition of the two types of data sources can be found in Section 3.4.1. The *second phase*, data preprocessing, takes raw data from phase one and transforms it into a format that can be understood and analysed by statistical and ML methods. Several tools are involved in the preprocessing of raw data. This phase includes three sub-steps: data cleansing, feature extraction, and data labelling, which will be described in detail in Section 3.4.2. The *third phase*, statistical analysis, includes descriptive and inferential statistics tests. The test results will help address H_{01}: There is no difference between Tor and nonTor traffic in terms of encrypted payloads. This phase consists of three sub-steps: descriptive statistics, normal distribution tests, and inferential statistics, detailed in Section 3.4.3. All statistical analysis tests are conducted using Statistical Package for the Social Sciences (SPSS), a widely used statistical software package. The *fourth phase*, ML, which involves all steps for classification using three supervised learning algorithms: DTs, RFs, and kNN. It consists of two major parts: classification and prediction of unseen data. The outcomes of this phase are key to addressing H_{02}: A single encrypted payload cannot be used to identify Tor traffic. Section 3.4.4 demonstrates how to perform the ML processes. The

well-known ML platforms, WEKA and Scikit-learn are used to run classification tasks.

3.4.1 Datasets Collections

A dataset is a collection of data that is used to answer research questions. The process of gathering data is known as data collection, and it involves collecting information from various sources using different methods such as surveys, experiments, and observations. It is important to ensure that the data collected is of high quality and represents the population being studied. As previously mentioned in Section 3.2, this study employs both public and private datasets to address the research questions. By utilising two sources of datasets, the study can ensure the validity of the findings and provide a more accurate and complete analysis of the study. The following section provides a detailed description of both types of datasets used in this study.

Public Dataset Description

Before 2016, Tor datasets were not publicly available, so research on identifying Tor traffic relied entirely on private datasets. However, freely downloadable Tor datasets have since been released for academic use in Tor traffic studies. Using accessible datasets as a starting point ensures trustworthiness and previous results can serve as benchmarks.

Two openly available Tor network traffic datasets are ISCXTor2016[1] (Lashkari et al., 2017) and Anon17[2] (Shahbar & Zincir-Heywood, 2017). ISCXTor2016 is a traffic dataset that contains eight application types in Tor and nonTor networks, stored in PCAP[3] format and in ARFF[4] format. This dataset is owned by the Canadian Institute for Cybersecurity (CIC) at the University of New Brunswick (UNB) in Canada, making it also known as the UNB-CIC dataset. Anon17 comprises three datasets of anonymised network traffic in ARFF format, collected by the Network Information Management and Security lab (NIMS) at Dalhousie University in Canada: Tor, JonDonym, and I2P. Since our research aims to investigate encrypted payloads, we prefer network traffic in PCAP

[1]ISCXTor2016 can be downloaded from https://www.unb.ca/cic/datasets/tor.html
[2]Anon17 can be downloaded from https://projects.cs.dal.ca/projectx/Download.html
[3]a raw network trace that can be structured into a TCP/IP model using Wireshark
[4]a ML file compatible with WEKA

format; thus, we use the ISCXTor2016 dataset.

ISCXTor2016

ISCXTor2016 or UNB-CIC (Lashkari et al., 2017) dataset has been widely used in the research area of Tor traffic. The dataset contains Tor and nonTor traffic data collected from real-world traffic via Whonix, a Tor-integrated open-source Operating System (OS) based on Linux. Whonix consists of two Debian GNU/Linux virtual machines: a workstation and a gateway. Traffic captured by the Whonix workstation is regular traffic, whereas traffic captured by the Whonix gateway is Tor traffic since it is routed through the Tor network. The dataset was collected by simultaneously recording network activities on both virtual machines using a packet sniffing tool (e.g., Wireshark) and storing regular traffic and Tor traffic as separate PCAP files in Tor and nonTor directories. Figure 3.3 illustrates the Whonix architecture.

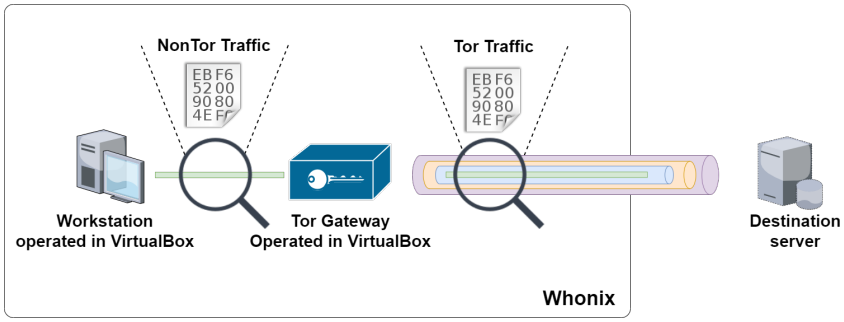


Figure 3.3: ISCXTor2016 dataset collection (Adapted from (Lashkari et al., 2017))

The ISCXTor2016 dataset developer released two types of files in their repository[5] for download: TorCSV.zip and TorPcap.zip. TorCSV.zip contains time-based features and is labelled in ARFF format, derived from TorPcap.zip files using ISCXFlowMeter[6]. Our focus is on the 20 GB TorPcap.zip file, modified on September 9, 2019, at 13:46. This file contains eight traffic types (Audio, Browsing, Chat, Email, FTP, P2P, VDO, and VoIP) from over 18 software applications (e.g., Firefox, Chrome, Facebook, Skype, Gmail, etc.), offering a diverse range for our research purposes. The ISCXTor2016 was

[5]http://205.174.165.80/CICDataset/ISCX-Tor-NonTor-2017/Dataset/

[6]ISCXFlowMeter (now renamed as CICFlowMeter) is a Java-based network traffic analyser developed by the Tor dataset team that reads and converts PCAP data to CSV files. It can produce bidirectional network traffic and calculate over 80 statistical flow-based attributes.

captured and classified through the following means, representing real-world internet usage:

Audio-Streaming: Continuous audio data streams from Spotify were used to capture audio traffic.

Browsing: HTTP and HTTPS traffic were collected using Chrome and Firefox.

Chat: Chat traffic was collected using Facebook and Hangouts instant-messaging applications via web browsers, as well as Skype, IAM, and ICQ using Pidgin.

Email: Thunderbird was used to generate email traffic. Emails were sent via SMTP/S and received via POP3/SSL on one client and IMAP/SSL on the other.

FTP: FTP traffic was collected when sending or receiving files via Skype file transfers, FTP over SSH, and FTP over SSL traffic sessions.

P2P: File-sharing protocols, such as BitTorrent, were generated and captured as Peer-to-Peer (P2P) traffic using the Vuze application.

Video-Streaming: Video traffic was captured with continuous video data streams from YouTube (HTML5 and Flash versions) and Vimeo services via Chrome and Firefox.

VoIP: Voice call traffic was captured from Facebook, Hangouts, and Skype.

Private Dataset Description

To demonstrate that the results are not influenced by any particular network environment, we expand our experiment to include a self-collected dataset in addition to the initial public dataset. Due to time constraints and to simplify the system setup, we choose only one Browsing application instead of capturing Tor and nonTor traffic from eight applications like in the public dataset. Encrypted Browsing traffic data is collected from Firefox in both regular and Tor networks. Collecting data from the browsing application is made carefully to ensure that irrelevant network packets are excluded from the dataset, thus improving the accuracy and relevance of the results. A detailed description of the private data collection process is presented in Section 3.5.1.

3.4.2 Data Preprocessing

Data preprocessing is an essential step in the data analysis process because it involves preparing raw data by transforming, cleaning, and formatting it into a structured dataset that can be used for further analysis. The preprocessing stage is necessary because raw data is often incomplete, noisy, and unformatted, and cannot be used directly for analysis. Meaningful patterns can be identified by filtering and extracting relevant features from preprocessed data, which can provide valuable insights into a variety of problems. Data preprocessing in this study comprises three sub-processes: data cleansing, feature extraction, and data labelling, as illustrated in the second step in Figure 3.2.

Data Cleansing

Data cleansing is a crucial step in the data preprocessing phase as it involves removing irrelevant data from the acquired dataset. In this study, our focus is on network traffic classification, which involves raw network traffic data in PCAP format as input. As described in Section 3.4.1, the publicly collected data was categorised into eight application types, including Audio, Browsing, Chat, Email, FTP, P2P, VDO, VoIP, and Private-browsing, generated from 18 software, and one from the private dataset. To ensure consistency and enhance the results, the datasets corresponding to the same application types in the public dataset are grouped into a single dataset, following the same grouping pattern used for categorisation. This resulted in nine groups of application types for both Tor and nonTor networks across the two dataset sources. Grouping the captured network traffic can also provide benefits for investigating the security protocols utilised by these applications. In the context of our study, the security protocols found in the datasets are TLS, SSH and secure proprietary protocols.

During the data cleansing process, there are two factors to consider to ensure that only relevant data is retained. Firstly, encrypted packets must be generated by network traffic from specific applications for secure communication. As explained in Section 2.2, this operation is usually performed by considering only secure protocols. Therefore, any application that does not provide secure communication is disregarded. Secondly, only packets containing payload need to be obtained. Captured network traffic typically

consists of all packet types involved in device communication, travelling in and out of the network interface card. This includes the initial TCP handshake packets, which do not contain any payload data but rather information required to establish the connection located in the packet's header. As a result, network connection-related packets must be ignored because only packets containing application data or payload are considered. The data cleansing process with the inclusion of these two conditions is essential in improving the quality of the data for analysis and ultimately, in obtaining more accurate results.

Algorithm 1 provides the data cleansing pseudocode, an overview of the process to extract relevant packets containing payload while filtering out unimportant packets like TCP handshake flows. The operation takes the network flow in PCAP format as input and processes each packet individually.

Algorithm 1 Data Cleansing Algorithm

Require: F: Network flow in PCAP format
Ensure: E: The encrypted payload

```
1: i ← 0 //Initialise with the first packet in the PCAP file
2: while i < F do
3:     Read packet i
4:     if topmost_layer_protocol = "TLS" then
5:         Read payload i
6:         if payload i != "0" then
7:             E ← Get tls.app_data //Extract non-empty TLS payloads and append to the encrypted payload set E
8:         end if
9:     else if topmost_layer_protocol = "SSH" then
10:        Read payload i
11:        if payload i != "0" then
12:            E ← Get ssh.encrypted_packet //Extract non-empty SSH payloads and append to the encrypted payload set E
13:        end if
14:    else if topmost_layer_protocol = "TCP" or proprietary protocol then
15:        Read payload i
16:        if payload i != "0" then
17:            E ← Get tcp.data //Extract non-empty TCP payloads and append to the encrypted payload set E
18:        end if
19:    end if
20:    i ← i + 1
21: end while
22: return E
```

During the processing, the algorithm iterates through each packet:

1. If the topmost layer protocol is TLS, the algorithm checks if the payload is non-empty. If so, it uses the command 'tls.app_data' to extract the payload and adds it to the encrypted payload set (E).

2. If the topmost layer protocol is SSH, the algorithm checks if the payload is non-empty. If so, it uses the command 'ssh.encrypted_packet' to extract the payload and adds it to the encrypted payload set (E).

3. If the topmost layer protocol is TCP or a proprietary protocol, the algorithm checks if the payload is non-empty. If so, it uses the command 'tcp.data' to extract the payload and adds it to the encrypted payload set (E).

The algorithm iterates through all packets in the network flow. After the entire network traces have been processed, the extracted encrypted payload set (E) is returned as the output. This approach ensures that only relevant payload data is retained for further analysis, while unnecessary packets are excluded from the dataset.

Feature Extraction

Feature extraction is the process of deriving meaningful attributes from the collected and cleansed data, which is then transformed into a well-formatted dataset suitable for data analysis. This study employs statistical and classification analyses. Before performing classification analysis, the extracted features undergo statistical analysis to identify the most relevant and informative features. Properly designed features are crucial for effective statistical analysis and improved prediction accuracy, enabling precise problem-solving using a trained model with appropriate features (Robertazzi & Shi, 2020). As mentioned in Section 3.3, our proposed features include a set of hex statistic calculations. To obtain these statistics, we use the mathematical equations presented below.

The frequency of each hex character in a single payload can be computed using the following equation:

$$F_{c_i}(p) = \sum_{k=1}^{|p|} \delta(c_i, p(k)) \tag{3.1}$$

$$\forall i \in \{0, 1, \ldots, F\}$$

$$\delta(c_i, p(k)) = \begin{cases} 1, & \text{if } c_i = p(k) \\ 0, & \text{otherwise} \end{cases}$$

In this equation, $F_{c_i}(p)$ represents the frequency of hex character c_i in payload p. The hex character c_i ranges from 0 to F. The payload p has a length denoted by $|p|$, and k is an index variable iterating through each character in the payload. The indicator function $\delta(c_i, p(k))$ takes the value 1 when hex character c_i matches the k-th character of payload p and 0 otherwise.

Equation 3.1 calculates the frequency of each hex character c_i in the payload p by iterating through the payload's characters and comparing them with the hex characters. The indicator function $\delta(c_i, p(k))$ counts the matches, and the summation accumulates the counts to compute the final frequency for each hex character in the payload.

The total character count in a payload can be computed using the following equation:

$$T(p) = \sum_{i \in 0,1,\ldots,F} F_{c_i}(p) \tag{3.2}$$

In Equation 3.2, $F_{c_i}(p)$ denotes the frequency of hex character c_i in payload p as calculated using Equation 3.1. The summation in Equation 3.2 adds up the frequencies of all hex characters to obtain the total character count in the payload.

The ratio of each hex character in a payload can be computed using the following equation:

$$R_{c_i}(p) = \frac{F_{c_i}(p)}{T(p)} \tag{3.3}$$

In this equation, $R_{c_i}(p)$ represents the ratio of hex character c_i in payload p. The

hex character c_i ranges from 0 to F. $F_{c_i}(p)$ denotes the frequency of hex character c_i in payload p, as calculated using Equation 3.1. The payload p has a total character count represented by $T(p)$.

Equation 3.3 computes the ratio of each hex character c_i in the payload p by dividing the frequency of hex character c_i (calculated using Equation 3.1) by the total character count in the payload $T(p)$. This ratio represents the proportion of each hex character in the payload, which is useful for analysing the distribution of hex characters in encrypted data.

The entropy of a payload can be computed using the following equation:

$$H(p) = -\sum_{i \in 0,1,\ldots,F} R_{c_i}(p) \log_2 R_{c_i}(p) \tag{3.4}$$

In this equation, $H(p)$ represents the entropy of payload p. $R_{c_i}(p)$ denotes the ratio of hex character c_i in payload p, as calculated using Equation 3.3. The hex character i ranges from 0 to F.

Equation 3.4 calculates the entropy of a payload p by summing the product of the ratio of each hex character c_i in the payload and the base-2 logarithm of that ratio.

These equations are integrated into Algorithm 2 to calculate four sets of features in a series of payloads. The algorithm provides the pseudocode for the feature extraction process. The input to the algorithm is an extracted encrypted payload (P), and the output is a set of relevant features derived from the payload. These features include the frequency of individual hex characters appearing in the encrypted payload (F), the frequency ratio of individual hex characters (R), the total number of characters in the encrypted payload (T), and the entropy of the encrypted payload (E).

Algorithm 2 Feature Extraction Algorithm

Require: P: Extracted encrypted payloads
Ensure: F: Frequency of individual hex characters in the encrypted payloads
R: Ratio of the frequency of individual hex characters in the encrypted payloads
T: total characters in the encrypted payloads
E: entropy in the encrypted payloads

```
for all packet in P do
    if packet contains "," then
        Split and append a new row
    end if
    F ← Frequency count //calculated using Eq. 3.1
    R ← Ratio of F //calculated using Eq. 3.3
    T ← Total Character //calculated using Eq. 3.2
    R ← Entropy of P //calculated using Eq. 3.4
end for
return F, R, T, E
```

The algorithm iterates through each packet in the extracted encrypted payload (P). If the packet contains a comma (‘,’), it splits the packet and appends a new row. Then, for each packet, the algorithm performs the following actions:

1. Counts the frequency of each character in the payload (F).
2. Normalises the frequency count by calculating the ratio of individual hex characters (R).
3. Calculates the total number of characters in the payload (T).
4. Calculates the entropy of the payload (E).

After processing all packets in the payload, the algorithm returns the extracted features: F, R, T, and E. These features will be used for further statistical analysis and classification tasks.

Building upon this concept, we developed a Python-based script called *Character Statistics Analysis using Python (CHARSTAT)* for research purposes. This script processes data cleaning and extracts the encrypted payloads and relevant features described earlier. Utilising CHARSTAT simplifies the task of data preparation and enhances the precision of the resulting features. This, in turn, ensures the efficacy of statistical and classification analyses, leading to more accurate and reliable outcomes.

Data Grouping and Labelling

As described in Section 3.4.1, the publicly collected data was categorised into eight application types, including Audio, Browsing, Chat, Email, FTP, P2P, VDO, VoIP, and Private-browsing, generated from 18 software, and one from the private dataset. To ensure consistency and enhance the results, the datasets corresponding to the same application types in the public dataset were grouped into a single dataset, following the same grouping pattern used for categorisation. This resulted in nine groups of application types for both Tor and nonTor networks across the two dataset sources. By grouping multiple datasets based on application types, it can mitigate the impact of software-specific characteristics on the analysis and enhance the statistical study by integrating a larger sample size. This approach can improve the reliability and generalisability of the results, particularly when examining and comparing Tor and nonTor traffic across various applications.

Following the grouping of application types, data labelling is performed. In ML, data labelling is crucial for training models using supervised learning algorithms, as the accuracy of the classifier's learning capabilities depends on both the extracted features and the labelled data. Moreover, in our study, statistical tests require the use of labelled data to validate the analysis. The labelled data is assigned following the group of application types, resulting in multiple files that contain labels for nine pairs of Tor and nonTor networks across the two dataset sources. The labelling process is integrated into CHARSTAT, enabling automatic labelling of the data, which provides a more accurate and efficient labelling process compared to manual labelling. Section 3.4.2 provides a detailed description of the data labelling.

3.4.3 Statistical Analysis

This study employs statistical analysis to test the research hypothesis by investigating and comparing the characteristics of Tor and nonTor traffic using three subprocesses, as shown in the third phase of Figure 3.2. As mentioned in Section 2.5, descriptive analysis, including Data distribution(*Frequency*), average (*Mean*), standard deviation (*SD*), minimum (*Min*), and maximum (*Max*), is calculated from continuous quantitative

data. These measurements are presented both quantitatively and graphically to facilitate comparison. Prior to applying inferential statistics to scientifically validate the research hypothesis, the data is subjected to a normality test. The Kolmogorov–Smirnov test is used for this purpose, and if the observed data does not show a normal distribution, the Mann–Whitney U test is applied for inferential analysis.

Statistical analysis is conducted on nine groups of application types from both dataset sources. To compare the characteristics of encrypted payloads in Tor and nonTor networks efficiently, it is essential to investigate each aspect of character analysis based on payload features. Therefore, the 34 features discussed in Section 3.5.2 derived from encrypted Tor and nonTor payloads belonging to nine sets of applications will be analysed. The use of these nine sets of applications ensures data analysis consistency and strengthens the validity of the study's findings. In the context of statistical analysis, all elements can be interconnected, as illustrated in Figure 3.4.

The linkage of each object involving three elements can be described as follows: (1) two types of network traffic (Tor and nonTor), (2) nine types of applications (Audio, Browsing, Chat, Email, FTP, P2P, VoIP, VDO, and Private-Browsing), and (3) four sets of features. These four sets of features, which consist of 34 features (*F_ 0-F_f, R_ 0-R_f, T, E* derived from the nine Tor applications, are mapped one by one to the corresponding 34 extracted features obtained from the nine nonTor applications. Descriptive statistics, including *Frequency, Mean, SD, Min* and *Max*, are used to describe and compare the characteristics of each pair of features. The Mann-Whitney U test is then used to evaluate these characteristics and provide insight into the first null hypothesis. The statistical analysis is covered in Chapter 4. All statistical tasks are performed using the (SPSS), a powerful statistical software that simplifies the work.

In summary, the statistical analysis aims to provide a comprehensive understanding of the characteristics of Tor and nonTor traffic by examining all aspects of extracted features derived from various applications. The use of both descriptive and inferential statistics allows researchers not only to describe the trends, patterns, and relationships within the data but also to draw conclusions about the larger population. This approach helps to validate the research hypothesis and provides valuable insights for further

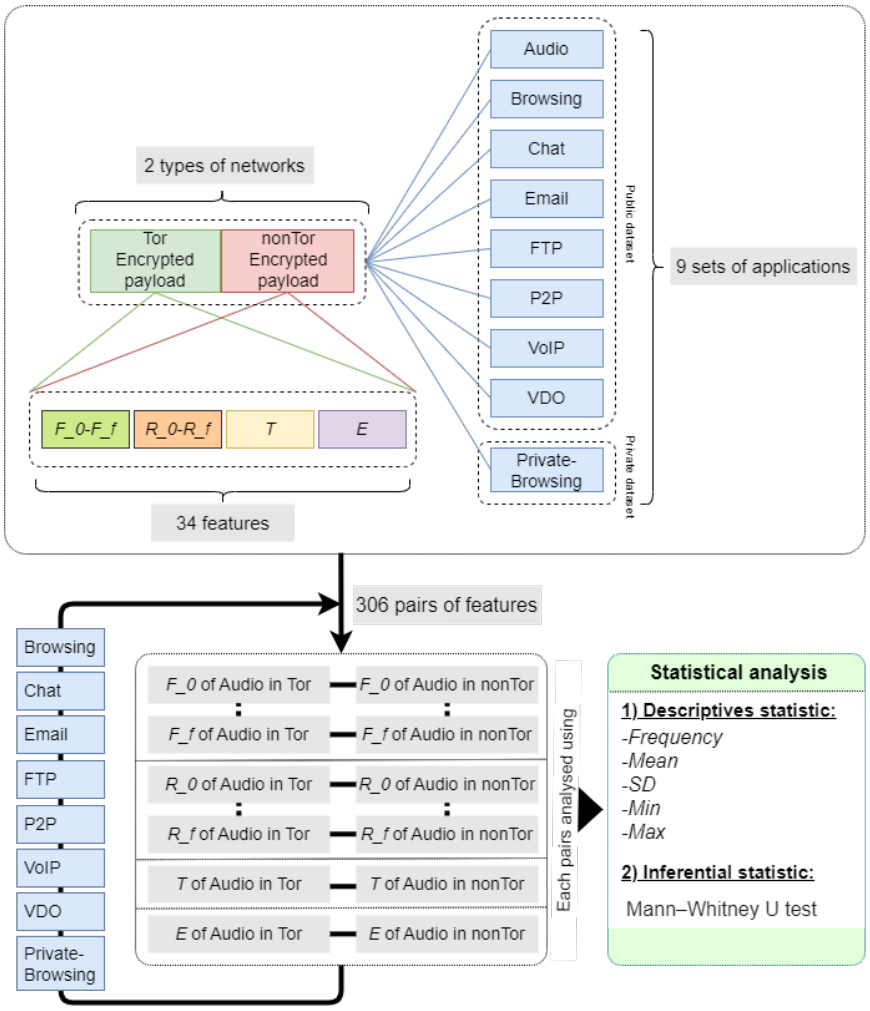


Figure 3.4: Relationship of nine sets of applications on statistical analysis approach

analysis, such as building classification models.

3.4.4 Machine Learning

ML deals with developing algorithms that allow machines to learn patterns and relationships from data. In the context of this study, ML is used to classify Tor traffic and test the second research hypothesis. The entire classification process is depicted in the fourth phase of Figure 3.2. The ML process consists of two phases: the classification phase and the unseen data prediction phase. Before both phases can be performed, the preprocessed dataset must be split into two portions: a training dataset for building

the classification model and a validation dataset for evaluating the performance of the model on unseen data.

In the classification phase, the dataset is divided into a training set, which is used to fit the chosen classification model, and a testing set, which is used to assess the model's performance on previously unseen data. It is important to have a proper dividing ratio to ensure there is enough data to train and test the model effectively. In this study, we employ a supervised learning approach using three algorithms: DTs, RFs, and kNN. These algorithms are selected based on their effectiveness in classification tasks, as discussed in Section 2.6.5. To improve the performance of the classifier and reduce training time, feature selection is performed to select the best and most promising features from Tor and nonTor traffic. The classification results are measured using metrics including accuracy, precision, recall, F1 score and ROC/AUC score.

In the unseen data prediction phase, the finalised model is employed to make predictions on a validation dataset that has not been utilised for training or testing, to prevent overfitting. Overfitting is a prevalent issue in ML models, is unable to generalise or fit accurately to new and unseen data (Ying, 2019). Mitigating overfitting by evaluating the model's performance on previously unused data is essential, as unlabelled data typically serve as the input for real-world implementation in the prediction task. A detailed overview of every step of the ML process will be provided in Chapter 5. In this study, the classification results are analysed using two platforms, Waikato Environment for Knowledge Analysis (WEKA) and Scikit-learn, which are detailed in the next section.

At this point, the reader should have a clear understanding of the methodology employed in the present study, which has been detailed in the preceding section. In the next section, we will present the various research tools utilised to support the implementation of the study.

3.4.5 Research Tools

The successful completion of a research project necessitates the employment of various tools and methodologies for tasks such as data collection, data processing, data analysis, and experimentation. Our work similarly relies on an assortment of software to facilitate

these tasks. The instruments employed in this research are outlined as follows:

The Amnesic Incognito Live System (Tails)

The Amnesic Incognito Live System (Tails)[7] is an open-source, live operating system (OS) specifically designed to offer comprehensive anonymity to its users (Tails, 2022). Tails operates by booting directly from a portable storage device, such as a flash drive or DVD, thereby leaving no trace on the local system it is executed on and eliminating user data upon system shutdown (Cardenas-Haro & Dawson, 2017). This OS is built on the Debian Linux distribution and is compatible with the majority of Unix commands.

In the context of this study, Tails 5.1 is employed for generating a private dataset and reporting its results in conjunction with the public dataset. Tails will be installed as a live OS within a virtual machine environment to ensure that unrelated data from nonTor background network activity is excluded in the analysis.

Wireshark

Wireshark[8] is the most widely employed network traffic analyser utilised by a diverse range of users, such as system administrators and security professionals for network troubleshooting, software developers for investigating communication protocols, and academics for educational purposes. This free, open-source software is renowned for its user-friendly graphical interface and extensive capabilities in capturing and filtering data.

Wireshark captures local network traffic in raw binary data format from any network adapter, encompassing Ethernet, Bluetooth, and wireless connections, enabling both real-time and offline analyses. The captured binary data is typically not in a human-readable format, but Wireshark provides a feature to convert the captured data into a more human-readable format for analysis purposes (e.g., hex and ASCII). which is displayed hierarchically in the breakdown sections of the network communication model (TCP/IP or OSI). Wireshark facilitates traffic filtering during packet capture and offline analysis, refining searches within the network trace. For instance, one can configure a

[7] https://tails.boum.org/
[8] https://www.wireshark.org/

filter to display only TCP traffic from a local network (Sanders, 2017). As mentioned in Section 3.4, Wireshark 3.2.1 is employed in this study to generate a private dataset by capturing Tor and nonTor traffic and serves as a packet analyser tool in the network packet examination step for both datasets.

CHARSTAT

CHARSTAT is a lightweight, custom Python script that runs in Jupyter Notebook[9], which is one of the best Interactive Development Environments (IDEs) for Python and Data Science. CHARSTAT is open-source software that anyone interested in detecting Tor traffic using character statistics on payload-based methods is available to download[10]. It serves as an important tool for the data preprocessing phase of our Tor traffic classification approach, and is inspired by a function in the Posit text profiling toolset, called *charcount* (Weir, 2007, 2009; Weir et al., 2018). The Posit toolset is designed to analyse textual data and count the occurrences of words and characters within a text, which served as a starting point for our approach to feature extraction. CHARSTAT is designed to allow multiple tasks in data preprocessing to be completed in a single execution through two main functions. The first function is *packet filtering and payload extraction*, which removes irrelevant packets and extracts only those containing the payload. The second function is *feature extraction and data labelling*, which involves extracting character statistics-based features and labelling the data. Its user-friendly interface enables CHARSTAT to run the two aforementioned functions in one click on the loaded file after loading input, which is the raw network traffic data. The output is exported in CSV format, allowing for statistical analysis and compatibility with classification employing WEKA and Scikit-learn. The main CHARSTAT procedures involve filtering relevant packets and extracting encrypted payloads, as described in Algorithm 1, and extracting features and labelling the data, as described in Algorithm 2. These procedures are then converted into Python scripts, as presented in Appendix A.1.

[9]https://jupyter.org/

[10]https://github.com/pitpimon/Tor-traffic-classification/blob/main/charstat.py

SPSS

SPSS[11] is a statistical software package for interactive and batch statistical analysis of simple to complex data. It debuted in 1968 and was purchased by IBM in 2009 (Field, 2013). The University of Strathclyde provides free access to the licensed software, SPSS, for university staff and students. Although PSPP[12], an open-source analytical tool that performs similar tasks to SPSS, lacks some of the advanced features of SPSS in managing and analysing large-scale data. As a result, SPSS is the preferred tool for the statistical analysis part of this study and plays a vital role in analysing the descriptive and inferential statistics to prove our first null hypothesis. Its Graphical User Interface (GUI) covers a broad range of capabilities for the entire analytical process, greatly simplifying our statistical analysis tasks. In this research, IBM SPSS Statistics version 28.0.0.0 was utilised.

WEKA

WEKA[13] is a powerful data mining, Java-based open-source software platform created by a developer team at Waikato University in New Zealand (Holmes et al., 1994). The latest stable version of WEKA is 3.8, and for this research, version 3.8.4 was utilised. Java JDK 8 is required to run the software on a local machine. WEKA contains a collection of supervised and unsupervised learning algorithms for ML tasks, accessible through several graphical user interfaces that enable easy access to the underlying functionality. Moreover, WEKA allows running attribute selection algorithms on the trained data to select the most relevant attributes and provides a visual representation of the relationship between attributes (Hall et al., 2009). The software is available for free download from the developer's website and is easy to use without requiring programming knowledge. Therefore, WEKA is a suitable tool for the preliminary ML experiments in this study, where it is used for data preprocessing, classification, feature selection, and unseen data prediction.

[11] https://www.ibm.com/products/spss-statistics
[12] https://www.gnu.org/software/pspp/
[13] https://www.cs.waikato.ac.nz/ml/weka/

Scikit-learn

The Scikit-learn[14] is a popular open-source ML library for Python that provides a wide range of tools for various ML tasks, such as classification, regression, clustering, and dimensionality reduction. It includes a large number of powerful algorithms and techniques for data preprocessing, model training, and evaluation, making it a versatile and valuable tool for many different types of analyses. In classification problems, Scikit-learn includes a wide range of popular classifiers, such as DTs, SVM, and RFs, as well as tools for feature selection and feature engineering (Géron, 2022). This study utilises Scikit-learn in addition to WEKA for running the ML experiments. Comparing the results of the same ML task in both platforms can help to validate the accuracy of the models generated. If both platforms produce similar results, it provides more confidence in the accuracy of the models.

Jupyter Notebook

The Jupyter Notebook[15], formerly known as the IPython Notebook, is an open-source web-based computing platform and is free to use. Jupyter Notebook implements Read-Eval-Print-Loop (REPL) for creating and sharing documents that contain programming codes, visualisations and text interactively on the terminal. It is made up of rows of cells, each of which contains static content such as text, images, graphs, video, audio or the Python programming language, as well as support for over 40 different programming languages (Nagar, 2018; Perkel, 2018). The Notebook's REPL functionality enables users to execute individual code snippets contained in each cell, whereas entire source code files are typically interpreted using traditional programming languages.

In this research, we utilised Jupyter Notebook server version 6.0.3, which was running on Python version 3.7.7 to create and run Python scripts for developing CHARSTAT in data preprocessing and applying the Scikit-learn library for classification tasks. Python-based approaches are highly adaptable, as the code can be easily customised to meet our specific needs. For instance, CHARSTAT is designed to run multiple tasks in

[14]https://scikit-learn.org/
[15]https://jupyter.org/

data preprocessing with just one click upon input loading. Similarly, when it comes to classification, this custom script can efficiently execute multiple tasks in classification with a single click using Scikit-learn. This facilitates testing different classifiers with four distinct feature sets or comparing results across various parameter settings without manually running individual tests on WEKA. Although using Python-based scripts simplifies the ML tasks, it requires some programming skills to fully interpret and leverage the results. In contrast, WEKA is an excellent option for quickly viewing preliminary results and conducting early analyses of the dataset with just a few clicks, without requiring coding experience. However, WEKA may be less flexible when it comes to customisation or modifying the algorithms. Overall, both platforms have their unique advantages and disadvantages, and the choice of which to use depends on the specific needs of each analysis.

3.4.6 Validity and Reliability

This study aimed to investigate and compare the characteristics of encrypted payloads in Tor and nonTor traffic using character analysis and ML techniques. Our research questions centred on the efficacy of statistical and ML analyses in distinguishing Tor and nonTor traffic. To guarantee the validity and reliability of our study, several key steps were taken throughout the research process to maintain and enhance the credibility of our findings.

To ensure the validity of our study, we commenced with a comprehensive literature review that identified gaps in both the existing knowledge and previous works related to the classification of Tor and nonTor encrypted payloads. This review process facilitated our understanding of the distinctions between Tor and nonTor encrypted payloads, enabling us to develop clear research questions and devise a novel approach to address the identified gaps. Our research questions defined the formulation of our methodology, which centred on character-based analysis. By examining individual hex characters in ciphertexts, we employed a suitable approach for the study of encrypted payloads. The features extracted—such as frequency hex, frequency ratio, total characters, and entropy—proved relevant for the classification of Tor and nonTor encrypted payloads,

thereby ensuring the appropriateness of our approach within the context of this study. Quantitative data played a pivotal role in substantiating our chosen methods. We utilised statistical analysis and ML techniques to conduct a comprehensive examination of the encrypted payloads. By applying the quantitative data derived from the extracted features to these approaches, we achieved unbiased results that facilitated the precise classification of Tor and nonTor traffic. The careful selection and implementation of appropriate methods, coupled with the leverage of quantitative data, strengthened the overall quality of our research and its potential contribution to the field.

In terms of reliability, our study employed comprehensive data collection procedures, incorporating quality control measures and steps to ensure consistent data collection across both Tor and nonTor traffic. For instance, we obtained the public dataset from a reputable source, while capturing the private dataset in a controlled network environment. The data collection and preprocessing steps adhered to standardised protocols, such as data cleaning, handling missing data, and transforming variables, ensuring that our statistical and machine-learning analyses were performed accurately. Specifically, classification experiments were conducted using both WEKA and Scikit-learn platforms. The utilisation of two platforms for the classification experiments contributed to the consistency of the results, confirming that observed patterns were not due to chance or errors. Moreover, the adoption of standardised software and tools facilitated the maintenance of our findings' reliability. Lastly, clear and detailed explanations for all steps involved in conducting the research were provided, which is important for enabling other researchers to reproduce our study and achieve comparable results.

Drawing on the foundation provided in previous sections, the purpose of this study is to introduce a novel quantitative method for investigating how the characteristics of encrypted payloads can facilitate network traffic classification. To address our research questions through hypothesis testing, we provided a detailed discussion of the techniques and instruments used, demonstrating the appropriateness of our chosen method for the research. The previous discussion has given readers a clear understanding of the methodology used in this study. The next section will detail the implementation of the proposed research methodology.

3.5 Experimental Setup

The research methodology detailed how to conduct the study in a structured manner, ensuring the reliability and validity of the results. Once the methodology was established, the next step was to set up the components for the experiment. This involved collecting relevant datasets for the study and processing them properly to remove any errors or inconsistencies. Research tools were also configured appropriately to effectively collect and analyse the data.

3.5.1 Dataset Collection

As discussed in Section 3.4.1, we required an additional data source besides the first one acquired from a reliable public site. Consequently, we carefully collected a private dataset through self-collection, ensuring an accurate representation of both Tor and nonTor network traffic. Capturing nonTor traffic was relatively easy since it simply involved capturing network traffic originating from a system without any installed Tor clients. However, capturing Tor traffic needed to ensure that nonTor traffic was involved, as certain protocols transmit and receive packets in the background without user intervention, causing potential interference. To avoid this issue, we collected both network traffic types using a live OS. We installed the live OS on a VirtualBox[16] virtual machine, which served as a clean environment. Data collection for both networks followed the same procedure, with the except for the OS employed. The data-gathering process was conducted on a machine equipped with an Intel Core i5-6000 @ 3.20GHz processor and 16.0 GB of DDR3 RAM. Figure 3.5 provides a visual representation of the Tor and nonTor traffic collection.

We used Tails 4.22 as the OS to emulate the Tor environment, as outlined in Section 3.4.5. Tails is developed by the Tor project and includes an integrated Tor browser, providing maximum privacy and anonymity by leaving no trace on the local system. All incoming and outgoing traffic from Tails is forced to pass through the Tor network, making it classified as Tor traffic. To capture Tor traffic from Tails, we

[16]https://www.virtualbox.org/

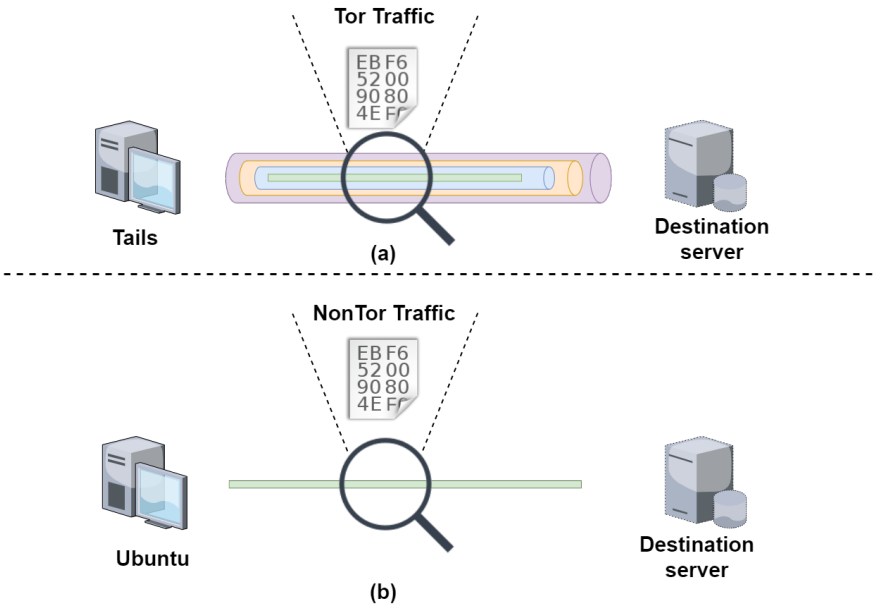


Figure 3.5: Collection of Private Dataset for (a) Tor Traffic and (b) nonTor Traffic

installed Tails on VirtualBox and then installed Wireshark on Tails. We generated browsing traffic by compiling a list of the top 50 domains from the 500 most popular websites ranked by MOZ[17]. We filtered out HTTP domains and similar domains, such as google.com, play.google.com, and others, to select the top 50 domains as input for a Python-based automated browsing script. Appendix A.2 contains the automated Python script and the website list. The traces from Tails were captured using Wireshark, saved as PCAP files, and labelled as 'private-browsing-Tor' traffic. Collecting Tor network traffic proved more challenging than we had expected. We encountered two issues: installing Wireshark in live OSs like Tails required administrative privileges and was wiped upon system reboot. Additionally, some websites, such as https://www.reuters.com, https://www.samsung.com, and https://www.nasa.gov, were inaccessible via the Tor browser and had to be removed from the website lists.

The process of collecting nonTor traffic was similar to capturing Tor traffic, but using different tools used. Firstly, Ubuntu 20.04.3 LTS was utilised as the live OS instead of Tails OS, and Mozilla Firefox was used as the web browser instead of the Tor browser. However, the same automated Python script as described in Appendix A.2 was used

[17]https://www.moz.com/top500 retrieved on 26 August 2021

for both Tor and nonTor traffic capture. Similar to Tails, Wireshark was installed in Ubuntu to capture the network traffic data, and the resulting traces were saved as PCAP files. The captured traffic was then labelled as 'private-browsing-nonTor' traffic.

3.5.2 Data Preprocessing

The process of data preprocessing generally involves several steps, and in the context of this study, it also requires the manipulation of multiple files. To simplify this process, we utilised CHARSTAT.

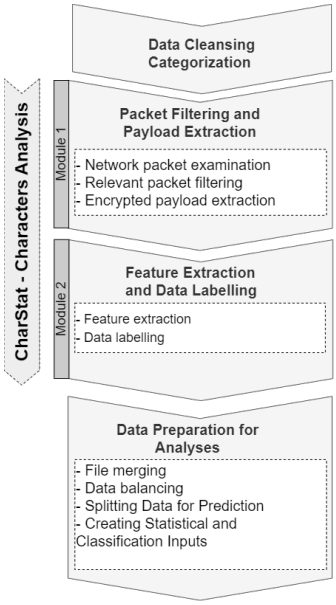


Figure 3.6: Data preprocessing steps

In the data preprocessing phase, the first step was to manually categorise the variety of public dataset into individual application type. After this initial step, we performed the remaining data preprocessing steps on both the public and private datasets using the CHARSTAT tool. The CHARSTAT tool comprises two primary modules: *Module 1*, which involves packet filtering and payload extraction, and *Module 2*, which includes feature extraction and data labelling. Each module consists of several sub-steps. In the next sections, each module will be discussed in detail.

Data Cleansing and Categorisation

The first step of data preprocessing was applied only to the public dataset, with the aim of cleaning the data and categorising the multiple files into their corresponding groups, as discussed in Section 3.4.2. However, this step could not be supported by CHARSTAT because it required a manual examination of the ISCXTor2016 dataset. During the investigation, it was discovered that some files had identical sizes and content but different names, resulting in duplicate files that were removed. Additionally, to ensure an unbiased outcome, unencrypted data generated by insecure protocols such as FTP and HTTP were ignored. Consequently, the total number of PCAP files was reduced from 85 to 68, and these files were categorised into their respective groups.

The categorisation of the data was based on our observations as the file structure was not explained in the literature. Our aim was to ensure that the data were correctly categorised into their corresponding groups. As discussed in Section 3.4.1, the data were collected by recording software activities on both Tor and nonTor networks, resulting in files appearing twice: once in the Tor directory and once in the nonTor directory. We found that the applications' names were based on their primary function. For example, the traffic generated by Spotify, which provides a continuous stream of audio data, was classified as 'Audio', and Vuze's traffic, which is a BitTorrent client used for file transfers via the BitTorrent protocol, was classified as 'P2P'. Some software applications provide more than one primary service, such as Facebook and Skype, which offer services for both Chat and VoIP. The dataset owner included the service name within the file, allowing us to categorise the files correctly. For instance, the file `CHAT_gate_facebook_chat.pcap` was identified as traffic captured from Facebook and categorised as a chat application.

After categorising the files, we investigated the encryption protocols used by each software in order to include relevant packets by applying the accurate filtering command. The following presents the encryption protocols used by each software, organised into application types and provided as a reference to facilitate network packet examination in the next section. However, this list is only applicable to nonTor traffic because when any software operates within the Tor network, its traffic is encrypted using TLS, resulting in all traffic generated from Tor being classified as belonging to the TLS protocol.

Below is a listing of the encryption protocols utilised by each software, categorised by application type:

Audio: Spotify[18] employs a proprietary protocol (Hjelmvik & John, 2009; B. Zhang et al., 2013) to protect data in transit, using a symmetric key stream cipher and enhancing data encryption with the AES block cipher (Wood & Uzun, 2014).

Browsing: When collecting browsing traffic via Chrome[19] and Firefox[20], only HTTPS is considered.

Chat: Various chat applications use different protocols to generate chat traffic, which can make packet examination challenging. Thus, we need to ensure that each application employ secure protocols. ICQ[21], AIM[22], and Skype[23] use secure proprietary protocols, while Hangouts[24] and Facebook[25] encrypt their packets using TLS (Azfar et al., 2016; Datta et al., 2015; Tong et al., 2018).

Email: Thunderbird[26] uses TLS to protect emails (Modadugu & Rescorla, 2004).

FTP: SFTP uses SSH to ensure secure communication while transferring files, while Skype employs its protocol that adds security to file transfers through the use of a strong encryption method (Cheng et al., 2013).

P2P: Vuze[27] employs the BitTorrent protocol and improves security and privacy with MSE/PE (Hjelmvik & John, 2010).

VDO: Vimeo[28] and YouTube[29] encrypt their traffic and use Chrome and Firefox to stream videos continuously. The traffic of TLS can be captured due to secure communication.

VoIP: While Skype uses a secure proprietary protocol (Adami et al., 2009; Mamun et al., 2015; Velan et al., 2015), Hangouts also uses its proprietary protocol to ensure

[18] https://open.spotify.com/
[19] https://www.google.com/intl/th/chrome/
[20] https://www.mozilla.org/en-GB/firefox/browsers/
[21] https://icq.com/
[22] https://aim.chat/
[23] https://www.skype.com/en/
[24] https://hangouts.google.com/
[25] https://www.facebook.com/
[26] https://www.thunderbird.net/
[27] https://www.vuze.com/
[28] https://vimeo.com/
[29] https://www.youtube.com/

secure transmission of video, audio, and data (Vápeník et al., 2014).

Table 3.1: Number of files in Tor and nonTor directories for eight application types obtained from the ISCXTor2016 dataset.

Application Types	Source Applications	# Tor Files	# NonTor Files
Audio	Spotify	2	3
Browsing	Chrome, Firefox	5	7
Chat	ICQ, AIM, Skype, Facebook, Hangouts	6	10
Email	SMTPS, POP3S , IMAPS	4	4
FTP	Skype, SFTP, FTPS	3	2
P2P	Bittorent, Vuze	2	2
Video	Vimeo, Youtube	3	3
VoIP	Facebook, Skype, Hangouts	6	6
Total		31	37

*Noted that the ten repeated items in the Tor and nonTor folders were excluded and uncounted.

Packet Filtering and Payload Extraction

After data categorisation, the next step involved applying packet filtering to extract the encrypted payload from network traces. To ensure the accuracy of the analysis input, a thorough examination of network packets was required to correctly filter out any irrelevant packets. Only packets containing payloads of secure protocols were included to prevent bias in the data. In this study, Wireshark was used to facilitate the examination of network packets. Wireshark allowed us to display all captured packets in the 68 files and provided a comprehensive view of the network traffic. Figure 3.7 depicts how Wireshark was utilised to display all captured packets. TLS packets with payloads were outlined in red borders and were not excluded, while irrelevant packets such as TCP and TLS initial handshake packets were removed to focus only on the relevant data. This approach ensured that only relevant packets were obtained for accurate analyses in both

statistical and ML methods.

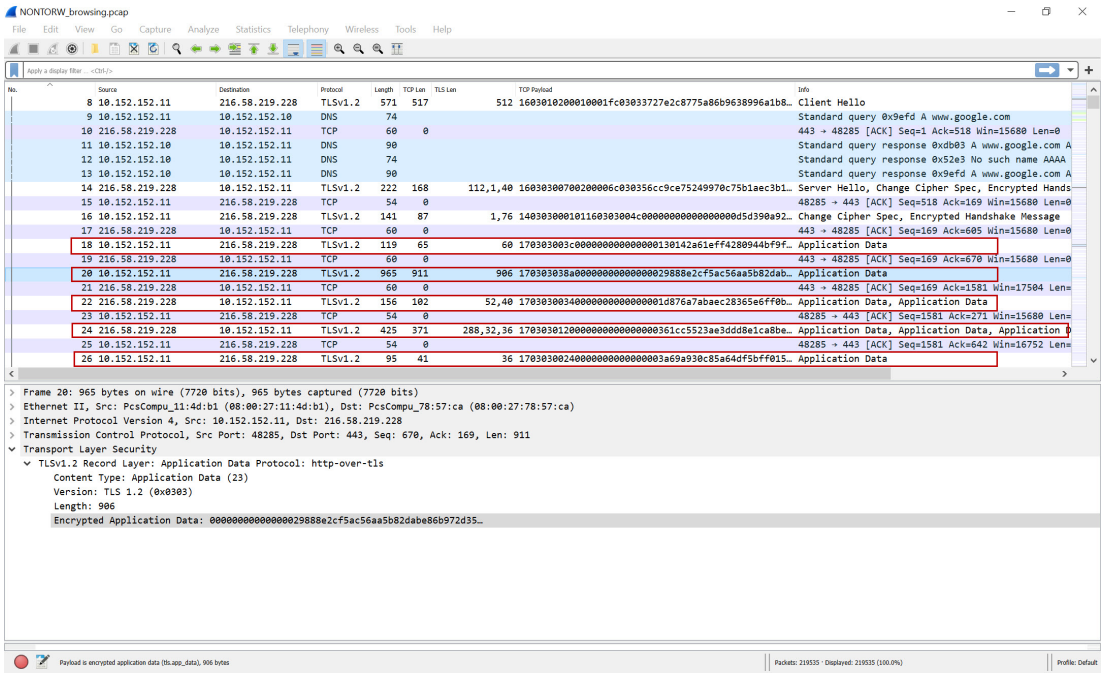


Figure 3.7: Packet examination using Wireshark

The process of extracting encrypted payloads from network traces involved using filtering commands specific to the encryption protocols used by each software application. To identify the appropriate filtering command for each application, we compiled a comprehensive list of the encryption protocols employed by each software, drawing from our previous data analysis. Our findings, for example, revealed that Firefox and Chrome use TLS for secure data transmission, and thus the filtering command for these applications is 'TLS.app data'. However, some applications, such as Skype and Spotify, use proprietary protocols that make it challenging to detect and analyse their traffic. In such cases, we used 'TCP.payload' as the filtering command to extract their payloads. It is worth noting that the captured traces were obtained exclusively from the control environment, as presented in Section 3.4.1. Therefore, it can be reasonably assumed that the packets within the traces predominantly, if not entirely, belong to the software being investigated.

However, after a thorough investigation of the captured traces, it was discovered that there were other packets present that were unrelated to the target applications,

possibly stemming from background processes. For instance, some TLS packets were found in the Spotify traces despite its proprietary protocol. Therefore, it was necessary to exclude such packets to ensure that the extracted payloads were as closely aligned as possible to the relevant packets. This issue required an examination of all network trace files. Consequently, to facilitate extraction of encrypted payloads from both Tor and nonTor traffic, filtering commands were summarised for displaying only encrypted protocols for all applications in Table 3.2. This table was designed to work in conjunction with Algorithm 1 in Section 3.4.2 to extract encrypted payloads. For example, The filter expression `tls.app_data && !tls.handshake && !_ws.expert` that is used to capture only packets belonging to the TLS protocol that contain application data. The `tls.app_data` filter ensures that only packets belonging to the TLS protocol that contain application data are captured, while `!tls.handshake` ignores any handshake packets. Additionally, the `_ws.expert` filter excludes any packets that Wireshark is unable to dissect, such as packets with an unrecognised protocol. This filter expression was applied to capture TLS packets containing application data and exclude handshake packets and any packets that Wireshark cannot dissect.

For audio applications like Spotify, the filter expression `tcp.payload && !http && !tls && !_ws.expert` was used to extract Spotify's encrypted payload. This was because Spotify uses its own proprietary protocol over HTTPS to ensure the privacy and security of its users, and TLS packets were not found in the captured traces. Therefore, using `tcp.payload` as the filter for the encrypted payload was most appropriate. Furthermore, any HTTP packets found were disregarded to ensure that only the relevant data was extracted.

For Vuze, which uses the MSE/PE protocol to encrypt its traffic, a filter expression of `tcp.payload && !bittorrent && !tls && !_ws.expert` was used to extract encrypted payloads belonging only to Vuze, while excluding bittorrent handshake packets used for establishing the session and packets that Wireshark cannot dissect.

Finally, Hangouts uses proprietary protocols for voice and video calls, which can be transmitted over both TCP and UDP. To extract the encrypted payload, we used the filter expression `tcp.payload && !http && !tls && !_ws.expert` since only TCP

packets were found in the network traces. This filter excluded any HTTP packets and TLS packets, which are not relevant to Hangouts' proprietary protocol.

In contrast to nonTor traffic, where filtering commands vary depending on the encryption protocol used, the filtering command for encrypted payloads in Tor traffic is always 'TLS.app data', as Tor traffic always uses the TLS protocol. It should be noted that this filtering method may not be 100% accurate due to the proprietary nature of some network protocols. However, our approach provided a logical rationale for obtaining relevant data by carefully selecting filtering commands based on the specific encryption protocols used by each application.

Table 3.2: Filtering command of encrypted protocols of nonTor traffic

Traffic Type	Application	Encryption Protocol Filtering Command
Public Dataset		
Tor		
All 9-class	N/A	TLS *tls.app_data && !tls.handshake && !_ws.expert*
nonTor		
Audio	Spotify	Proprietary *tcp.payload && !http && !tls && !_ws.expert*
Browsing	Chrome, Firefox	TLS *tls.app_data && !tls.handshake && !_ws.expert*
Chat	ICQ, AIM, Skype	Proprietary *tcp.payload && !tls && !_ws.expert*
	Facebook, Hangouts	TLS *tls.app_data && !tls.handshake && !_ws.expert*
Email	Thunderbird	TLS *tls.app_data && !tls.handshake && !_ws.expert*
FTP	SFTP	SSH *ssh.encrypted_packet && !_ws.expert*
	Skype	Proprietary *tcp.payload && !tls && !_ws.expert*
P2P	Vuze	Bittorent *tcp.payload && !bittorrent && !tls && !_ws.expert*
Video	Vimeo, Youtube	TLS *tls.app_data && !tls.handshake && !_ws.expert*
VoIP	Hangouts, Skype	Proprietary *tcp.payload && !http && !tls && !_ws.expert*
Private Dataset		
Tor & nonTor		
Browsing	Firefox for nonTor Tor browser for Tor	TLS *tls.app_data && !tls.handshake && !_ws.expert*

These filter commands were applied in the `display-filter-specification` window of Wireshark to investigate encrypted payloads and extract them, as shown in Figure 3.8. However, manually filtering and extracting payloads from the large number of files (68 in total) would have been extremely time-consuming. To overcome this challenge, we utilised CHARSTAT, which automated this process and significantly reduced the

time required.

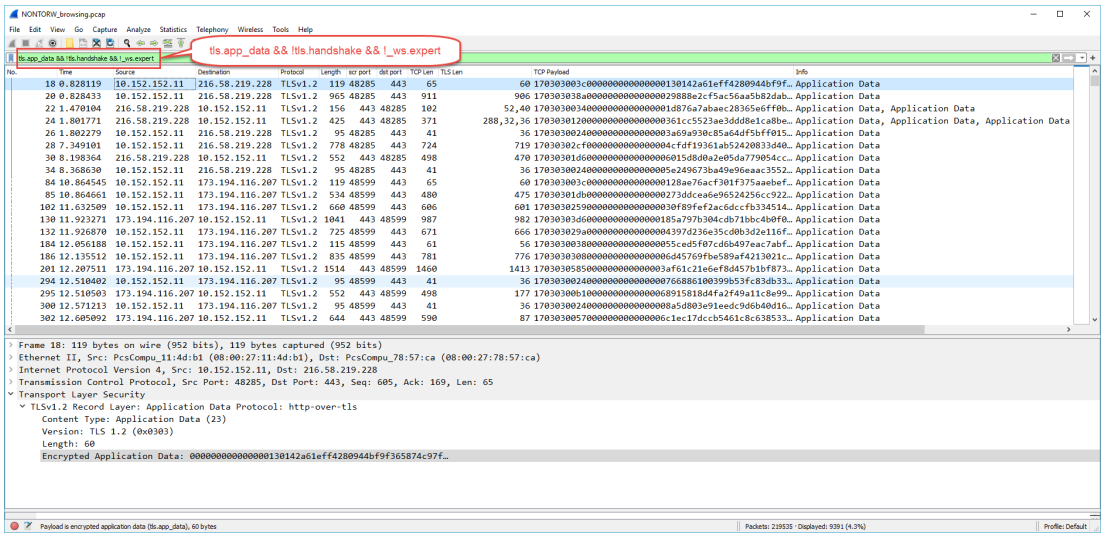


Figure 3.8: Encrypted payloads examination using Wireshark

The results of the encrypted payload extraction using CHARSTAT are illustrated in Figure 3.9. The extracted features were stored in a tab-delimited CSV file containing four columns: ID, source IP, destination IP, and the encrypted payload. The first three columns were included for validation purposes and were eventually removed. The final column, representing the primary feature for analysis, consisted of a string of hex characters. This column was further converted into a format compatible with statistical and ML analyses.

Feature Extraction and Data Labelling

The next step was to extract features and label the data, which was performed in module two of CHARSTAT. To perform feature extraction, the encrypted payload obtained from the previous section was transformed into the proposed features. Algorithm 2, in conjunction with Equations 3.1 through 3.4, illustrates the feature extraction process within CHARSTAT, which enables the automatic computation of 34 features for each individual extracted payload.

After extracting the features using CHARSTAT, the data were labelled based on their corresponding application and network traffic types. This labelling process is

Figure 3.9: Exporting the encrypted payload into CSV format

critical for supervised learning algorithms as it enables the models to learn from labelled examples and make predictions on unseen data. CHARSTAT assigned labels to the data according to their application types and network traffic types. For instance, a network packet containing audio traffic and from the Tor network was labelled as 'Tor-Audio', while a packet containing browsing traffic from a nonTor network was labelled as 'nonTor-Browsing'.

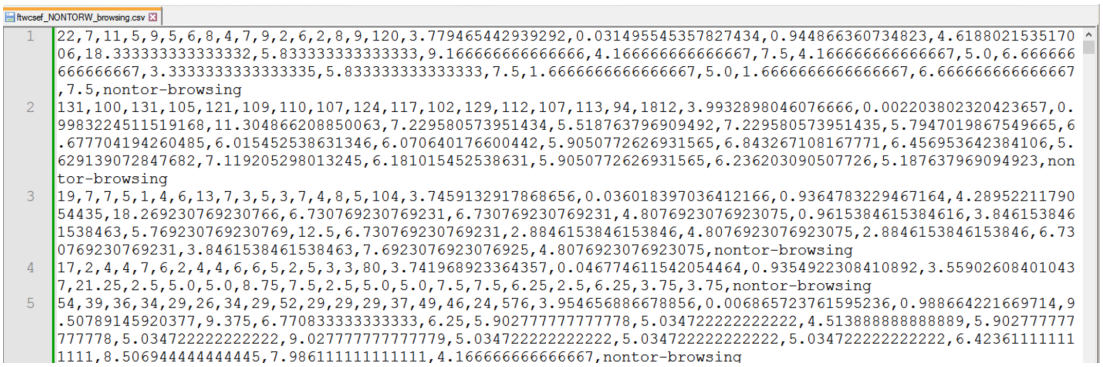


Figure 3.10: Feature extraction and data labelling output in CSV format

The output of the CHARSTAT feature extraction process is presented in Figure 3.10 in CSV format, consisting of 35 columns. the first 16 columns represent F_0-F_f, the second 16 columns denote R_0-R_f, the 33^{rd} column corresponds to T, the 34^{th} column

corresponds to E, and the 35^{th} column represents the label that identifies the class.The features are separated by commas. This process was applied to all 68 files. The next step involves preparing the data for analysis, which will be explained in the following section.

Data Preparation for Analyses

Preparing data for statistical and ML analyses is a manual process that involves four sub-processes: file merging, data balancing, splitting Data for Prediction, and creating Statistical and classification inputs.

File Merging: In the data preparation step, the preprocessed data from the previous step, which were in multiple files, were combined to create binary class and multi-class datasets. For binary classification, the files with their groups from both Tor and nonTor networks were merged into a single file, resulting in nine binary class files, eight from the public dataset and one from the private dataset. For multi-class classification, which was only applied to the public dataset, files with the same application type were merged into a single file, resulting in a single file containing eight classes.

After merging the files, it became apparent that the binary class resulted in imbalanced data. For example, the number of instances of Audio in the Tor class (13,727) was much less than in the nonTor class (148,335), which is a common issue encountered in many classification problems. When one class has significantly fewer instances than the other, some ML algorithms tend to prioritise the majority class and ignore the minority class, resulting in a biased classification model with poor performance. To avoid such issues, it is crucial to balance the data by adjusting the number of instances in each class to ensure that the classification model learns from both classes equally.

Data Balancing: Data balancing is a critical step in data preprocessing, as imbalanced datasets or classes, characterised by an uneven distribution of classes, can pose significant risks during model training. Imbalanced data can negatively impact the learning process, leading to a biased model with poor performance (Ali et al., 2019; Azab et al., 2022). Various techniques can be applied to address imbalanced data, such as oversampling, undersampling, or a combination of both. In our study, we employed

undersampling, which involves reducing the size of the majority class to match that of the minority class. This approach was chosen because the size of the minority class was adequate, and training the model with a larger dataset would require more time and computational resources.

Table 3.3: Number of Tor and nonTor encrypted payloads before and after balancing data for public and private datasets across all application types

Application types	Before undersampling		After undersampling	
	Tor	nonTor	Tor&nonTor (each)	Total
Public Dataset				
Audio	13,727	148,335	13,727	27,454
Browsing	85,840	37,868	37,868	75,736
Chat	3,423	6,066	3,423	6,846
Email	28,559	6,474	6,474	12,948
FTP	271,027	512,339	271,027	542,054
P2P	228,300	1,339,363	228,300	456,600
Video	103,603	16,923	16,923	33,846
VoIP	877,700	388,096	388,096	776,192
Private Dataset				
Browsing	15,579	58,600	15,579	31,158

Table 3.3 provides an overview of the number of Tor and nonTor instances in the public and private datasets of all applications before and after applying undersampling using the `SpreadSubsample` filter in WEKA. The 'before undersampling' columns display the number of instances of Tor and nonTor obtained from the feature extraction step. In contrast, the 'after undersampling' columns present the number of Tor and nonTor instances after undersampling. For example, the before undersampling for the Tor of Audio resulted in 13,727 instances, while the nonTor class had 148,335 instances. The SpreadSubsample filter was then applied to reduce the number of nonTor instances to match that of Tor instances. Similarly, for Browsing, there were 85,840 instances in the Tor class and 37,868 instances in the nonTor class before undersampling. The SpreadSubsample filter was applied to reduce the size of Tor instances to match that of nonTor instances.

Splitting Data for Prediction: As discussed in Section 3.4.4, performing predictions on unseen data is essential, as it helps prevent the overfitting problem. In some cases, such unseen data is provided separately from the training data. However, if no unseen data is available, one approach is to split the existing preprocessed data into training (seen) and validation (unseen) datasets. In this study, a 5% ratio of the balanced data was designated as unseen data, while the remaining 95% was used for the classification phase. This proportion ensures sufficient data for both training and testing. Table 3.4 presents the number of instances in both the seen and unseen datasets after applying the data balancing process.

Table 3.4: Number of instances of seen and unseen data after data balancing

Traffic type	# 95% Seen Data	# 5% Unseen Data	Total
Public Dataset			
Audio	26,082	1,372	27,454
Browsing	71,950	3,786	75,736
Chat	6,504	342	6,846
Email	12,300	648	12,948
FTP	514,952	27,102	542,054
P2P	433,770	22,830	456,600
Video	32,154	1,692	33,846
VoIP	737,382	38,810	776,192
Private Dataset			
Browsing	29,600	1,558	31,158

Creating Statistical and Classification Inputs: This was the last step of data preparation and the final stage of data preprocessing, which involved transforming the preprocessed data for analyses in statistical and ML applications. In this step, the preprocessed data, stored in CSV format, was converted into formats suitable for specific analytical tools. For statistical analysis using SPSS, the preprocessed data in CSV format was converted into the SAV format, and the appropriate data types were assigned to variables. This process was accomplished using the functionalities provided by SPSS. On the other hand, for ML analysis, a header that included the data types

corresponding to each column was added to the top of each CSV file. This header was necessary for ML algorithms to correctly interpret and process the data. It was essential to ensure that the header accurately reflected the data types stored in each column, such as numerical or categorical data. This conversion process was carried out using a text editor or Unix command line tool.

The raw data files were subjected to the necessary data preprocessing steps and transformed into appropriately formatted files to facilitate further processing. Prior to conducting the data analyses, it was necessary to adjust the configurations of the tools utilised in this study. A detailed account of the tool configurations will be presented in the next section.

3.6 Tools Configurations

The analysis presented in Table 3.3 showed that the majority of the files were considerably large in size, which created significant challenges during the data preparation and experimentation stages. Large files required extensive system resources, leading to slow processing and an increased likelihood of errors. To overcome these challenges, it was essential to increase the computer's memory and carefully consider the memory configuration of the tools used. In this context, we described how the memory configurations of the SPSS and WEKA platforms were adjusted to prevent memory issues. It also recommended setting the project directory as the startup folder in Jupyter Notebook to work directly with the project directory.

SPSS

SPSS can encounter a memory error when working with large files exceeding 100 MB or approximately 400,000 records. To address this issue, it is advisable to increase the workspace memory from its default value of 6,184 KB. The command `SHOW WORKSPACE` is for displaying the default memory, while `SET WORKSPACE='new memory'` is to set the desired memory. It is recommended to gradually increase the workspace memory as needed until the task can be executed without errors.

WEKA

To address the issue of *Out-of-Memory* warnings that may appear when working with large files (>100 MB) in WEKA, it is necessary to modify the environment variable `Maxheap` in the `RunWeka.ini` file, located in `C:\Program Files\Weka-3-8-4`, from its default value of 1024 MB to the maximum amount of memory available to WEKA. The amount of memory that WEKA can use can be checked by running the command `java weka.core.SystemInfo` at `SimpleCLI` and examining the `memory.max` property. In this study, the maximum amount of memory available to WEKA was 4068 MB.

Jupyter Notebook

To work directly with the project directory, it was necessary to modify the configuration in the `config.py` file to change the Jupyter Notebook's startup folder. This allowed for direct access to the project directory when working in Jupyter Notebook.

In terms of Python libraries, this study utilised several popular ones for data analysis and ML. These included *Scikit-learn* for importing ML algorithms, *NumPy* for working with arrays and conducting scientific computing, and *pandas* for data manipulation and analysis. These libraries are typically included in Anaconda's free distribution, but it is always recommended to ensure that the latest version is installed to take advantage of the latest features and bug fixes.

3.7 Summary

In the first section of this chapter, a novel quantitative methodology was presented to investigate the role of encrypted payload in facilitating network traffic classification, which is grounded in hypothesis testing. The approach was comprehensively outlined to address the research questions and ensure the reader's comprehension of the chosen method's applicability to the study. Validation and reliability were also discussed to ensure that the research was conducted with validity and reliability.

In the second section, a detailed discussion was presented on the processes and tools utilised in the experiment setup, mainly in dataset collection and data preprocessing.

The necessary steps to transform the raw data into the proper format for analyses were outlined, and the data preprocessing steps were comprehensively explained to ensure the reader's comprehension of the data preparation procedures. Furthermore, the configuration of the tools used to ensure that the analyses were performed without any issues was also discussed in the chapter. The steps taken to address memory-related issues in IBM SPSS Statistics and WEKA were detailed, and the use of the Scikit-learn, NumPy, and pandas Python libraries for data analysis was also mentioned. The next chapters will present the findings of the analyses.

Chapter 4

Statistical Analysis of Tor Traffic

This chapter presents a statistical analysis of sample Tor and nonTor traffic across a variety of applications, aimed at comparing the encrypted payloads of both types based on character analysis. The analysis includes both descriptive and inferential statistics to provide insight into the first research question, which asks, "Can we distinguish Tor from nonTor traffic based on their encrypted payload?" This relates to the first hypothesis, which states, "There is no difference between Tor and nonTor traffic in terms of encrypted payloads". The results of the statistical analysis are expected to provide valuable insights into the unique characteristics of Tor traffic and may facilitate the development of more effective methods for detecting and classifying Tor traffic.

4.1 Features Measurement

In Section 2.4, we explored the efficacy of statistical calculations of word occurrences in various text classification scenarios, substantiated by several studies (Falck et al., 2020; Luo, 2021; Singh et al., 2021). However, these methods have not been explored within the context of computer networks.

In cryptography, the production of ciphertext is the result of encryption, which is usually presented as a set of hex characters that should appear random, based on rigorous cryptography principles. A fundamental goal of cryptography is to ensure that a ciphertext leaks no additional information about the underlying plaintext, regardless

of any information an attacker may already possess (Katz & Lindell, 2020). However, in real-world implementations of cryptography in computer networks, several factors can potentially weaken or compromise the security of the encryption, leading to information leakage.

These factors include vulnerabilities in the encryption algorithms or their implementation, weaknesses in the key management system, and the possibility of side-channel attacks that exploit weaknesses in the physical realisation of the system. For instance, a comprehensive survey on SSL/TLS and their vulnerabilities by Satapathy, Livingston, et al. (2016) highlights the various types of cyber attacks that can compromise the security of SSL/TLS protocols, such as the exploitation of symmetric encryption and cipher block chaining in TLS 1.0, and the use of RC4-generated keys to recover plaintext through statistical analysis of individual ciphertext locations. Although all the vulnerabilities presented in the study have been fixed, this still reflects the need for further research to improve the security of SSL/TLS by reducing bugs.

The unique implementation of Tor's packets and connections, including encryption mechanism and fixed cell size, as discussed in Section 2.3.3, presents an opportunity to exploit these distinctive characteristics for ciphertext analysis. Incorporating character analysis into the ciphertext may provide valuable insights into the security of the Tor network, which is facilitated by statistical analysis. Among the different types of ciphertext representation in the computer networks, we consider hex character representation, consisting of 0-9, a-f in 1-hex form, suitable for analysis (see Section 3.3).

The approach, depicted in Figure 3.4, comprehensively examines all aspects of hex character statistics in Tor and nonTor encrypted payloads using descriptive and inferential statistical analysis across all applications, including Audio, Browsing, Chat, Email, FTP, P2P, VDO, VoIP and Private-Browsing. Four features are considered: (1) the frequency of hex characters (F_0-F_f), (2) the ratio of the frequency of hex characters (R_0-R_f), (3) the total number of characters (T), and (4) entropy (E). This thorough investigation of character statistics aims to reveal the distinct characteristics of encrypted payloads in the two networks under study (Tor and encrypted nonTor).

The four sets of features are analysed using descriptive statistics to account for the

features' characteristics and inferential statistics to comprehensively address the research question and hypothesis H_{01}, which assumes that Tor and encrypted nonTor traffic have identical characteristics. Through this statistical analysis, the research hypothesis is tested, and the research question is answered, providing insights into the effectiveness of the approach in distinguishing between Tor and nonTor traffic.

4.2 Descriptive Statistics

Descriptive statistics are crucial in providing insights about the characteristics of a dataset, as discussed in Section 2.5.2. In our study, we employed descriptive statistics to examine Tor and nonTor encrypted data based on character analysis. The four sets of features were analysed using descriptive statistics, which include measurements such as frequency, Mean, SD, Min and Max values. The frequency measurements are presented in the context of the data distribution to enable effective analysis. Data distribution is used to reveal the pattern of how individual features are distributed across the range of values in Tor and nonTor encrypted payloads. The Mean measurement provides an understanding of the central tendency of each feature, helping to identify the average value of the individual features. SD serves as a measure of the variability or dispersion of each feature, helping to identify trends or patterns in the individual features. Min and Max values provide an understanding of the range of each feature within each Tor and nonTor application, where a wider range may indicate greater variability in traffic characteristics. These measurements, applied to each feature, can be instrumental in identifying differences in data patterns and understanding the distinct characteristics of encrypted payloads in the two networks under study.

The study involved a descriptive statistics analysis, and a comprehensive analysis of the results is provided in the appendices to this thesis. Detailed information about the descriptive statistics analysis results can be found in Appendix B.1 through B.13. Due to space limitations, the findings are presented in a summarised form to offer a concise overview to the reader. To display and effectively compare the results, violin plots were utilised. Violin plots are a useful tool to visualise the distribution of each feature. They represent the probability density of the data at different values, with the width of the

violin at a particular point indicating the estimated probability density at that point. A wider section of the plot indicates a higher concentration of data points, while a narrower section indicates a lower concentration. The median lines within the violin plots represent the middle value of distribution, making them useful for summarising and comparing distributions (Hintze & Nelson, 1998). In the context of our study, the violin plot visually represents the distribution of each feature's values. The width of the plot shows the density of values at different points, while the median line represents the middle value of each feature distribution. The height of the plot represents the frequency or number of features. The y-axis displays the values, while the x-axis labels Tor and nonTor. It is important to note that while the violin plot does not provide the actual values of the data, it still provides a visual representation of the distribution and density, which can be useful for identifying patterns and trends. The following section displays the summary of findings from the descriptive statistics analysis.

4.2.1 Hex Character Frequency (F_0-F_f)

The individual hex character frequency (F_0-F_f) features were analysed using data distribution, Mean, SD, Min, and Max values in Tor and nonTor encrypted payloads across all applications. The significant findings are presented below.

Data Distribution

Figure 4.1 illustrates the violin plots for the data distribution of individual frequency (F_0-F_f) features in encrypted payloads of Tor (left-side violin) and nonTor (right-side violin) across all applications. Based on the violin plots, it can be observed that the data distribution of individual frequency features (F_0-F_f) is similar between Tor and nonTor encrypted payloads across all applications. In Tor, the majority range of individual frequency features is around 100 characters across all applications, indicating a consistent frequency of each hex character frequency appearing in their ciphertext for each application. On the other hand, nonTor has a higher majority of individual frequency features in Audio, FTP, and P2P applications, while the remaining applications have a consistently lower majority compared to Tor. The outliers in nonTor appear

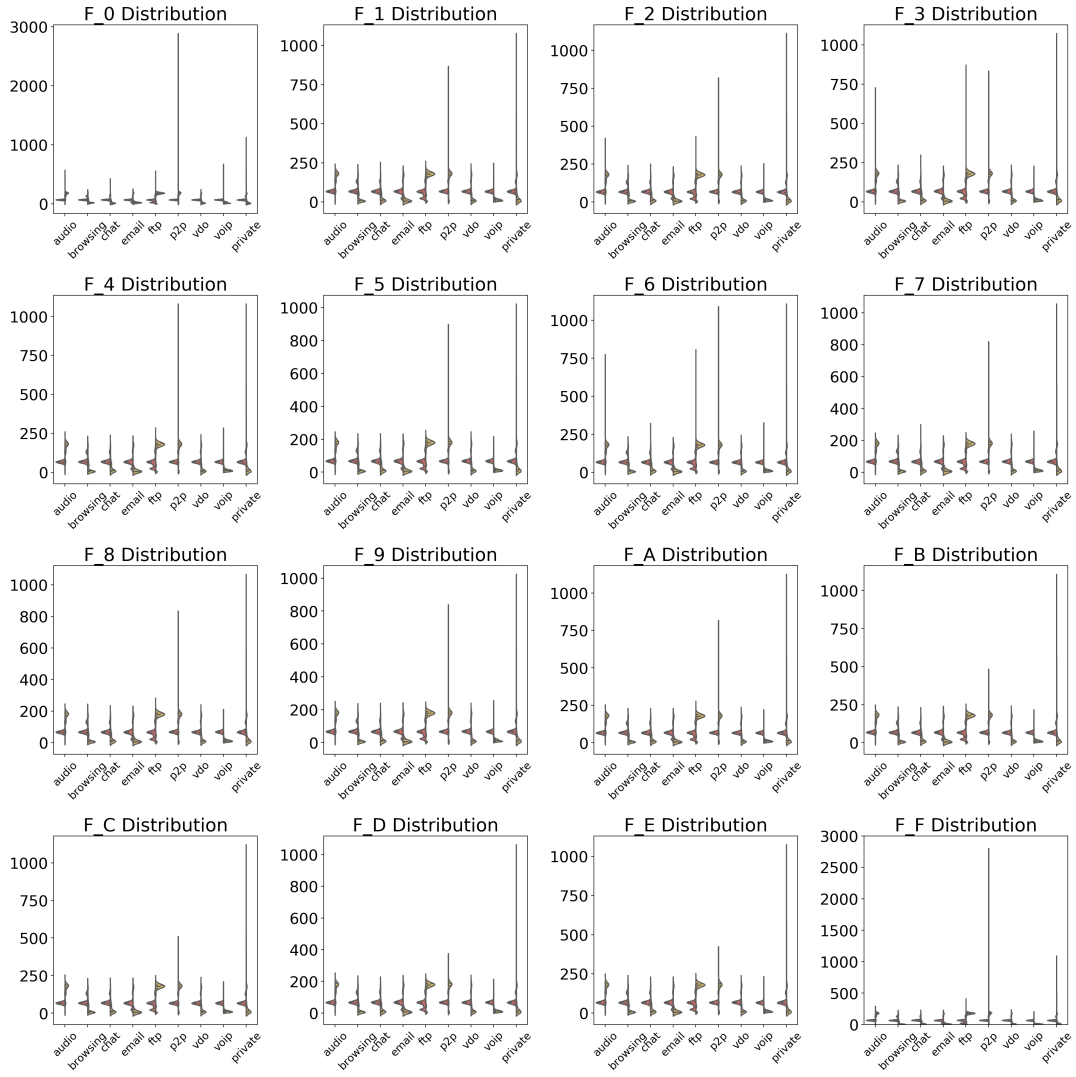


Figure 4.1: Distribution of individual frequency (F_0-F_f) features in encrypted payloads of Tor (left-side violin) and nonTor (right-side violin) across all applications, illustrated with violin plots

in all applications, particularly with significantly higher frequency in P2P and Private Browsing applications.

Mean, SD, Min and Max

Figure 4.2 illustrates the violin plots for the dispersion measures (Mean, SD, Min, and Max) of the individual hex character frequency (F_0-F_f) features in the form of range

values in Tor and nonTor encrypted payload across all applications. The significant findings are discussed below.

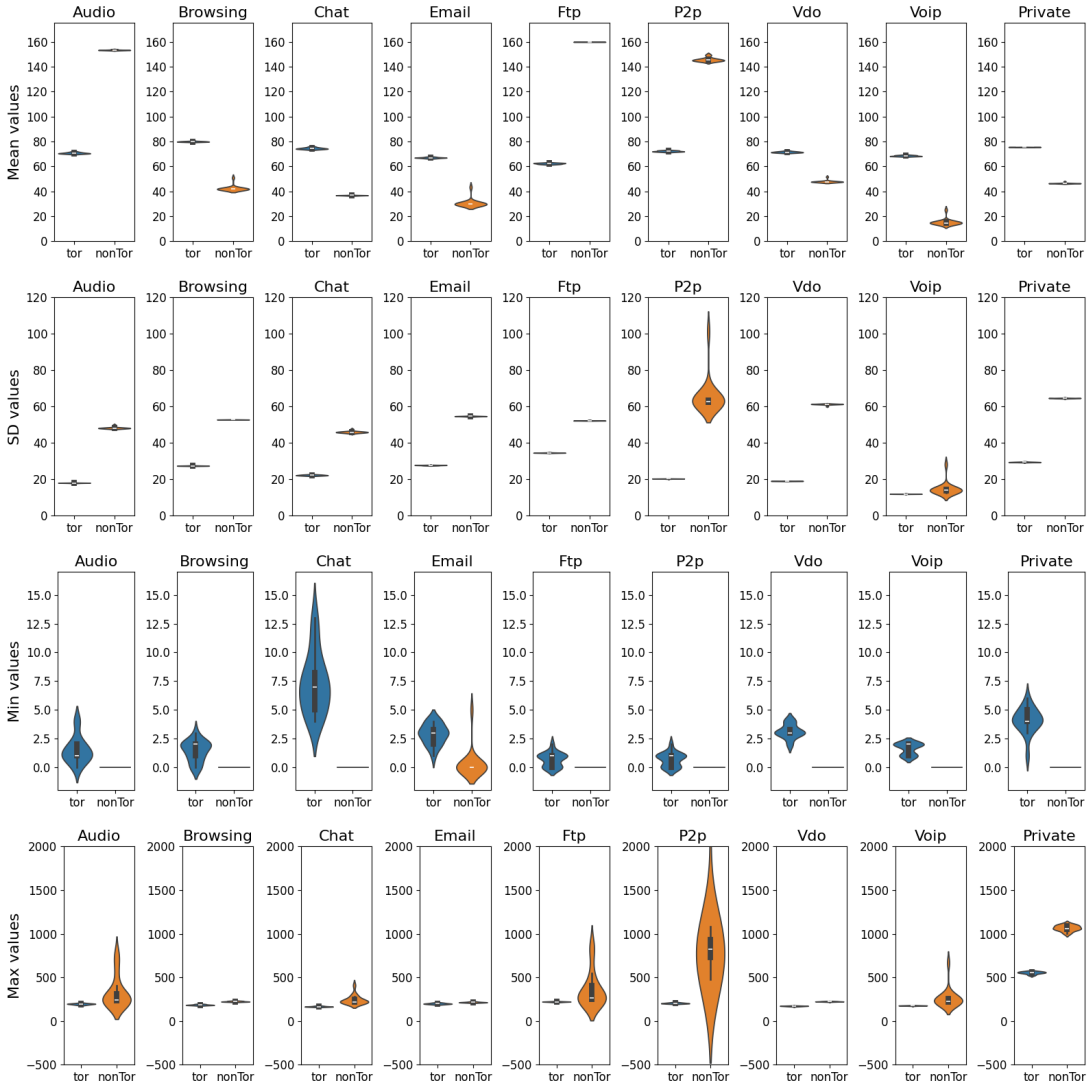


Figure 4.2: The dispersion measurements (Mean, SD, Min and Max) of the individual hex character frequency (F_0-F_f) features in the form of range values in Tor and nonTor encrypted payload across all applications, illustrated with violin plots

Mean: It is observed that the Mean distribution of individual hex character frequency features in nonTor traffic generally covers a higher range than in Tor traffic for most applications, except for Audio, Chat, and FTP. Specifically, the majority Mean distribution of individual hex character frequency features in Tor traffic is consistent

with being concentrated in the range of 60 to 80 characters across all applications, while nonTor traffic is mostly lower than Tor traffic, except for Audio with the majority around 153 characters, FTP around 159 characters, and P2P around 145 characters. Additionally, some nonTor applications exhibit a wider range in the Mean distribution of individual frequency features, with Browsing having the majority of around 42 characters, Email around 30 characters, and VoIP around 13 to 15 characters.

SD: The SD distribution of individual hex character frequency features in nonTor traffic was generally more dispersed than in Tor traffic across all applications. In Tor traffic, the SD ranged around 12 to 35 characters and was concentrated around 18 characters for Audio and VDO, 27 for Browsing and Email, 22 characters for Chat, 34 characters for FTP, 20 characters for P2P, and 12 characters for VoIP. On the other hand, nonTor traffic generally exhibited higher SD distributions than Tor traffic, except VoIP which had a majority of around 11 to 16 characters.

Min and Max: The distribution of Min and max values for individual hex character frequency features revealed a clear contrast between Tor and nonTor traffic. Specifically, Tor displayed higher Min values, with the majority ranging between 0 and 4 characters across all applications, whereas nonTor values predominantly remained at 0 characters. The distribution of Min values showed consistency between Tor and nonTor encrypted payloads, while nonTor Max values were substantially higher than those of Tor. In contrast, Tor had a consistent Max distribution, with the majority ranging between 160 and 233 characters, except for the private dataset, where it was around 550 characters. On the other hand, nonTor applications displayed a broader range of Max values.

4.2.2 Hex Character Frequency Ratio (R_0-R_f)

The individual hex character frequency ratio (R_0-R_f) features were analysed using data distribution, Mean, SD, Min, and Max values in Tor and nonTor encrypted payloads across all applications. The significant findings are discussed below.

Data Distribution

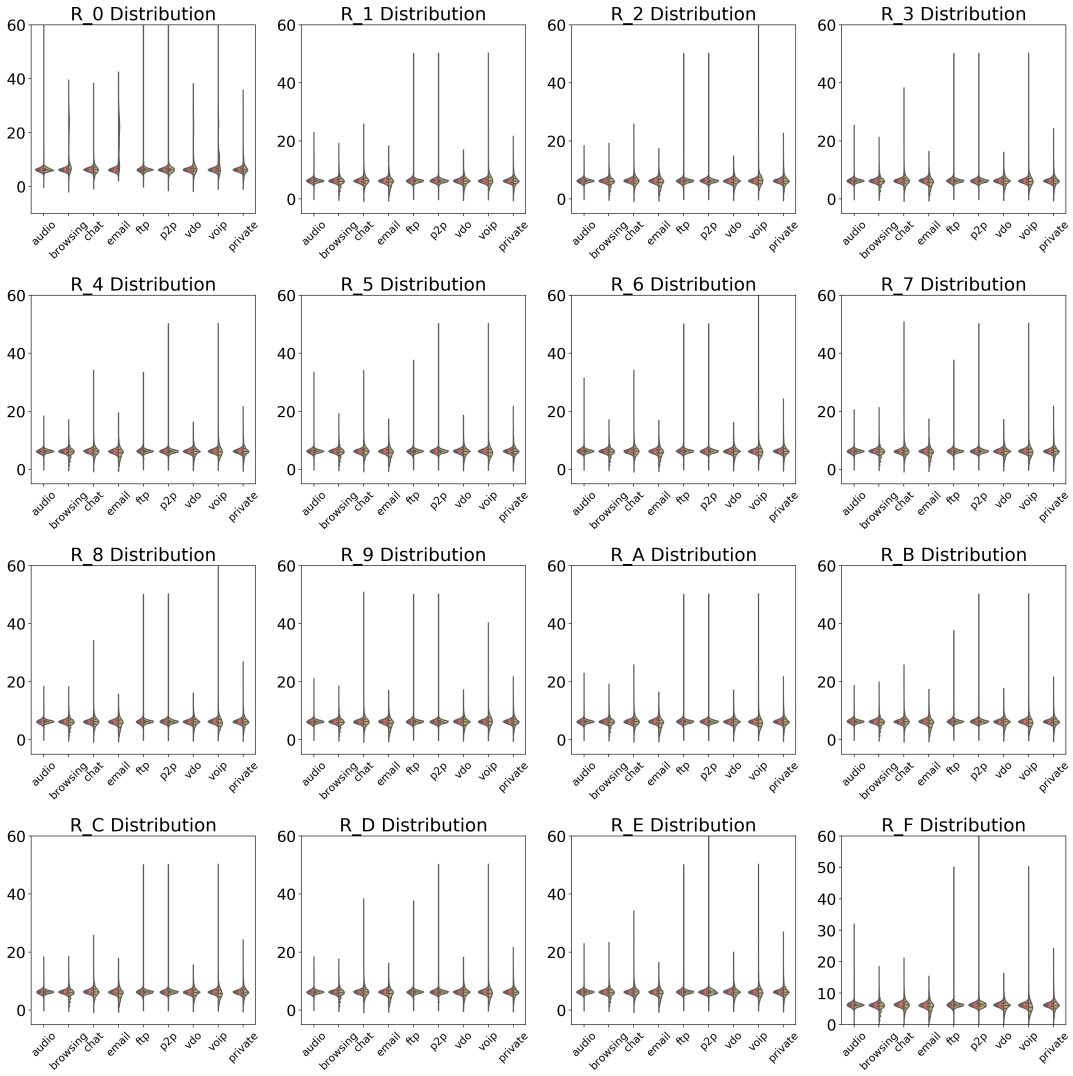


Figure 4.3: Distribution of individual frequency ratio (R_0-R_f) features in encrypted payloads of Tor (left-side violin) and nonTor (right-side violin) across all applications, illustrated with violin plots

Figure 4.3 illustrates the violin plots for the data distribution of individual hex character frequency ratio (R_0-R_f) features in encrypted payloads of Tor (left-side violin) and nonTor (right-side violin) across all applications. The violin plots show that the data distribution of individual frequency ratio features exhibits a consistent pattern between Tor and nonTor encrypted payloads across all applications. However, some nonTor applications, such as Email and VoIP, display a greater range than in Tor. Additionally, outliers are observed in nonTor encrypted payloads across all applications.

Mean, SD, Min and Max

Figure 4.4 illustrates the violin plots for the dispersion measurements (Mean, SD, Min, and Max) of the individual hex character frequency ratio (R_0-R_f) features in the form of range values in Tor and nonTor encrypted payloads across all applications. The significant findings are discussed below.

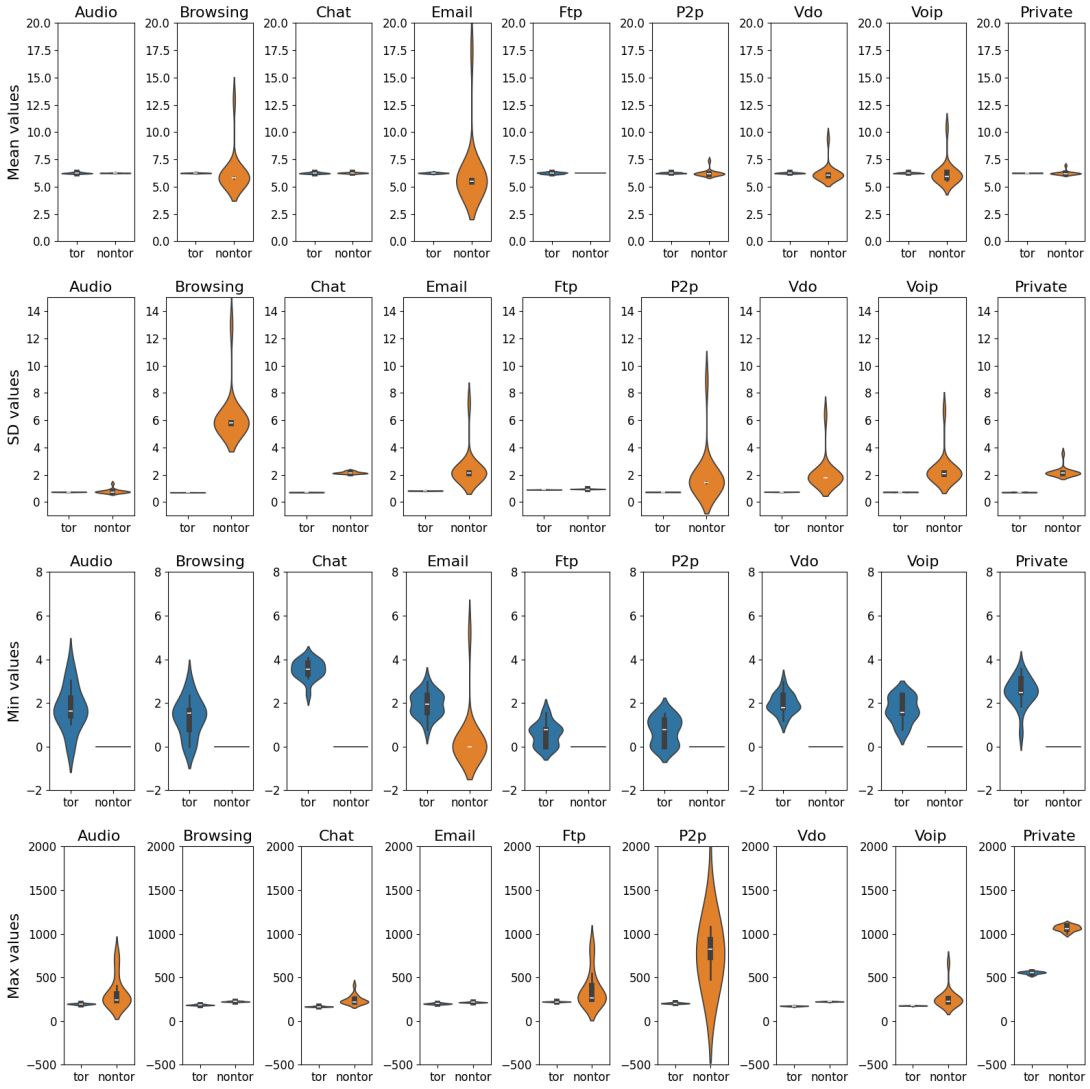


Figure 4.4: The dispersion measurements (Mean, SD, Min and Max) of the individual frequency ratio (R_0-R_f) features in the form of range values in Tor and nonTor encrypted payloads across all applications, illustrated with violin plots

Mean: The analysis showed that the Mean distribution of individual hex character frequency ratio features in Tor had a consistent range between values of 6.1 to 6.5, while nonTor displayed greater diversity values across all applications, except for FTP which exhibited a significant consistent concentration at around 6.25. However, nonTor had a similar concentration to Tor in Audio, with Mean values ranging between 6.2 to 6.3. Chat also exhibited a relatively consistent concentration in both Tor and nonTor,

ranging from values of 6.15 to 6.35.

SD: The SD of the distribution of individual hex character frequency ratio features showed consistency between Tor and nonTor encrypted payloads, with Tor exhibiting a relatively stable range of values between 0.8 and 0.9 across all applications. However, the variability in the distribution of individual hex character frequency ratio features for nonTor was generally higher than that of Tor applications. Browsing had the highest SD range of around 5, indicating a wider variability of this feature for nonTor traffic in this application.

Min and Max: The distribution of the Min and Max values for individual hex character frequency ratio features showed a clear contrast between Tor and nonTor applications. Tor had higher Min values, with a wider range across all applications, including Audio, Browsing, and Private with Min values ranging from 0 to 4, Chat values from 2 to 4, and FTP and P2P values from 0 to 2. On the other hand, nonTor values were mostly concentrated at 0. The Max distribution, on the other hand, showed a consistent pattern between Tor and nonTor encrypted payloads, with nonTor Max value ranges being substantially higher than those of Tor. Tor had a consistently smaller Max distribution value, with the majority ranging around 10, in contrast to a broader range in all nonTor applications. The widest range of Max distribution was found in FTP, P2P, and VDO, with the majority values around 25 to 70, while the narrowest range was found in Browsing, Email, and VDO, with the majority values around 10 to 25.

4.2.3 Total Characters (T)

The total characters (T) feature was analysed for the data distribution, SD, Min, and Max values in Tor and nonTor encrypted payloads across all applications. The significant findings are discussed below.

Data Distribution:

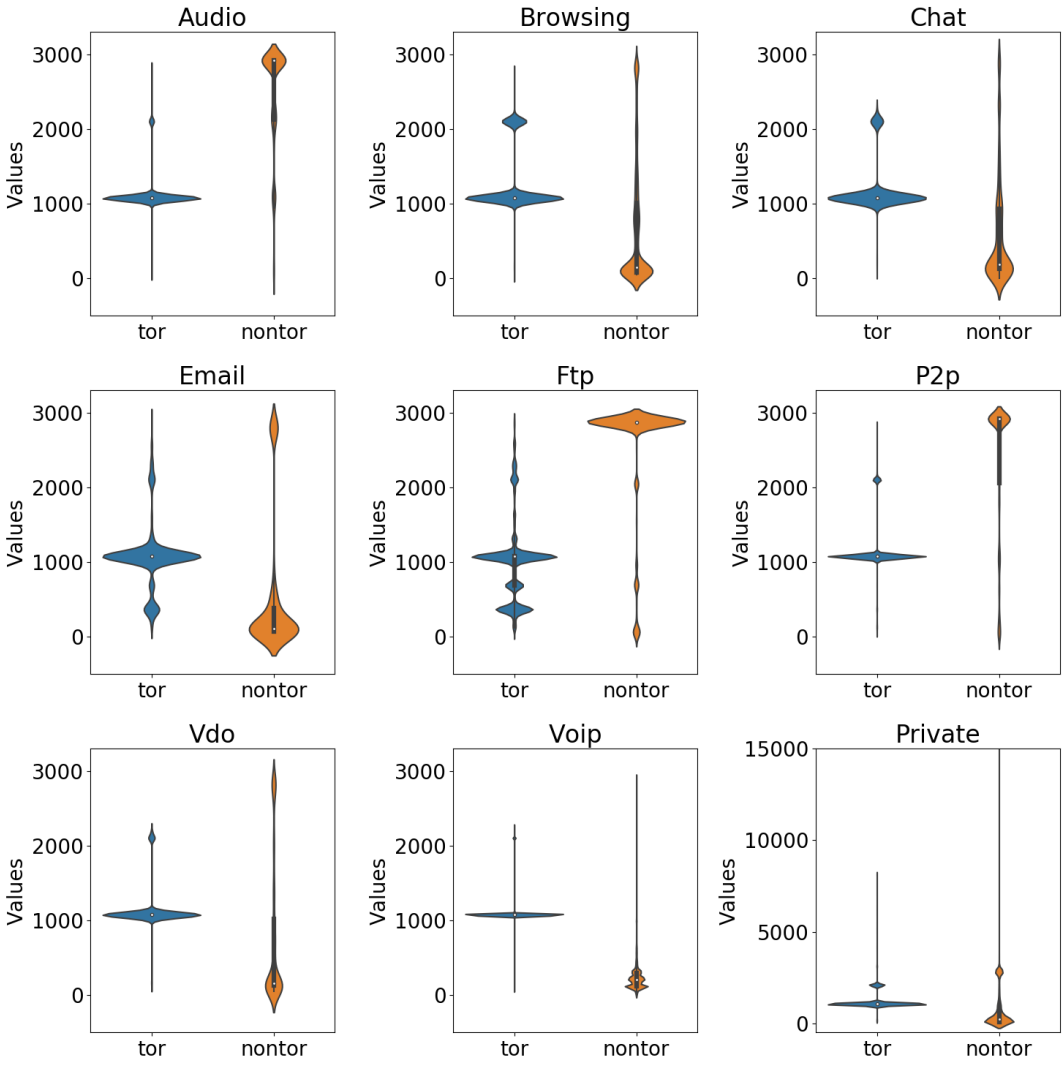


Figure 4.5: Distribution of total character (T) feature in Tor and nonTor encrypted payload across all applications, illustrated with violin plots

Figure 4.5 displayed violin plots for the data distribution of the total characters (T)

feature in Tor and nonTor encrypted payloads across all applications. The violin plots for Tor showed a consistent shape, with the majority of total characters concentrated at around 1,000 characters across all applications. However, there were multiple wider sections in most Tor applications, except for VoIP. For example, Browsing had two denser regions, with the majority of characters around 1,000 and a smaller concentration of around 2,000. FTP exhibited multiple denser regions, with the majority around 1,000, 400, and 600 characters. In contrast, the distribution of total characters in nonTor encrypted payloads was more diverse. The highest majority of total characters were found in Audio, FTP, and P2P applications, with values around 3,000 characters.

Mean, SD, Min and Max

Figure 4.6 illustrates the violin plots for the dispersion measurements of the total character (T) feature, which ranges across all applications in Tor and nonTor encrypted payload, in terms of Mean, SD, Min, and Max. The significant findings are discussed below.

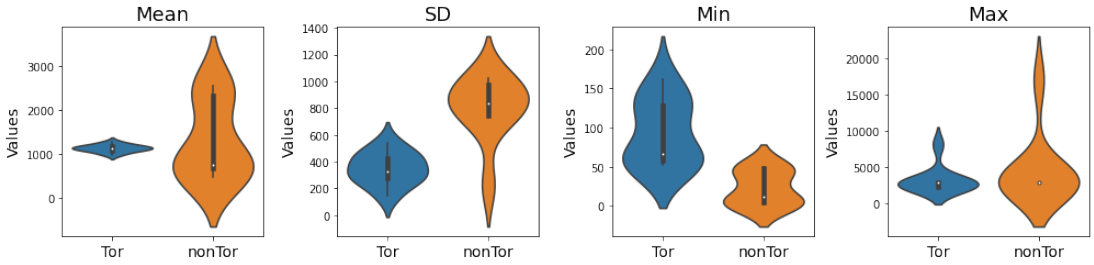


Figure 4.6: The dispersion measurements (Mean, SD, Min and Max) of the total character (T) feature, which ranges across all applications in Tor and nonTor encrypted payload, illustrated with violin plots

Mean: The analysis of Mean values for the total characters feature across all applications indicated that nonTor traffic generally exhibited greater variability in the total number of characters in encrypted payloads compared to Tor traffic. The observed difference ranged from approximately 600 to 2500 characters, with two majority sections being around 600 to 700 characters and around 2300 to 2500 characters, implying that nonTor traffic tends to have a diverse range of total character numbers in encrypted payloads on average.

SD: The SD values of the total characters feature across all applications indicated that nonTor traffic had greater variability in the total characters of encrypted payloads, which were concentrated around 600 to 1,200 characters, compared to Tor traffic, which exhibited wider sections at around 200 to 600 total characters across all applications. This suggests that the total characters of encrypted payloads for nonTor applications are more diverse than the consistency observed in the encrypted payloads of Tor applications.

Min and Max: The analysis of the Min and Max values for the total characters feature in encrypted payloads revealed the range of character counts for both Tor and nonTor applications. NonTor applications generally had a higher minimum total character count compared to Tor applications, with a few exceptions where Tor applications had slightly higher Min values. However, in some nonTor applications, the Min total character count was very low. In contrast to the Min values, both types of applications exhibited similar shapes in their Max values, with the majority of applications having a Max character count of around 3000 characters across all applications.

4.2.4 Entropy (E)

The entropy (E) feature was analysed for the data distribution, SD, Min, and Max values in Tor and nonTor encrypted payloads across all applications. The significant findings are discussed below.

Data Distribution

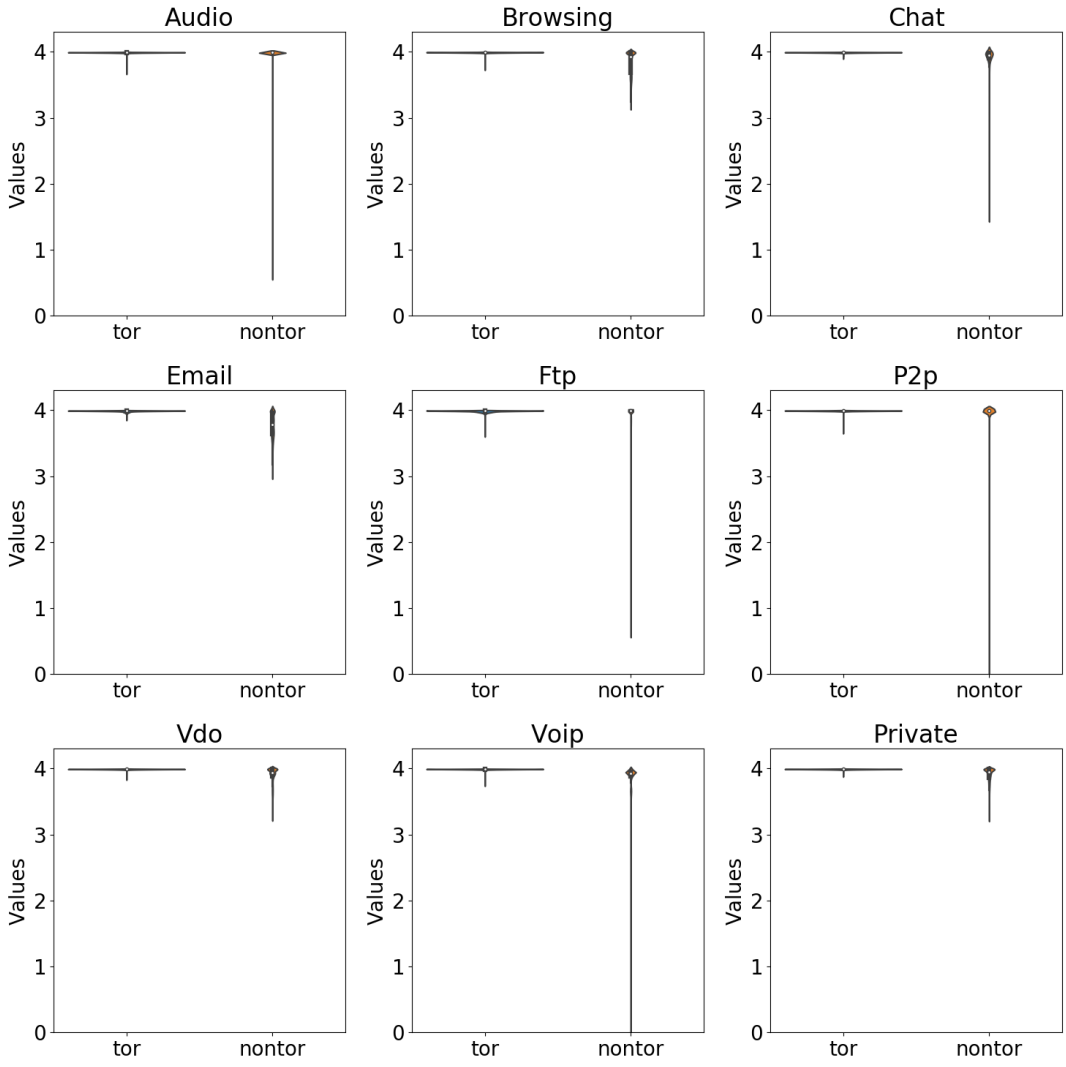


Figure 4.7: Distribution of entropy (E) feature in Tor and nonTor encrypted payload across all applications, illustrated with violin plots

Figure 4.7 displayed violin plots for the data distribution of the entropy (E) feature in

Tor and nonTor encrypted payloads across all applications. The distribution of entropy showed consistency between Tor and nonTor applications, with density parts at around 4 across all applications. However, in nonTor applications, extreme values were observed, with values as low as around 3 in Browsing, Email, VDO and Private, and even lower in the remaining applications. This implies that the minimum total character counts in these applications were considerably lower in nonTor traffic compared to Tor traffic, where the extreme values were much higher, at around 3.8 of the minimum entropy value found in all Tor applications.

Mean, SD, Min and Max

Figure 4.8 illustrates the violin plots for the dispersion measurements of the entropy (E) feature, which ranges across all applications in Tor and nonTor encrypted payload, in terms of Mean, SD, Min, and Max. The significant findings are discussed below.

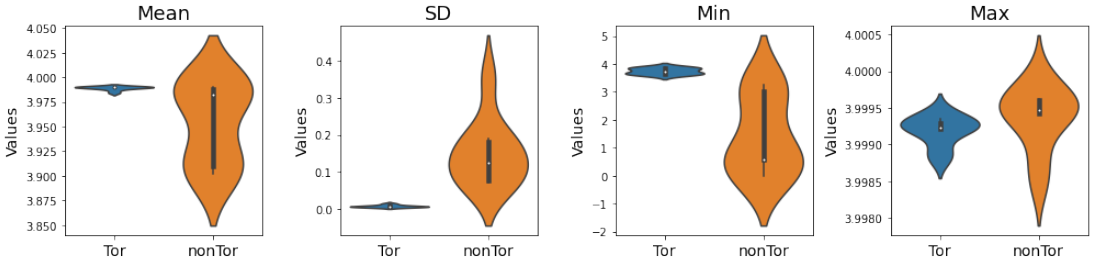


Figure 4.8: The dispersion measurements (Mean, SD, Min and Max) of the entropy (E) feature, which ranges across all applications in Tor and nonTor encrypted payload, illustrated with violin plots

Mean: The analysis of Mean entropy values for Tor and nonTor traffic revealed a close similarity, with Tor traffic displaying slightly higher Mean entropy values in most cases. The majority of Mean entropy values for Tor traffic were consistently close to 4, indicating a relatively high level of randomness in the encrypted payloads. On the other hand, the range of Mean entropy values for nonTor traffic spanned from 3.9 to 4, indicating a moderately consistent level of randomness across different types of nonTor traffic.

SD: The SD values for entropy in nonTor applications showed greater variability, ranging from around 0 to 0.4, compared to Tor applications, where the variability was

almost 0 across all applications. This suggests that nonTor traffic had a more diverse set of encrypted payloads in terms of entropy. In contrast, Tor traffic had a smaller variability of entropy values than nonTor traffic, indicating a tighter and more consistent distribution of entropy values across different applications.

Min and Max: NonTor applications generally exhibited lower Min entropy values in their encrypted payloads compared to Tor applications, but had a wider range of values ranging from around 0 to 4, while Tor applications had a more narrow range of values around 4. Despite this difference, both types of applications had a similar distribution shape for Max entropy values, with the majority of values being around 3.9995. Overall, nonTor applications displayed a wider range of Max entropy values compared to Tor applications.

We have seen how descriptive statistics can reveal whether Tor and nonTor payloads are similar or different in various aspects. However, we need to make a judgement based on a comparison of the two if they belong to the same population. Inferential statistics can be used to draw conclusions for summarising the null hypothesis.

4.3 Inferential Statistics

This section aims to analyse and contrast encrypted payloads in Tor and nonTor traffic, focusing on the attributes of 306 distinct features. As discussed in 2.5.2, inferential statistics are used to test hypotheses and generalise results from a sample to an entire population (Allua & Thompson, 2009). Before conducting the inferential tests, we checked the assumption of normality for the dataset, as this is necessary to determine the appropriate inferential test to use. Normality was assessed using the Kolmogorov-Smirnov test on a total of 306 features, as presented in Appendix B.14. The results indicated that all 306 features had p-values less than 0.05, suggesting that the data do not follow a normal distribution. Additionally, the two groups of data (Tor and nonTor) we analysed are independent of each other. The data under study is discrete in nature, making it rankable. Given these characteristics, the Mann-Whitney U test, an alternative to parametric tests, can be employed to compare the two independent

samples and determine if they originate from the same population when the normality assumption is not satisfied. The results of the Mann-Whitney U test are provided in Appendix B.15. This test was used to determine the p-values of all features from all applications in both public and private datasets using SPSS. The test produces a p-value ranging from 0 to 1. A p-value less than 0.05 indicates statistical significance, implying that there's less than a 5% probability of observing the given data (or something more extreme) if the null hypothesis were true. If the p-value is less than 0.05, we reject the null hypothesis and accept the alternative one. In our context, this indicates distinct populations between the two networks.

Table 4.1: Features with p-values > 0.05 in Tor and nonTor encrypted payloads

Application types	**Features**	***p*-value (Mann-Whitney U)**
Audio (3)	*R_5*	0.077
	R_c	0.066
	R_e	0.803
Chat (10)	*R_1*	0.368
	R_5	0.156
	R_6	0.964
	R_7	0.741
	R_8	0.481
	R_9	0.770
	R_a	0.064
	R_b	0.225
	R_d	0.185
	R_e	0.051
P2P (1)	*R_0*	0.380

The summarised results are presented in Table 4.1. This table highlights features with p-values greater than 0.05, indicating that they did not reach the conventional threshold for significance. This suggests non-significant differences between Tor and nonTor encrypted payloads. From these results, we can infer similarities between Tor

and nonTor encrypted payloads across different application types. Specifically, within the Audio application type, the features *R_ 5*, *R_ c*, and *R_ e* were indistinguishable between Tor and nonTor payloads, resulting in a similarity rate of 8.82% for both Tor and nonTor Audio encrypted payloads. For the Chat application, ten features were identified (*R_ 1, R_ 5, R_ 6, R_ 7, R_ 8, R_ 9, R_ a, R_ b, R_ d, and R_ e*), leading to a similarity rate of 29.41% for both Tor and nonTor Chat encrypted payloads. Conversely, P2P had only one identical feature, *R_ 0*, yielding a resemblance rate of 2.94% for both Tor and nonTor P2P encrypted payloads. The findings indicated that considering all features from all application types, the Mann-Whitney U test findings revealed that out of the 306 features analysed, 292 features demonstrated significant differences between Tor and nonTor encrypted payloads, resulting in a high differentiation rate of 95.42%.

4.4 Statistical Analysis Findings

The four sets of features, including (1) the frequency of hex characters (*F_ 0-F_ f*), (2) the ratio of the frequency of hex characters (*R_ 0-R_ f*), (3) the total number of characters (T), and (4) entropy (E) were analysed using descriptive statistics to study the characteristics of Tor and nonTor encrypted payloads based on character analysis, and inferential statistics to generalise the findings. The insights and implications of this study, as well as the addressing of Research Question Q1 and Hypothesis H_{01}, are presented in the following section.

Descriptive Statistics

A Hex Character Frequency Set: The findings suggest that the frequency of any individual feature in a single encrypted payload with around 100 characters can potentially be used to identify Tor packets, as this characteristic is unique to Tor and not present in nonTor traffic. Additionally, the measures of dispersion indicate that Mean values of individual hex character frequency features ranging from 65 to 80 characters can potentially be used to identify Tor packets, as this characteristic is also unique to Tor. Similarly, SD values ranging from 12 to 35 characters can potentially be used to

identify Tor packets, except in VoIP where they are quite close to nonTor, making them less reliable as identifiers. A Min value of more than 0 characters is likely to be indicative of Tor traffic, but Max values do not reveal any clear patterns between Tor and nonTor applications, and are therefore not recommended as identifiers.

A Hex Character Frequency Ratio Set: The findings of this study suggest that the hex character frequency ratio exhibits a very similar pattern in both Tor and nonTor applications, although there are slight differences in some applications. This may be the effect of normalisation, which produces very small and subtle values that make it difficult to notice any significant differences in the graphs. In terms of the measures of dispersion, similar to a frequency set, the Min with ratio values more than 0 is likely to be Tor. However, it is not recommended to use Mean, SD, and Max as identifiers for Tor as there is no clear pattern between Tor and nonTor traffic.

Total Characters: The findings suggest that the measurements analysed in this study can potentially be used to identify Tor packets based on the following characteristics: total characters with around 1000 characters, Mean values with around 1000 characters, SD values with less than around 500 characters, and Min values more than 60 characters, except for Max values, as they do not reveal any unique pattern compared to nonTor applications. This insight is supported by the fact that Tor utilises fixed-size cells of 512 bytes, as opposed to the variable packet lengths in nonTor applications.

Entropy: The entropy value of Tor and nonTor encrypted payload across all applications was found to be almost reaching 4. However, the measurements analysed in this study suggest that Tor packets can be potentially identified based on specific characteristics, including an entropy value of less than 3.98, SD value of less than 0.01, and Min value of more than 3.99, except for Mean and Max values which do not reveal any unique patterns compared to nonTor encrypted payload. This insight can be further supported by the fact that Tor uses multiple encryptions on fixed-size cells of 512 bytes, as opposed to the one-time encryption on variable packet lengths in nonTor applications.

The findings on these four sets of features suggest that nonTor traffic tends to have larger and more variable encrypted payloads compared to Tor traffic. This could be attributed to the diverse nature of nonTor traffic, which might encompass a wider range

of applications, data types, and communication patterns. In contrast, Tor traffic appears to be more homogeneous across most features, which reflects a potentially effective approach for classifying Tor traffic.

Inferential Statistics

The results discussed in Section 4.3 revealed that among the 306 features analysed, a significant difference was observed in 292 features between Tor and nonTor traffic, resulting in a high differentiation rate of 95.42%. This finding serves as strong evidence to reject the first null hypothesis and supports the conclusion that encrypted payloads differ between Tor and nonTor traffic. The statistical analysis findings suggest that hex character statistics analysis can be used as an effective method to distinguish between Tor and nonTor traffic based on the observed differences in feature frequencies.

Similar to our study, which employs inferential statistics to test the hypothesis and draw conclusions from the results. In a study by Cuzzocrea et al. (2017), machine learning techniques were employed to detect Tor traffic using time-based features. The study hypothesised that there were no significant differences between the considered features of Tor-generated traffic flow and normal traffic flow. This hypothesis was tested using the Mann-Whitney test at a significance level of .05. The results of the hypothesis testing showed that all of the considered features passed the test, suggesting that the feature set could be a viable approach for discriminating between Tor and nonTor network traffic.

In conclusion, the findings presented in this study significantly contribute to the understanding of the distinctive characteristics of encrypted payloads in both Tor and nonTor traffic. Our analysis reveals that nonTor traffic typically exhibits larger and more variable encrypted payloads in comparison to Tor traffic. This observation can likely be attributed to the diverse nature of nonTor traffic, which encompasses a broader range of applications, data types, and communication patterns. In contrast, Tor traffic demonstrates greater homogeneity in terms of payload size, reflecting the main research objective of distinguishing between Tor and nonTor traffic.

4.5 Summary

This study presented the results of both descriptive and inferential statistical analyses. Descriptive statistics were used to describe the characteristics of Tor and nonTor encrypted payloads, followed by hypothesis testing using the Mann-Whitney test in inferential statistics. The findings suggest that there were significant differences between Tor and nonTor payloads, which could aid in the development of more effective methods for detecting and classifying encrypted network traffic. These methods will be discussed in the next chapter.

Chapter 5

Machine Learning Modelling and Analysis of Tor Traffic

The results presented in Chapter 4 indicate a clear distinction between Tor and nonTor encrypted payloads based on character analysis. This finding highlights an opportunity to develop an automated tool for the efficient identification and classification of Tor traffic. The present chapter introduces an approach that utilises ML modelling to analyse Tor traffic. This approach aims to address the second research question: "Can we distinguish Tor from nonTor traffic using the encrypted payload in a data-efficient manner?" and to test its corresponding null hypothesis: "A single encrypted payload cannot be used to identify Tor traffic". To ensure the reliability and validity of our scientific findings, we performed classification experiments using two different datasets. Additionally, we strengthened the experimental design by demonstrating feature selection and predicting unseen data from the final model. To provide a thorough comparison, experiments were conducted using both the WEKA and Python-based platforms, both of which are widely used for classification tasks. The findings of the ML analysis are discussed in detail at the end of this chapter.

5.1 Character Statistics-Based Features Approach

In this section, we provide an overview of the key aspects that constitute the classification process. These components include defining the features used as input, splitting the data into training and testing sets, selecting supervised learning classifiers, and evaluating model performance.

5.1.1 Proposed Classification Features

Character analysis of the encrypted payload serves as the basis for identifying the features to be used subsequently in classification. As summarised in Section 4.3, the statistical findings clearly demonstrate that the majority of the proposed features (292 out of 306 features, or equivalently 95.42%) are unique to Tor traffic, allowing for the possibility of exploiting these features in classifying Tor traffic. However, it is important to consider the robustness of cryptographic designs, which ideally should not reveal any meaningful data from a given ciphertext of the same length. Yet, such an ideal cannot always be guaranteed in real-world implementations, as noted by (Katz & Lindell, 2020). In response to these considerations, we selected a set of features that include the hex character frequency ratio (R_0-R_f) set, which is unaffected by variations in payload length, and the hex character frequency (F_0-F_f) set for comparison purposes. The ratio set is our primary focus, with the frequency set serving as a comparative benchmark. However, we excluded the total characters (T) and entropy (E) features from our analysis as they were not of primary interest.

To ensure that the R_0-R_f features were not influenced by the payload size, a correlation test was conducted using the Pearson correlation coefficient. This test aimed to determine the correlation between two quantitative variables, particularly between R_0-R_f and T. For comparative purposes, a similar correlation test was performed between F_0-F_f and T to assess their relationship. This analysis enabled an understanding of the impact of payload size on the selected features.

The Pearson correlation coefficient ranges from -1 to +1, with a negative value indicating an inverse correlation, a positive value indicating a direct correlation, and a

value of zero indicating no correlation (Benesty et al., 2009). The null hypothesis H_0 was defined as two unrelated variables, while the alternative hypothesis H_1 was defined as two related variables. The significance level for the test was set at 0.05. If the calculated *p-value* was less than this threshold, then the null hypothesis was rejected.

Table 5.1: Pearson correlation coefficient between Total Character (T) and Hex Character Frequency Ratio R_0–R_f features across all applications

Application Types	Pearson Correlation Coefficient between Total Characters (T) and Hex Character Frequency Ratio Values of															
	R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_a	R_b	R_c	R_d	R_e	R_f
Public Dataset																
Audio	.01	.04	.03	.02	.03	.00	.02	.00	.03	.03	.04	.06	.02	.03	.00	.05
Browsing	.58	.15	.16	.17	.18	.18	.18	.19	.18	.18	.18	.19	.19	.19	.19	.19
Chat	.20	.01	.01	.00	.02	.04	.02	.00	.02	.01	.01	.02	.01	.03	.01	.04
Email	.66	.16	.19	.21	.18	.22	.20	.20	.20	.21	.21	.23	.23	.21	.21	.21
FTP	.05	.01	.05	.03	.08	.05	.08	.03	.06	.01	.03	.03	.04	.04	.03	.02
P2P	.13	.02	.01	.06	.00	.06	.04	.08	.03	.06	.04	.05	.07	.04	.08	.06
Video	.37	.07	.07	.08	.08	.08	.08	.10	.08	.11	.09	.09	.10	.09	.09	0.1
VoIP	.38	.04	.06	.06	.00	.13	.05	.15	.16	.03	.17	.14	.15	.16	.08	.18
Private Dataset																
Browsing	.12	.02	.00	.00	.02	.02	.02	.01	.03	.00	.01	.00	.00	.00	.02	.00

* Note that the values in cells with a white background colour equal the asterisk (*) indicating that the correlation is significant at the 0.05 level

The results of the Pearson correlation coefficient tests conducted on SPSS between T and R_0-R_f features across different applications in public and private datasets are presented in Tables 5.1. This table shows the correlation coefficient values and their corresponding p-values resulting from the correlation tests. The asterisk (*) values, which are presented in cells with a white background colour in this table due to space limitations, indicate that the null hypothesis H_0 is rejected, suggesting a significant correlation between the variables. However, it is important to note that despite the rejection of the null hypothesis for many of these values, most of the correlation coefficients are still close to zero, with the majority of values ranging from 0.23 to 0.00, indicating a negligible positive correlation. This suggests that even though there is a statistically significant relationship between T and R_0-R_f features, the actual strength of this

relationship is very weak. Only a few cases have high values, such as *R_0* for Email at 0.66, *R_0* for Browsing at 0.58, *R_0* for VoIP at 0.38 and *R_0* for Video at 0.37.

The presence of predominantly weak correlations, despite the statistical significance, supports the idea that normalisation is an appropriate approach to eliminate the impact of payload size. By normalising the data, we can minimise the influence of total characters on the features. As a result, the ratio set is suitable for classification. We further investigated the features related to the absolute frequency values, without normalising them to a ratio and determine their correlation with the total characters.

Table 5.2: Pearson correlation coefficient between Total Character (T) and Hex Character Frequency Ratio *F_0–F_f* features across all applications

Application Types	Pearson Correlation Coefficient between Total Characters (T) and Hex Character Frequency Values of															
	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_a	F_b	F_c	F_d	F_e	F_f
Public Dataset																
Audio	.98	.98	.98	.97	.98	.98	.97	.98	.98	.98	.98	.98	.98	.98	.98	.98
Browsing	.98	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99
Chat	.98	.98	.98	.98	.98	.98	.98	.98	.98	.98	.98	.98	.98	.98	.98	.98
Email	.98	.99	.99	.99	.99	.98	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99
FTP	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99
P2P	.72	.98	.98	.98	.98	.98	.97	.98	.98	.98	.98	.98	.98	.98	.98	.94
Video	.98	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99
VoIP	.89	.98	.98	.98	.98	.98	.94	.98	.98	.98	.98	.98	.98	.98	.98	.98
Private Dataset																
Browsing	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99	.99

* Correlation is significant at the 0.05 level

Table 5.2 presents the results of the Pearson correlation coefficient tests (using SPSS) between T and *F_0-F_f* features in different types of applications in the public and private datasets. This shows the correlation coefficient values and associated *p-values* resulting from the correlation tests. The correlation coefficients between T and *F_0-F_f* range between 0.72 and 0.99 for all applications, with most values exceeding 0.95. This high correlation between the variables T and *F_0–F_f* indicate a strong positive relationship between them in these applications. Moreover, the rejection of the null

hypothesis H_0 for values marked with an asterisk (*) further confirms this association. Consequently, we can infer that the variable T has a significant impact on the features of *F_0–F_f*. However, for the experiments, we decided to retain the *F_0–F_f* feature for the purpose of analysis and comparison with the *R_0–R_f* features.

Before delving into the experimental results, we present the list of features used in the classification experiments for the sake of clarity and coherence. Table 5.3 comprehensively enumerates the 32 features used in the classification process. These features were categorised into three sets for easier identification: Set 1 comprises the hex character frequency (*F_0-F_f*), Set 2 is the hex character frequency ratio (*R_0-R_f*), while Set 3 combines both Set 1 and Set 2. The integration of Set 1 and Set 2 into Set 3 facilitates a more comprehensive perspective on the features, which may improve the accuracy of the classification process, ultimately leading to the development of a robust and reliable classification model. Consequently, utilising Set 3 in the classification experiments may yield a more robust and reliable classification model.

Table 5.3: List of features used in the classification experiments

Set Name	**List of Features**	**Total Number of Features**
Set 1	hex character frequency (*F_0-F_f*)	16
Set 2	hex Character frequency ratio (*R_0-R_f*)	16
Set 3	The combination of Set 1 and Set 2	32

The following section will explain the data-splitting technique used to train and test the supervised learning algorithms.

5.1.2 Data Splitting for Training and Testing Dataset

As outlined in Section 3.4.2, the datasets in our study were initially divided into 'seen' and 'unseen' sets. The unseen set was reserved for testing the final model and evaluating its performance on new data, as detailed in Section 5.5. The seen set was used for the classification process and further split into training and testing sets. This splitting can be conducted in various ways; in our study, we implemented both 10-fold cross-validation (10-CV) and a 70-30 percentage split.

We acknowledge a methodological limitation in not repeating the 70-30 percentage split multiple times. Repeating splits and averaging results are standard practices that provide a more robust evaluation by minimising bias from any single data split. This approach ensures that model performance is not excessively influenced by the specific distribution of a particular split. Although 10-CV helps reduce bias by utilising all available data for both training and testing, unlike the percentage split, it has its drawbacks. The 10-CV process can be time-consuming, especially with large datasets, and was constrained in our study by the limitations of our computational resources.

Consequently, the choice between 10-CV and percentage splitting was influenced by the size of the dataset. For datasets with fewer than 10,000 instances, 10-CV was employed. In contrast, for larger datasets, the percentage split method proved more practical. This decision was based on balancing the need for robust data analysis against the computational resources available for the study. Future research could address this limitation by considering multiple repetitions of the data splitting process using a robust three-phase CV approach including training, validation, and testing phases.

5.1.3 Classification Algorithms

The selection of the supervised learning algorithms for classification was influenced by previous research on NTC and text analysis, as discussed in Section 2.6.5. Three algorithms were selected: DTs due to its ease of understanding and interpretability while still being effective; RFs, which improves accuracy and reduces overfitting problems; and kNN for its simplicity and flexibility. These algorithms were employed to compare the classification results. Table 5.4 provides a list of classifiers on the WEKA and Scikit-learn platforms using their default parameters.

5.1.4 Model Performance Metrics

In this study, the evaluation of classification models involved the use of various metrics, including accuracy, precision, recall, F1 score, and AUC score. These metrics were derived from the measurements of TP, FP, TN, and FP in a confusion matrix, as explained in Section 2.6.7. Accuracy was used to assess how well the models predicted

Table 5.4: Supervised learning algorithms used in WEKA and Scikit-learn

Supervised Learning Algorithms	**Libraries used in**	
	WEKA	**Scikit-learn**
DTs	J48	DecisionTreeClassifier
RFs	RandomForest	RandomForestClassifier
kNN	IBk	KNeighborsClassifier

whether a packet was Tor or nonTor, with a higher accuracy value being preferred. A 100% accuracy indicates a perfect model. Precision was calculated by dividing TP by the total number of positive predictions, while recall was calculated by dividing TP by the total number of actual positives. A perfect classifier would have precision and recall values of one. The F1 score is a weighted harmonic mean of precision and recall, providing a single score that balances both metrics. The AUC score measures the degree of distinction between the predictions of the two classes by comparing the TP Rate to the FP Rate. A higher AUC score indicates a greater distinction between the predictions of the two classes.

5.2 Classification on Two-Class

This section presents the binary classification results of nine different application types, comprising eight from the public dataset and one from the private dataset. It is essential to have rich datasets with multiple binary class pairs to effectively test the null hypothesis and address the research question of whether a single packet can be used to efficiently identify Tor traffic.

Table 5.5 provides a comprehensive list of the number of instances for each application in the public and private datasets, along with the test options used in the classification. We used a 10-CV approach to split the Chat and Email, while a percentage split of 70/30 for training/testing was utilised for the rest of the application types. We employed both WEKA and Scikit-learn platforms to compare and discuss their results and processing tasks.

Table 5.5: Number of instances and testing options for nine applications

Application Types	#instances	Test Options
Public Dataset		
Audio	26,082	Percentage Split
Browsing	71,950	Percentage Split
Chat	6,504	10-CV
Email	12,300	10-CV
FTP	514,952	Percentage Split
P2P	433,770	Percentage Split
VDO	32,154	Percentage Split
VoIP	737,382	Percentage Split
Private Dataset		
Browsing	29,600	Percentage Split

5.2.1 WEKA Results

The classification of Tor traffic was performed in WEKA using three supervised learning algorithms - J48, RFs, and IBk - and nine binary-class pairs based on three feature sets. The results of the Tor classification accuracy using three set features among various supervised learning algorithms in WEKA are presented in Table 5.6.

The table displays the accuracy scores for three algorithms on nine binary-class pairs, utilising three feature sets in WEKA. Averaging the scores, RandomForest demonstrated the highest accuracy rates, with 96.59% for Set 1 and 97.81% for Set 3. J48 achieved the best performance on Set 2, with a 95.81% accuracy rate. Although IBk's accuracy rates were marginally lower than RandomForest's for Set 1 (96.46%) and Set 3 (95.30%), its performance significantly dropped to 73.23% for Set 2.

Examining the results for each dataset source revealed that, for the public dataset, IBk performed the best on Set 1, with accuracies ranging from 99.77% for VoIP to 92.04% for Chat. On Set 2, J48 had the highest accuracy rates, ranging from 99.63% for VoIP to 88.99% for audio. For Set 3, RandomForest achieved the best performance, with 99.90% for VoIP to 94.84% for audio. In the private dataset, RandomForest had the highest accuracy for Set 1 at 97.06%, while J48 achieved 97.12% for Set 2, and RandomForest reached 97.23% for Set 3.

Table 5.6: Accuracy of three feature sets in public and private datasets using J48, RandomForest and IBk in WEKA

Application types	J48			RandomForest			IBk		
	Set 1	Set 2	Set 3	Set 1	Set 2	Set 3	Set 1	Set 2	Set 3
Public Dataset									
Audio	90.01	88.99	94.95	91.97	88.99	94.84	92.04	65.80	90.51
Browsing	94.35	99.03	97.86	96.56	94.26	97.88	96.86	70.44	95.99
Chat	91.34	93.36	95.97	95.94	86.37	97.49	96.88	60.62	95.45
Email	97.08	91.85	98.05	98.20	92.85	98.41	97.64	89.24	96.08
FTP	93.92	96.43	98.28	95.52	91.58	97.97	94.40	72.41	93.75
P2P	95.00	98.44	98.45	96.53	96.82	98.30	96.23	69.14	95.24
VDO	96.38	97.46	97.43	97.78	91.18	98.24	97.57	68.53	96.75
VoIP	99.62	99.63	99.81	99.77	98.92	99.90	99.77	93.98	99.76
Private Dataset									
Browsing	95.55	97.12	97.07	97.06	90.81	97.23	96.73	68.90	94.18
Average Scores									
9 datasets	94.81	95.81	97.54	96.59	92.42	97.81	96.46	73.23	95.30

In conclusion, when comparing the different feature sets, Set 3 outperformed Set 1 and Set 2 for all models except IBk, indicating that integrating features from absolute frequency and frequency ratio can improve the models. The performance of Set 1 and Set 2 varied depending on the algorithm and binary class, with some applications showing only slight differences in accuracy between the two sets, while in others, there was no significant difference between the two sets. Moreover, in both public and private datasets, the accuracy results showed a similar trend of high accuracy, particularly in J48 and RandomForest, with overall scores exceeding 86% across all feature sets. This indicates that the proposed approach for Tor traffic classification can successfully detect Tor traffic from multiple data sources, including both public and private datasets. It is noteworthy that the lowest accuracy scores consistently emerged from the Audio or Chat categories, while the highest scores were consistently obtained from the VoIP category. This observation suggests that the accuracy scores may be heavily influenced by the size of the datasets, with larger datasets yielding higher accuracy scores. This emphasises

the importance of having adequate data for training and validating the models, as it can significantly impact their performance.

Our investigation was to identify the best model for Tor traffic classification using average accuracy scores, with a focus on Set 2 features. Our findings revealed that RandomForest achieved the highest accuracy score of 92.42%, while IBk had the highest accuracy score of 73.23%. However, the best model for Tor traffic classification was found to be J48, achieving an accuracy score of 95.81% and considered the most reliable for Tor traffic classification based on hex character frequency ratio features.

The corresponding bar graphs of the table can be found in Figure 5.1. The graphs illustrate the classification accuracy of Tor traffic using three feature sets and three different algorithms in WEKA, J48 in (a), RandomForest in (b), and IBk in (c), across all application types. The x-axis represents the application types, while the y-axis represents the accuracy scores, starting from 60%. The graphs indicate that Set 1 and Set 3 features displayed a consistent trend across all three classifiers, while Set 2 features displayed varying trends based on the algorithm used. J48 and RandomForest displayed a similar trend, both achieving an accuracy score over 80%, unlike, IBk produced significantly lower accuracy scores, all below 75%, except for VoIP.

As discussed above, J48 was identified as the best model for Tor traffic classification when considering Set 2 features, achieving an average accuracy score of 95.81% in WEKA. We conducted a more in-depth investigation into the performance of this model, and Table 5.7 presents the precision, recall, F1 score, and AUC/ROC score on average for the J48 algorithm using Set 2 features across all applications in WEKA.

From the table, the public dataset showed the highest precision, recall, F1 score, and AUC/ROC scores in Browsing with a score of 0.99, while the lowest scores were found in Email with a score of 0.80. Similarly, the private dataset, which contained only browsing applications, also showed high precision, recall, F1 score, and AUC/ROC scores for the J48 algorithm, with all scores being 0.96. The average scores for all application types from both the public and private datasets had average precision and recall values of 0.94 and 0.93, respectively. This suggests that the J48 algorithm can achieve a 94% correct detection rate and a 93% correct alarm rate when classifying Tor and nonTor

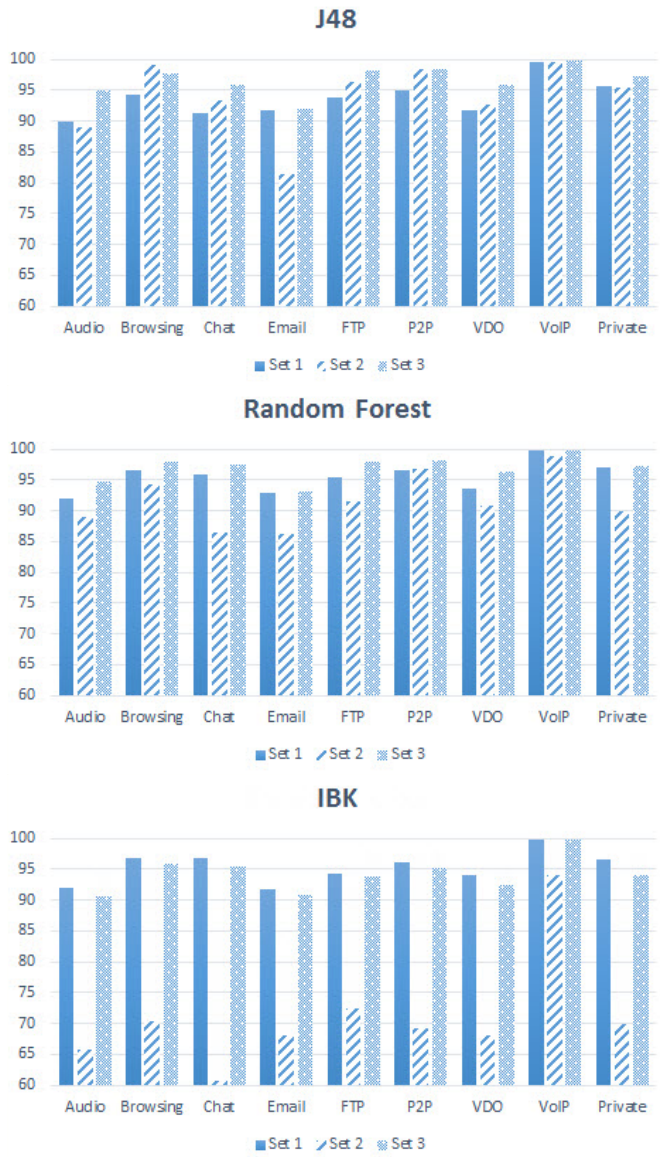


Figure 5.1: Accuracy of Tor traffic classification using J48, Random Forest, and IBk on 3 feature sets in WEKA for both public and private datasets

traffic. These results indicate that the J48 algorithm is a promising tool for accurately classifying Tor and nonTor traffic.

The evaluation metrics used, including accuracy, precision, recall, F1 score, and AUC/ROC scores, have substantiated the efficacy of the J48 classifier in detecting Tor traffic across various applications within both public and private datasets. The use of

Table 5.7: Precision, Recall, F1 score and AUC/ROC results of the J48 model with Set 2 features in WEKA for both public and private datasets

Application Types	**J48**			
	Avg. Precision	**Avg. Recall**	**Avg. F1 score**	**Avg. AUC/ROC**
Public Dataset				
Audio	0.89	0.89	0.89	0.89
Browsing	0.99	0.99	0.99	0.99
Chat	0.94	0.93	0.93	0.93
Email	0.80	0.80	0.80	0.80
FTP	0.97	0.97	0.97	0.97
P2P	0.99	0.99	0.98	0.98
VDO	0.93	0.93	0.93	0.93
VoIP	0.96	0.97	0.96	0.96
Private Dataset				
Browsing	0.96	0.96	0.96	0.96
Average Scores	0.94	0.94	0.94	0.95

these diverse datasets enhances the reliability and validity of our research findings. However, to ensure that our results are not solely influenced by the specific implementations of ML algorithms for DTs, RFs and kNN in WEKA, we conducted parallel investigations using the same types of algorithms in Scikit-Learn. The details of these additional investigations will be discussed in the following section.

5.2.2 Scikit-Learn Results

Similar to the classification conducted in WEKA, we applied three supervised learning algorithms, including Decision Tree, Random Forest and KNN to nine binary-class pairs based on three feature sets in Scikit-learn. The results of the Tor traffic classification accuracy using three set features among various supervised learning algorithms in Scikit-learn are presented in Table 5.8.

The table presents the accuracy scores for three algorithms on nine binary-class pairs, using three feature sets in Scikit-learn. Averaging the scores, KNN exhibited the

Table 5.8: Accuracy of three feature sets in public and private datasets using Decision Tree, Random Forest, and KNN in Scikit-learn across all applications

Application types	Decision Tree			Random Forest			KNN		
	Set 1	Set 2	Set 3	Set 1	Set 2	Set 3	Set 1	Set 2	Set 3
Public Dataset									
Audio	88.00	87.85	94.36	91.65	87.12	94.40	92.42	66.84	93.04
Browsing	93.47	97.95	97.39	96.51	90.34	97.81	97.13	75.45	97.22
Chat	92.55	83.50	94.59	96.59	81.41	97.01	97.56	67.29	97.89
Email	96.68	89.72	97.48	98.18	92.83	98.21	98.05	87.86	98.08
FTP	92.84	96.00	97.93	95.57	89.57	97.83	95.89	76.30	96.12
P2P	93.12	97.64	98.19	96.49	94.44	98.27	96.80	71.59	96.94
VDO	95.45	96.19	96.92	97.52	87.72	97.88	98.04	72.06	98.29
VoIP	99.67	99.64	99.89	99.76	98.32	99.88	99.88	93.42	99.89
Private Dataset									
Browsing	93.57	95.23	97.34	97.10	87.50	97.31	97.47	71.81	97.47
Average Scores									
9 datasets	94.06	93.87	97.15	96.60	89.92	97.62	97.07	75.99	97.25

highest accuracy rates, with 97.07% for Set 1. The Decision Tree achieved the best performance on Set 2, with a 93.87% accuracy rate. Meanwhile, RF attained the best performance on Set 3, with a 97.62% accuracy rate. Similar to the results in WEKA, KNN's performance was comparable to that of other classifiers but significantly dropped to 73.23% for Set 2.

Upon examining the results for each dataset source, it was found that for the public dataset, KNN performed the best on Set 1, with accuracies ranging from 92.42% for Audio to 99.88% for VoIP. In Set 2, the Decision Tree achieved the highest accuracy rates, ranging from 83.50% for Chat to 99.64% for VoIP. For Set 3, all algorithms displayed similar best performance with only slight and insignificant differences observed, with values above 93% for Audio to above 99.88% for VoIP. In the private dataset, KNN had the highest accuracy for Set 1 at 97.47%, while the Decision Tree reached 95.23% for Set 2, and KNN achieved 97.47% for Set 3.

In conclusion, when comparing different feature sets, Set 3 outperformed Set 1 and

Set 2 for all models, indicating that combining features from absolute frequency and frequency ratio can enhance the models. The performance of Set 1 and Set 2 varied depending on the algorithms and binary class, with some applications exhibiting only minor differences in accuracy between the two sets, while in others, no significant difference was observed. Furthermore, in both public and private datasets, the accuracy results displayed a similar trend of high accuracy, particularly with the Decision Tree and Random Forest algorithms, as overall scores exceeded 87.50% across all feature sets. This finding suggests that the proposed approach for Tor traffic classification can effectively detect Tor traffic from various data sources, including both public and private datasets. It is worth noting that the lowest accuracy scores consistently appeared in the Audio or Chat categories, while the highest scores consistently emerged from the VoIP category. This pattern implies that the accuracy scores may be heavily influenced by dataset size, with larger datasets yielding higher accuracy scores. This highlights the importance of having sufficient data for training and validating the models, as it can significantly impact their performance.

Our investigation was to identify the best model for Tor traffic classification using average accuracy scores, with a focus on Set 2 features. Our findings revealed that Random Forest achieved an accuracy score of 89.92%, while KNN had an accuracy score of 75.99%. However, the best model for Tor traffic classification was found to be Decision Tree, achieving an accuracy score of 93.87% and considered the most reliable for Tor traffic classification based on hex character frequency ratio features.

The corresponding bar graphs of the table can be found in Figure 5.2. The graphs illustrate the classification accuracy of Tor traffic using three feature sets and three different algorithms, Decision Tree in (a), Random Forest in (b), and KNN in (c), across all application types. The x-axis represents the application types, while the y-axis represents the accuracy scores, starting from 60%. The graphs indicate that Set 1 and Set 3 features exhibited a consistent trend across all three classifiers, while Set 2 features displayed varying trends based on the algorithm used. Decision Tree and Random Forest displayed a similar trend, both achieving an accuracy score over 80%. However, KNN produced significantly lower accuracy scores, all below 75%, except for VoIP and

Private-Browsing with above 72%.

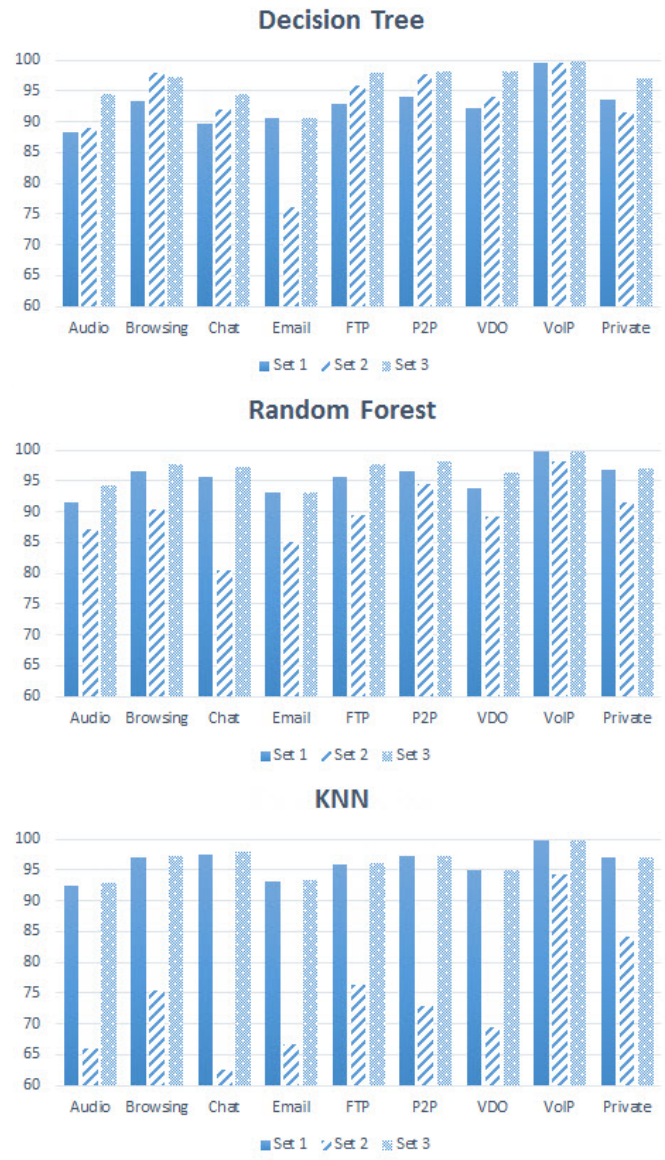


Figure 5.2: Accuracy of Tor traffic classification using Decision Tree, Random Forest and KNN on 3 feature sets in Scikit-learn for both public and private datasets

As discussed above, Decision Tree was identified as the best model for Tor traffic classification when considering Set 2 features, achieving an average accuracy score of 93.87% in Scikit-Learn. Further investigation was conducted into the performance of this model, and Table 5.9 presents the precision, recall, F1 score, and AUC/ROC score on average for the Decision Tree algorithm using Set 2 features across all applications in Scikit-Learn.

Table 5.9: Precision, Recall, F1 score and AUC/ROC results of the Decision Tree model of Set 2 features in Scikit-learn for both public and private datasets

Application Types	**Decision Tree**			
	Avg. Precision	**Avg. Recall**	**Avg. F1 score**	**Avg. AUC/ROC**
Public Dataset				
Audio	0.89	0.88	0.88	0.88
Browsing	0.98	0.98	0.98	0.98
Chat	0.84	0.84	0.83	0.83
Email	0.90	0.90	0.90	0.90
FTP	0.96	0.96	0.96	0.96
P2P	0.98	0.98	0.98	0.98
VDO	0.96	0.96	0.96	0.96
VoIP	1.00	1.00	1.00	1.00
Private Dataset				
Browsing	0.96	0.96	0.96	0.96
Average Scores	0.94	0.94	0.94	0.94

The results show that all metric performances were above 0.83 in all nine binary classes. In the public dataset, the lowest average scores for AUC/ROC and F1 score were 0.83, and for precision and recall were 0.84, obtained for the Chat, while the highest average score of 1.00 was achieved for VoIP. In the private dataset, which contained only browsing traffic, the Decision Tree algorithm achieved very high precision, recall, F1 score, and AUC/ROC scores, all at 0.96 on average. The average scores for all application types from both datasets had precision and recall values of 0.94, indicating that the Decision Tree algorithm can achieve a 94% correct alarm rate and correct detection rate when classifying Tor and nonTor traffic. These results suggest that the

Decision Tree algorithm is a promising tool for accurately classifying Tor and nonTor traffic.

The evaluation metrics used, including accuracy, precision, recall, F1 score, and AUC/ROC scores, have confirmed the effectiveness of the Decision Tree classifier in detecting Tor traffic across various applications within both public and private datasets. The use of both types of datasets enhances the reliability and validity of our research findings. Furthermore, the consistent results obtained from using WEKA and Scikit-learn provide confidence in our proposed approach, which is based on character analysis for classifying Tor traffic, and demonstrate that these results are independent of the specific implementations of the algorithms in these platforms. To gain deeper insights, we also performed multi-class classification within the Tor network, which is presented in the next section.

5.3 Classification on Eight-Class

As a means of further exploring the efficacy of our proposed approach for Tor traffic classification, we expanded our experiments to include an eight-class classification of the public dataset within the Tor network. This allowed us to investigate whether it was possible to classify different application types within Tor based on the same encrypted payload characteristics. To ensure the validity and reliability of our findings, we tested the classifications using both WEKA and Scikit-learn.

Table 5.10: Accuracy of eight-class classification of the public dataset in WEKA

Application types	J48			RandomForest			IBk		
	Set 1	Set 2	Set 3	Set 1	Set 2	Set 3	Set 1	Set 2	Set 3
8-class	17.68	14.75	17.48	21.43	18.23	21.88	17.95	13.63	17.25

Table 5.10 and 5.11 present the accuracy scores for the eight-class of Tor traffic using J48, RandomForest, and IBk in WEKA, and Decision Tree, Random Forest, and KNN in Scikit-Learn. The results indicate that all accuracy scores were significantly below 25%, with RF achieving the highest accuracy scores, ranging from 21.88% to 23.20% using

Table 5.11: Accuracy of eight-class classification of the public dataset in Scikit-Learn

Application types	Decision Tree			Random Forest			KNN		
	Set 1	Set 2	Set 3	Set 1	Set 2	Set 3	Set 1	Set 2	Set 3
8-class	17.89	14.28	17.97	23.16	19.05	23.20	19.62	13.93	19.82

Set 3 features. These findings highlight the complete ineffectiveness of the classification methods in distinguishing application types within Tor networks. The primary reason for this classification failure is that the encrypted payloads exhibit similar characteristics in terms of character analysis.

The results from the eight-class classification clearly demonstrate that hex character statistics analysis of a single packet cannot be used to classify application types within the Tor network due to the identical characteristics shared by encrypted Tor payloads. While our proposed features can distinguish between Tor and nonTor encrypted traffic, there is room for improvement in model performance by selecting only useful features, as discussed in the following section.

5.4 Feature Selection

As discussed in Section 2.6.4, feature selection is a crucial process in ML that involves selecting the best and most promising features from a feature set to reduce training time and maintain or improve classifier performance. In the case of network traces, where a large number of packets are involved, limiting the number of input variables for NTC can help reduce the computational cost and result in faster traffic classification, especially for real-time NTC that require high-speed processing with packet inspection. Therefore, feature selection is an important technique for reducing the search space of the ML algorithm, particularly when dealing with large datasets like network traces (Dehghani et al., 2010).

In our study, WEKA was employed to perform the feature selection experiment, utilising the *WrapperSubsetEval+BestFirst* method with the J48 classifier. This method combines the wrapper method (*WrapperSubsetEval*) with a search algorithm (*BestFirst*)

to select the best subset of features for the classifier. The *WrapperSubsetEval* method evaluates the entire feature subset by creating classifiers for each possible subset of features, aiming to find the best subset that fits the particular classifier being used. The *BestFirst* search algorithm then identifies the optimal subset of features to use in the classifier based on the evaluations provided by the chosen attribute evaluator. The combination of *WrapperSubsetEval* and *BestFirst* offers a balance between searching for the optimal feature subset and computational efficiency (Witten et al., 2017).

Table 5.12: Results of feature selection for nine application types from both public and private datasets using WEKA

Application types	Feature Selection *WrapperSubsetEval+BestFirst*		
	Before Feature Selection	After Feature Selection	
	Accuracy (%)	Selected Feature	Accuracy (%)
Public Dataset			
Audio	88.99	*R_0*	99.7
Browsing	99.03	*R_3*	99.2
Chat	93.36	*R_6*	97.4
Email	91.85	*R_0*	96.5
FTP	96.43	*R_d*, *R_e*	98.5
P2P	98.44	*R_3*, R_a	99.1
VDO	92.76	*R_1*	99.3
VoIP	99.63	*R_0*	99.9
Private Dataset			
Browsing	97.12	*R_2*, *R_e*	97.9
Average Scores			
9 Datasets	93.96	n/a	98.61

Table 5.12 presents the results of feature selection experiments using the *WrapperSubsetEval+BestFirst* method on nine different application types from both public and private datasets. The 'Before Feature Selection' column shows the accuracy achieved without feature selection, while the 'After Feature Selection' column displays the accuracy achieved after selecting the most relevant features. As demonstrated by the table, feature selection leads to an improvement in accuracy for all application types.

It is important to note that the most informative feature for distinguishing between the two classes will be selected. Consequently, different features (R_0 to R_f) may be chosen based on the specific binary class being examined, as they provide the most information for that particular classification task. For instance, for the Browsing of the public dataset, the feature R_3 was selected as the most relevant feature among 16 features. In contrast, for the FTP, features R_d and R_e were found to be the most relevant. Although the accuracy improvements in these applications, as well as in P2P, VDO, and private browsing, were only slight, the reduction of features can still enhance model performance. In certain application types, such as Audio, Email, and VoIP, feature selection significantly improves accuracy. The greatest improvement in accuracy was observed in the Audio, which increased from 88.99% to 99.7% after feature selection. On average, the scores for all nine datasets increased from 93.96% to 98.61% after feature selection. This demonstrates the effectiveness of feature selection not only in enhancing classification accuracy but also in reducing processing time.

5.5 Unseen Data Prediction

As discussed in Section 3.4.4, preventing overfitting is a critical aspect of developing robust ML models. To achieve this, it is essential to test the finalised model using new, unseen data to ensure that the model is not merely memorizing the training data. In our study, the additional unseen data was unavailable. Therefore, as explained in Section 3.5.2, we designated a 5% ratio of the balanced data as unseen data to evaluate the model's performance and minimize the risk of overfitting.

To perform predictions on the unseen data, we used the finalised model obtained from Section 5.4 and tested it with the new data, considering the selected features of each class. Table 5.13 presents the results of testing the finalised model on unseen data for various application types from both public and private datasets. The unseen data was split from the original dataset, as explained in Section 3.5.2. The 'Application Types' column lists the different types of applications, and the '#Unseen Instances' column shows the number of instances in the unseen dataset for each application type. The 'Accuracy (%)' column displays the accuracy of the model's predictions on the unseen

Table 5.13: Unseen dataset testing results with finalised model

Application Types	#Unseen Instances	Accuracy (%)
Public Dataset		
Audio	1,372	97.23
Browsing	3,786	99.29
Chat	342	98.83
Email	648	92.99
FTP	27,102	96.94
P2P	22,830	99.10
Video	1,692	99.62
VoIP	38,810	99.97
Private Dataset		
Browsing	1,558	98.53
Average	n/a	98.06

data for each application type.

The results indicate that the model's predictions on the unseen data for all application types of the public dataset have an accuracy exceeding 90%, with the highest accuracy of 99.97% achieved for VoIP. Similarly, Browsing in the private dataset also achieved a very high accuracy of 98.53%. The average accuracy across all application types was 98.06%. These results demonstrate the reliability and generalisability of the finalised model, which was tested on previously unseen data to ensure that it can accurately classify new instances.

We have demonstrated the classification of Tor traffic in both public and private datasets using the WEKA and Scikit-learning platforms. The classification experiments were conducted with detailed explanations and interpretations of the model performance reports, ensuring a thorough understanding of the process. The performance of the model was further improved using feature selection. The final models were tested on previously unseen data to evaluate their forecasting accuracy in a real-world scenario. The next section discusses the research findings and the comparison to the other approaches based on traditional ML and DL.

5.6 Machine Learning Analysis Findings

5.6.1 Research Finding

The primary objective of this chapter was to demonstrate the effectiveness of identifying Tor traffic using a single payload through our character analysis approach via ML. In this study, the classification performance was evaluated using various metrics, including accuracy, precision, recall, F1 score, and AUC/ROC scores on both WEKA and Scikit-learn platforms. The results revealed that the average accuracies obtained were 93.87% and 95.81%, with all other evaluation metrics achieving average scores of around 94%. Moreover, we investigated the improvement of model performance by implementing a feature selection process that identified only the most useful features. Our findings showed that by selecting only relevant features, the average accuracy improved from 93.96% to 98.61%, resulting in increased accuracy and reduced training time. Lastly, the final models were evaluated using previously unseen data to verify the reliability and generalisability of the approach. The results indicated that the accuracy of predicting with the final model, when applied to new unseen data, exceeded 90% across all nine applications, with an average accuracy of 98.06%. These results substantiated the effectiveness of our approach in detecting Tor traffic across various applications within both public and private datasets. This provided evidence against the null hypothesis H_{02} that a single encrypted payload cannot be used to identify Tor traffic, while also addressing research question *Q2* by demonstrating the efficient distinction of Tor traffic using the proposed approach.

The findings from this study can be summarised in terms of four key elements:

1) Features: To examine the robustness of cryptography when deployed in real-world scenarios, we utilised features derived from hex character statistics analysis, organising them into three distinct feature sets (Set 1 - an absolute frequency set, Set 2 - a frequency ratio set, and Set 3 - a combination of Sets 1 and 2). When comparing the different feature sets, Set 3 outperformed Sets 1 and 2 for all models in Scikit-learn, except for IBk in WEKA, indicating that integrating features from absolute frequency and frequency

ratio can improve the models. The performance of Sets 1 and 2 varied depending on the algorithms and binary classes, with some applications showing only slight differences in accuracy between the two sets, while in others, there was no significant difference between the two sets. However, to examine the robustness of cryptography when deployed in real-world scenarios, ciphertext must be of the same length. As a result, we focused on Set 2, which was unaffected by the encrypted payload length. The findings demonstrated that Set 2 can be used to classify Tor traffic with average accuracies of 93.87% and 95.81% across both platforms.

2) Datasets: To achieve consistent results and network independence, we conducted classification experiments on two dataset sources, containing nine binary classes, which included eight from the public dataset and one from the private dataset. In both public and private datasets, the accuracy results displayed a similar trend of high accuracy. For instance, in WEKA, the accuracy scores of the public dataset on Set 2 ranged from 88.99% to 99.63% in eight applications, while the private dataset achieved 97.12%. Consequently, the use of nine binary classes from both dataset sources indicated consistent results and network independence.

3) Algorithms: To ensure accurate and valid outcomes, we utilised three effective supervised learning algorithms, including DTs-based, RFs-based, and kNN-based algorithms. For Sets 1 and 3, all algorithms demonstrated similar performance levels. In contrast, for Set 2, the performances of DTs-based, RFs-based algorithms were quite similar, except for kNN, which experienced a significant decline of approximately 23% on average.

4) Platforms: To guarantee the reliability and confidence of our findings, we performed classification using both WEKA and Scikit-learn platforms. During the experiments, we found that WEKA was useful for quickly viewing preliminary results and conducting initial analyses with just a few clicks. However, it may have limited flexibility for customising or modifying algorithms. On the other hand, a Python-based script utilising the Scikit-learn library provided us with greater flexibility to customise and adapt to our specific needs, as well as more efficient data preprocessing and classification tasks. Overall, both platforms have unique advantages and disadvantages, and the choice

of which to use depends on the specific needs of each analysis.

The efficiency of our approach compared to traditional approaches can also be seen in the fact that it requires only a single packet to accurately identify Tor traffic. While traditional approaches that rely on time-based features may require the calculation of several packets to achieve high accuracy rates, our approach can achieve accurate identification with only a single packet. This not only saves processing time but also reduces the risk of misidentification due to unreliable time-based features. This approach is particularly useful in situations where time-based features are unreliable or unavailable due to the asymmetric nature of Internet routing, traffic aggregation, or modification of packet length in tunnels or proxies (Rezaei & Liu, 2019). The use of character analysis features can also provide a quick and efficient way to identify Tor traffic with high accuracy rates.

5.6.2 Comparison

Traditional Machine Learning

A comparison to previous works is given in order to compare the performance of our proposed approach to that of others. The public dataset we used during the research is known as the ISCXTor2016 dataset, which was created by researchers in Canada. One of their objectives (Lashkari et al., 2017) is to classify Tor traffic using time-based features. Their proposed approach achieved both precision and recall scores of 0.99 from flow-timeout value 120. The same dataset and flow-based features were employed by Cuzzocrea et al. (2017), in conjunction with different classifiers. Their results showed that JRip provided the best performance of 100% for both precision and recall when differentiating between Tor and nonTor traffic. Another recent work from Sarwar et al. (2021) used ISCXTor2016 to detect darknet traffic. They found that CNN-LSTM produced the best performance evaluation metrics of precision, recall, and F1 score, which were 0.97, 0.95, and 0.96, respectively.

Our approach achieved the highest scores of precision and recall scores of 0.99 from Browsing. When compared to the above works, our approach produced the same model performance as Lashkari et al. (2017), slightly higher than Sarwar et al. (2021), and

slightly lower than Cuzzocrea et al. (2017)'s approach. However, the features proposed by these three works are calculated from multiple flow packets, as opposed to our approach, which requires only one packet to identify Tor. This could be advantageous because, rather than computing the number of packets, only a single packet is required to detect Tor traffic, thereby consuming fewer computer resources. Aside from time-based features, which require multiple packets to generate and inevitably require capturing packets from the start of the connection, our approach is independent of the network flow's location. However, it should be noted that our proposed approach, while highly effective in identifying Tor traffic, is limited to detecting it based on packet characteristics only, without taking into account other factors such as network behaviour or user behaviour. Therefore, future research could explore combining our approach with other detection techniques to achieve more comprehensive and accurate results.

Deep Learning

NTC has increasingly incorporated DL techniques, which, in certain contexts, have shown promise in outperforming some classical ML models in accuracy. One of the prominent advantages of using deep learning models, particularly in domains like image and speech recognition, is their ability to automate feature extraction, potentially reducing the need for manual feature engineering that is often required in traditional machine learning approaches (Shaheen et al., 2016). In our study, we sought to determine if DL could outperform traditional ML techniques. To test this, we carried out Tor traffic classification using encrypted payloads as input for the DL model, leveraging the Keras[1] library with the Tensorflow[2] backend. In this experiment, we used the ISCXTor2016 dataset, identical to that in our study. We employed a ML model with a CNN architecture that includes a Conv1D layer, a Flatten layer, and a Dense output layer. This design aims to achieve a balance between model complexity and performance, all while ensuring efficient computational use. The Dense output layer uses a sigmoid activation function, reflecting for binary classification, differentiating between Tor and nonTor traffic. Our model processes input features R_0-R_f, corresponding to the

[1] https://keras.io/
[2] https://www.tensorflow.org/

labels of Tor and nonTor class labels. We divided the dataset into 60% for training, 20% for validation, and 20% for testing.

The model was trained for ten epochs, with a batch size of 32 and binary cross-entropy loss function. During training, we also utilised a validation set to measure the model's performance, ensuring it didn't overfit to the training data. After training, we assessed the model on a separate testing set, achieving an accuracy of 81%. This result suggests that DL can be used for Tor traffic classification using encrypted payload as input. Throughout the training phase, we observed the training and validation loss curves. These curves provide insights into the model's learning process. Ideally, both curves should decrease to a point of stability with a minimal gap between them. If the training loss is much lower than the validation loss, it suggests overfitting, meaning the model might perform poorly on new, unseen data. Conversely, if both the training and validation loss are high, it may indicate underfitting, implying that the model is not even fitting the training data properly. In our observations, both curves consistently decreased, indicating that our model achieved a good balance. This suggests the model generalised well to the validation set without significant overfitting or underfitting.

In our research, traditional ML outperformed DL, particularly the character analysis approach which achieved accuracy rates of 97.95% with Scikit-Learn and 99.03% with WEKA in browsing application traffic. The ML approach, focusing on handcrafted features, not only provided superior accuracy but also gave valuable insights into the network traffic differentiation, specifically between Tor and nonTor traffic. Decision trees, for example, offered clear transparency in feature selection from *R_ 0-R_f*, enabling understanding. In contrast, DL, despite its potential, presented challenges. The CNN we used still depended on handcrafted features but lacked the explanatory depth that ML models provided (Ismailaj et al., 2021). Furthermore, DL's computational and data demands make it less ideal for real-time packet detection, where speed and efficiency are paramount. Given these factors, traditional ML was the clear choice for our research objectives.

Therefore, while DL models can be effective in network traffic classification, our findings suggest that traditional ML models may be more suitable for certain types

of tasks, particularly when the goal is to gain insights into the relationships between features and their impact on the output. In the case of Tor traffic classification using character analysis, DTs-based algorithms are the most efficient option.

5.7 Summary

In this chapter, we detailed experiments to classify Tor traffic using public and private datasets and two different ML platforms. We demonstrated the classification process and discussed how to interpret the performance reports. To improve model performance, we implemented feature selection, which resulted in higher accuracy rates. The final models were also tested on previously unseen data to evaluate their performance in a real-world scenario. The chapter concludes with a discussion of the results and their implications for network traffic classification. In addition to presenting the ML approach, we also compared our findings to other traditional features such as time-based features. Furthermore, we presented preliminary results from our DL experiments using the same dataset and approach. While the results showed that DL can be effective for Tor traffic classification using encrypted payload as input, our findings suggest that traditional ML models may be more suitable for certain types of tasks like classifying Tor traffic using character analysis.

The next chapter will serve as the final chapter, in which we summarise the main findings of the study and discuss their contributions to the field as well as their implications for practice and research. We will also provide suggestions for future research based on the limitations and opportunities identified in the study.

Chapter 6

Conclusion and Future Work

In this concluding chapter, we provide a comprehensive summary of the key findings and contributions made through this research. The primary focus has been to develop and evaluate a novel approach for classifying encrypted network traffic, specifically differentiating between Tor and nonTor traffic. This chapter begins by revisiting the motivation behind our study and summarising the main results obtained from our extensive experimentation. We then discuss the implications of our findings, addressing the impact of our work on the field of network security and encrypted traffic analysis. Additionally, we acknowledge the limitations and challenges encountered during our research and suggest potential avenues for future work in this area. Overall, this chapter serves to consolidate our research efforts, emphasising the significance of our contributions and outlining the path forward for the continued advancement of encrypted traffic classification techniques.

6.1 Overview

This study investigated encrypted payload characteristics in both Tor and nonTor networks using character analysis, aiming to propose a novel Tor traffic classification approach. The research utilised statistical and ML techniques to analyse the hex characters within the ciphertexts.

While Tor is renowned for providing online anonymity, its use for illicit activities mo-

tivates researchers to seek accurate methods for detecting Tor traffic. NTC is the process of identifying and categorising network traffic using various methods to provide services such as QoS, billing, and anomaly detection. Traditional traffic classification methods, such as port-based classification, have been replaced by payload-based classification using DPI. However, the increasing use of encryption in network communication to ensure security and privacy have made traffic classification more difficult, as packet inspection is not possible with encrypted packets. ML approaches, such as supervised learning and DL, are now being utilised to classify network traffic. Although DL models that do not require handcrafted features have higher accuracy rates than supervised learning models, handcrafted features remain essential. The selection of features used in model training is crucial, as a poor choice of input features can negatively impact classification quality. Due to encryption, current trends in traffic classification are focused on flow-based characteristics of packets, especially in the case of Tor traffic. In many studies that used supervised learning for classification, Tor traffic detection relied primarily on the flow characteristics of packets, i.e., their time-based properties. This method depends on the fact that Tor packets travel longer distances through multiple proxy servers located worldwide to reach their destination than nonTor packets. However, this method requires many packets to generate the features, and the asymmetric nature of Internet routing may affect model performance.

Considering these limitations, it is important to explore alternative approaches for Tor traffic classification. One key aspect that sets Tor apart from nonTor networks is its packet routing mechanism. Specifically, Tor employs a multilayer encryption approach, unlike the single-layer encryption used in nonTor networks. This difference in encryption methods may provide a basis for novel classification techniques that overcome the drawbacks associated with relying solely on time-based properties. While encryption serves to secure data, weak cryptographic algorithms can result in compromised security. Consequently, an ideal encryption scheme incorporates key principles of modern cryptography design to ensure robust security, preventing ciphertexts of the same length from revealing any additional information about the underlying plaintext message, regardless of the attacker's knowledge. However, this theoretical security does not always guarantee

real-world security, particularly in computer networking. Therefore, investigating the characteristics of encrypted payloads may reveal unique insights, potentially contributing to the development of novel classification techniques for Tor traffic.

In light of this, statistical computational approaches, which have been successful in text classification, have yet to be tested in the context of computer networks. Tor encrypts data using the TLS protocol, leading us to expect that the hex characters appearing in the payload should exhibit similar characteristics to other TLS traffic and to traffic in other encryption scenarios. However, Tor's multilayer encryption process may influence the distribution of data in the payload differently compared to the single-layer encryption used in nonTor communication. This difference prompts us to question whether Tor and nonTor encrypted traffic should appear similar in terms of character analysis. To address these issues, we formulated our first null hypothesis H_{01} as follows: "There is no difference between Tor and nonTor traffic in terms of encrypted payloads". We tested this hypothesis by answering the first research question *Q1*, "Can we distinguish Tor from nonTor traffic based on their encrypted payload?". Our analysis supported the rejection of the first null hypothesis, leading us to test the second null hypothesis H_{02}, "A single encrypted payload cannot be used to identify Tor traffic", by exploring the second research question *Q2*, "Can we distinguish Tor from nonTor traffic using the encrypted payload in a data-efficient manner?". To rigorously examine both research questions and hypotheses, we extracted hex statistics-based features from the ciphertext of Tor and nonTor traffic.

6.2 Research Findings

This study introduces a novel approach that challenges conventional assumptions in the field of encryption theory. Our research was conducted with scientific rigour, utilising appropriate methodologies to ensure the validity of our arguments to address our research objectives. Data used was drawn from two different sources. One source was publicly available repositories, which provided a vast and diverse range of sample traffic data suitable for addressing our research questions. This reliable data was complemented by self-collected data to ensure the independence of the network environment during

analysis, thereby reinforcing the validity of our results. In order to ensure consistent results, datasets from both sources were transformed into nine binary classes for analysis. However, a significant challenge in data preprocessing was noise exclusion, as traffic traces often contained a mixture of packets and protocols running in the networking media. To isolate the relevant encrypted payload, it was essential to thoroughly examine the nature of each individual application and its handling of packet transport. This process laid the foundation for developing precise packet-filtering commands, ensuring accurate encrypted payload extraction for further analysis. Below are our key findings aligned with the four research objectives:

1. CHARSTAT: Our primary objective was to develop an automated method for extracting features based on hex character statistics from both Tor and nonTor public and private datasets. To achieve this, we utilised our custom-written Python script, CHARSTAT, available in Appendix A.1 and openly accessible[1] to extract encrypted payloads from raw traffic data. This resulted in four distinct feature sets based on hex character statistics. These features, extracted from the encrypted payload, encompass: (1) the frequency of hex characters (F_0-F_f), representing raw hex values;(2) the ratio of hex character frequencies (R_0-R_f), reflecting raw hex values;(3) the total character count (T), indicating the payload size;(4) and entropy (E), measuring the randomness of hex values in ciphertexts. The output of CHARSTAT served as input for both statistical analysis and classification, playing a critical role in our research.

2. Statistical Analysis: Our second objective focused on examining the differences in hex character statistics between Tor and nonTor encrypted payloads based on statistical methods for understanding their underlying statistical patterns. To this end, we undertook a comprehensive statistical analysis comparing four feature sets: (1) F_0-F_f, (2) R_0-R_f, (3) T, and (4) E, across nine application types for both Tor and nonTor encrypted payloads. We leveraged both descriptive and inferential statistics to comprehensively address the first research question and the hypothesis H_{01}, which assumes that Tor and nonTor encrypted payloads have identical characteristics.

Descriptive statistics revealed notable differences between Tor and nonTor traffic in

[1] https://github.com/pitpimon/Tor-traffic-classification/blob/main/charstat.py

terms of hex character frequency, total characters, and entropy values. These observations can be attributed to Tor's use of fixed-size cells of 512 bytes and unique encryption methods, as opposed to the variable packet lengths and diverse encryption mechanisms employed in nonTor applications. In contrast, the hex character frequency ratio exhibited a very similar pattern in both Tor and nonTor applications, with only slight differences observed in some cases, which may be due to the effect of normalisation.

Overall, The characteristics of Tor traffic varied considerably from nonTor traffic in most attributes, except for the ratio features. Using the Mann-Whitney U test on four attribute sets, we found that 292 out of 306 features showed significant differences between Tor and nonTor payloads, leading to a 95.42% differentiation. This outcome led to the rejection of the null hypothesis H_{01} that the two types of packets have similar character frequencies. Consequently, our findings address research question *Q1*, demonstrating that it is certainly possible to distinguish Tor from nonTor traffic based on their encrypted payloads.

3. ML Analysis: The third objective aimed to assess the capability of classifying Tor and nonTor traffic using features from hex character statistics via ML techniques. To address this, we experiment with multiple ML models, including DTs-based, RFs and kNN-based algorithms. These experiments were conducted on both the WEKA and Scikit-learn platforms. We processed data from two sources (eight application types from public sources and one from a private dataset), resulting in nine binary classification groups.

To examine the robustness of cryptography in real-world scenarios, we focused on hex character frequency and hex character frequency ratio features, organising them into three distinct feature sets (Set 1 - F_0-F_f), Set 2 - R_0-R_f), and Set 3 - a combination of Sets 1 and 2). In our analysis using Scikit-learn, Set 3 consistently surpassed the performance of Sets 1 and 2 for all models, except for IBk in WEKA, indicating that integrating features from Set 1 and Set 2 can improve the model performance. However, the effectiveness of Sets 1 and 2 varied depending on the algorithms and binary classes, with some applications showing only slight differences in accuracy between the two sets, while in others, there was no significant difference. A key consideration for analysing the

robustness of cryptography in real-world settings is the consistency of ciphertext length. Hence, we focused on Set 2, which was unaffected by the encrypted payload length. The findings demonstrated that Set 2 effectively classified Tor traffic using DTs-based algorithms, achieving average accuracies of 93.87% and 95.81% across both platforms, with other evaluation metrics yielding similar scores.

We further enhanced our model's performance by employing the top-performing DTs-based model, using Set 2, and conducting a feature selection process to identify the most useful features. By selecting only relevant features, the average accuracy improved from 93.96% to 98.61%, resulting in increased accuracy and reduced training time. When evaluating the final models using previously unseen data, the accuracy of predicting with the final model exceeded 96% across all nine applications, with an average accuracy of 98.61%. The obtained results validated the efficacy of our approach in identifying Tor traffic across diverse applications in both public and private datasets. These evidences disprove the null hypothesis H_{02} stating that a single encrypted payload cannot be used to recognise Tor traffic. Additionally, it addresses research question *Q2*, confirming that the encrypted payload can differentiate Tor traffic in a data-efficient manner.

4. A Novel Payload Size-Independent Approach: Our fourth objective aimed to introduce a unique payload size-independent method for effective Tor and nonTor traffic differentiation. To achieve this, we explored the correlation between features and payload size to ensure the proposed methodology is robust against payload size variations. Through the Pearson correlation coefficient tests, we identified a significant correlation between the variables. However, most correlation coefficients were proximate to zero, primarily falling between 0.23 and 0.00. This denotes a marginal positive correlation. Thus, despite a statistically significant relationship between the T and R_0-R_f features, their actual correlation strength is insubstantial. Consequently, R_0-R_f emerges as a reliable payload size-independent feature for Tor traffic classification.

Although our findings may contradict the encryption theory assumption that a ciphertext should not leak any additional information about the underlying plaintext message, our approach does not reveal the message's content from the ciphertext. Instead, it distinguishes Tor and nonTor traffic based on their unique characteristics. Several

reasons may explain why our hex character statistics analysis approach is effective in identifying encrypted traffic:

1. NonTor networks implement various encryption algorithms and parameters across a multitude of applications, each utilising a distinct encryption protocol. In contrast, the Tor network's packets are generated by a uniform encryption algorithm, creating a more homogeneous traffic pattern in Tor compared to nonTor networks.

2. The Tor network employs a uniform packet structure, wherein every packet is consistently sized at 512 bytes. This standardized size results in more uniform packet splitting patterns, distinguishing Tor traffic from nonTor traffic, which typically displays more variability in packet sizes.

3. In Tor, varying packet sizes are first split into 512-byte fixed-size cells at the application layer, and then encrypted into Tor segments. Meanwhile, in nonTor networks, varying packet sizes are first encrypted and then subjected to data splitting, which could involve either segmentation (at the TCP layer) or fragmentation (at the IP layer). The order of packet splitting could affect the uniformity of the ciphertext.

Any or all of these proposed explanations could contribute to the distinguishable characteristics of their encrypted traffic. It is essential to acknowledge that the proposed explanations for the effectiveness of our hex character statistics analysis approach are based on plausible theories, but they remain assumptions at this stage. Further research is required to corroborate these hypotheses and enhance our understanding of the differences between Tor and nonTor traffic. Importantly, the insights derived from our study could also highlight potential channels to exploit vulnerabilities within the Tor network. By revealing these areas of concern, Tor developers might find the information valuable in their ongoing efforts to improve the security and robustness of the network.

6.3 Research Contributions and Impact of Work

Our research offers a significant contribution to the field of network security by presenting a novel and effective approach for detecting Tor traffic that is privacy-preserving and does not require the decryption of packets or knowledge of cryptographic keys. The following is a list of our main contributions:

1. We have developed a novel approach for detecting Tor traffic based on hex character statistics analysis of encrypted traffic, involving the computation and analysis of hex character statistics in the ciphertext. To support this method, which does not require packet decryption or knowledge of cryptographic keys, we also created CHARSTAT, a versatile tool that computes and generates classification features based on encrypted payloads and can be used for other types of character analysis tasks.

2. Our novel approach requires only one packet to determine whether the traffic originated from Tor or nonTor networks. This represents a significant improvement compared to conventional methods, which usually require multiple packets for accurate classification.

3. Our novel approach requires only a single packet to determine whether the traffic originated from Tor or nonTor networks. This independence from sequential packet analysis allows us to examine packets from any location within network traces, eliminating the necessity to analyse packets in a specific order. Such flexibility enhances the versatility of our method in detecting Tor activity.

4. While our approach is to inspect encrypted network traffic to differentiate packets between Tor and nonTor networks, it does not decrypt or access the actual content of the encrypted payload, thereby ensuring privacy preservation. Our approach accommodates the growing concern for user privacy in the digital age.

As a direct result of our contributions, our work significantly impacts enhanced monitoring. Particularly, organisations involved in cybersecurity can employ our method

to inspect Tor traffic with greater efficiency. This is especially beneficial in detecting malicious activities that use the Tor network as a hiding place. Specifically, our approach offers a substantial tool for both law enforcement and cybersecurity investigations, contributing to the reduction of cybercrime. Moreover, the precision of our traffic classification not only increases network security but also enhances its performance, leading to a more secure and optimised digital environment.

6.4 Challenges

In spite of the promising results achieved by our proposed approach, there were several challenges that we encountered during our study. These challenges need to be taken into account to ensure the generalisability and validity of our findings.

1. **Incompatibility of Classifiers Across Different Network Environments:** The two final models from the different contexts were found to be incompatible with each other, as discussed in Section 5.5. This means that the final model from VoIP cannot accurately classify Tor traffic in Browsing or other applications. Therefore, it is crucial to build a new model based on the specific sample data from each new network environment. Prior to implementation, it is advised to retrain the model for the new computer network context. This phenomenon aligns with the suggestion made by Rezaei and Liu (2019) that a model trained on a dataset collected at a single capturing point may not be effective when applied at a different capturing point.

2. **Processing Time Consumption:** DPI has the disadvantage of taking a long time to examine traffic, as it must scan each individual packet. Our approach also requires packet inspection, which can be time-consuming. However, with the prevalence of supercomputers, packet investigation can be speeded up. Furthermore, we can improve our promising accurate detection approach by investigating light-weight packet detection, such as selecting a few packets from the target flows after the TLS connection has been terminated, rather than scanning every packet that contains payload data. This alternative may allow us to overcome the overhead on

inspection time.

3. **Theoretical Assumption in Hex Character Statistics Analysis Approach:** One limitation of this study is the proposed explanations for the effectiveness of our hex character statistics analysis approach in distinguishing Tor and nonTor traffic. Although we have suggested several plausible reasons, such as differences in encryption layers, encryption algorithms, and packet uniformity, these explanations are theoretical and require further investigation to be confirmed. Future research should aim to verify these hypotheses and enhance our understanding of the differences between Tor and nonTor traffic. This could involve designing and conducting experiments to test the specific impact of encryption layers, algorithms, and packet properties on the frequency distribution of characters in encrypted payloads. By addressing these gaps in our knowledge, we can contribute to the development of more robust and accurate methods for identifying and classifying encrypted network traffic.

6.5 Future Work

While our study has provided a useful understanding of the classification of Tor and nonTor encrypted traffic using hex character statistics analysis, there are several routes for future research that could build upon our findings. Exploring these areas, we believe, would advance the field of network traffic analysis and improve our ability to detect and classify encrypted traffic in a variety of contexts.

1. **Enhancing compatibility of classifiers:** Investigate methods to improve the compatibility of classifiers across different network environments. This could involve developing adaptive models that can adjust to new contexts or exploring transfer learning techniques to leverage knowledge from one domain to another.

2. **Expanding the scope of traffic analysis:** Extend the research to other types of encrypted network traffic beyond Tor and nonTor, such as those generated by different VPN protocols, anonymity networks, or other encryption technologies.

This would provide a more comprehensive understanding of encrypted traffic and contribute to the development of more robust classification methods.

3. **Optimising packet inspection techniques:** Explore the development of more efficient packet inspection methods to reduce the time consumption associated with traffic analysis. This could include investigating lightweight packet detection techniques, such as sampling strategies or ML approaches that can quickly identify relevant packets in the traffic flow.

4. **Incorporating additional features:** Investigate the potential benefits of incorporating additional features, such as packet timing or flow-based features, in the classification process. Combining multiple features could lead to more accurate and reliable classification models.

5. **Validating theoretical assumptions:** Conduct experiments to test the impact of encryption layers, algorithms, and packet properties on the frequency distribution of characters in encrypted payloads. This would help confirm or refute the theoretical conjectures proposed in this study and provide a better understanding of the differences between Tor and nonTor traffic.

Bibliography

Abdullah, A. M., et al. (2017). Advanced Encryption Standard (AES) algorithm to encrypt and decrypt data. *Cryptography and Network Security*, *16*, 1–11.

Adami, D., Callegari, C., Giordano, S., Pagano, M., & Pepe, T. (2009). A real-time algorithm for skype traffic detection and classification. In S. Balandin, D. Moltchanov, & Y. Koucheryavy (Eds.), *Smart spaces and next generation wired/wireless networking* (pp. 168–179). Springer Berlin Heidelberg.

Aggarwal, C. C. (2018). *Machine learning for text* (Vol. 848). Springer.

Aggarwal, C. C., et al. (2018). Neural networks and deep learning. *Springer*, *10*, 978–3.

Alenezi, M. N., Alabdulrazzaq, H., & Mohammad, N. Q. (2020). Symmetric encryption algorithms: Review and evaluation study. *International Journal of Communication Networks and Information Security*, *12*(2), 256–272.

Ali, H., Salleh, M. N. M., Hussain, K., Ahmad, A., Ullah, A., Muhammad, A., Naseem, R., & Khan, M. (2019). A review on data preprocessing methods for class imbalance problem. *International Journal of Engineering & Technology*, *8*, 390–397.

Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., & Kochut, K. (2017). A brief survey of text mining: Classification, clustering and extraction techniques. *arXiv preprint arXiv:1707.02919*.

Allua, S., & Thompson, C. B. (2009). Inferential statistics. *Air Medical Journal*, *28*(4), 168–171.

AlSabah, M., Bauer, K., & Goldberg, I. (2012). Enhancing Tor's performance using real-time traffic classification. *Proceedings of the 2012 ACM conference on Computer and communications security*, 73–84.

AlSabah, M., & Goldberg, I. (2016). Performance and security improvements for tor: A survey. *ACM Computing Surveys (CSUR)*, *49*(2), 1–36.

Annessi, R. (2014a). *Lower-latency anonymity* [Doctoral dissertation, Vienna University of Technology].

Annessi, R. (2014b). *Lower-latency anonymity latency reduction in the Tor network using circuit-level round-trip-time measurements* [Master's thesis, Vienna University of Technology].

Azab, A., Khasawneh, M., Alrabaee, S., Choo, K.-K. R., & Sarsour, M. (2022). Network traffic classification: Techniques, datasets, and challenges. *Digital Communications and Networks*.

Azfar, A., Choo, K.-K. R., & Liu, L. (2016). Android mobile voip apps: A survey and examination of their security and privacy. *Electronic Commerce Research*, *16*(1), 73–111.

Bai, X., Zhang, Y., & Niu, X. (2008). Traffic identification of Tor and web-mix. *2008 Eighth International Conference on Intelligent Systems Design and Applications*, *1*, 548–551.

Bakhshi, T., & Ghita, B. (2016). On internet traffic classification: A two-phased machine learning approach. *Journal of Computer Networks and Communications*, *2016*.

Barker, J., Hannay, P., & Szewczyk, P. (2011). Using traffic analysis to identify the second generation onion router. *2011 IFIP 9th International Conference on Embedded and Ubiquitous Computing*, 72–78.

Barrett, D. J., Silverman, R. E., & Byrnes, R. G. (2009). *Ssh, the secure shell: The definitive guide* (2nd). O'Reilly Media, Inc.

Behrouz, A. F. (2022). *Data communications and networking* (5th ed.). The McGraw-Hill Companies, Inc.

Benamira, A., Gerault, D., Peyrin, T., & Tan, Q. Q. (2021). A deeper look at machine learning-based cryptanalysis. *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 805–835.

Benesty, J., Chen, J., Huang, Y., & Cohen, I. (2009). Pearson correlation coefficient. In *Noise reduction in speech processing* (pp. 1–4). Springer.

Bhargava, N., Sharma, G., Bhargava, R., & Mathuria, M. (2013). Decision tree analysis on j48 algorithm for data mining. *Proceedings of International Journal of Advanced Research in Computer Science and Software Engineering*, *3*(6).

Bhavsar, P., Safro, I., Bouaynaya, N., Polikar, R., & Dera, D. (2017). Machine learning in transportation data analytics. In *Data analytics for intelligent transportation systems* (pp. 283–307). Elsevier.

Bijalwan, V., Kumar, V., Kumari, P., & Pascual, J. (2014). KNN based machine learning approach for text and document mining. *International Journal of Database Theory and Application*, *7*(1), 61–70.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (2017). *Classification and regression trees*. Routledge.

Cao, Z., Xiong, G., Zhao, Y., Li, Z., & Guo, L. (2014). A survey on encrypted traffic classification. *International Conference on Applications and Techniques in Information Security*, 73–81.

Cardenas-Haro, J. A., & Dawson, M. (2017). Tails linux operating system: The amnesiac incognito system in times of high surveillance, its security flaws, limitations, and strengths in the fight for democracy. In *Security solutions for hyperconnectivity and the internet of things* (pp. 260–271). IGI Global.

Chakravarty, S., Barbera, M. V., Portokalidis, G., Polychronakis, M., & Keromytis, A. D. (2014). On the effectiveness of traffic analysis against anonymity networks using flow records. In M. Faloutsos & A. Kuzmanovic (Eds.), *Passive and active measurement* (pp. 247–257). Springer International Publishing.

Chakravarty, S., Portokalidis, G., Polychronakis, M., & Keromytis, A. D. (2011). Detecting traffic snooping in Tor using decoys. *International Workshop on Recent Advances in Intrusion Detection*, 222–241.

Chawla, N. V. (2009). Data mining for imbalanced datasets: An overview. *Data mining and knowledge discovery handbook*, 875–886.

Cheng, R.-G., Tsai, P.-Y., Lin, P.-C., Lian, C.-L., Liu, T.-H., Chou, H.-C., Tsao, S.-L., & Yang, S. (2013). Design and implementation of a skype protocol analyzer. *2013 IEEE Conference on Communications and Network Security (CNS)*, 375–376.

Creswell, J. W., & Creswell, J. (2018). *Research design* (5th ed.). Thousand Oaks, CA: Sage.

Croll, G. J. (2013). Bientropy-the approximate entropy of a finite binary string. *arXiv preprint arXiv:1305.0954*.

Crotti, M., Gringoli, F., & Salgarelli, L. (2009). Impact of asymmetric routing on statistical traffic classification. *GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*, 1–8.

Cuzzocrea, A., Martinelli, F., Mercaldo, F., & Vercelli, G. (2017). Tor traffic analysis and detection via machine learning techniques. *2017 IEEE International Conference on Big Data (Big Data)*, 4474–4480.

Dainotti, A., Pescapé, A., & Sansone, C. (2011). Early classification of network traffic through multi-classification. *International Workshop on Traffic Monitoring and Analysis*, 122–135.

Dang, N. C., Moreno-García, M. N., & De la Prieta, F. (2020). Sentiment analysis based on deep learning: A comparative study. *Electronics*, *9*(3), 483.

Das, B., & Turkoglu, I. (2018). A novel numerical mapping method based on entropy for digitizing dna sequences. *Neural Computing and Applications*, *29*(8), 207–215.

Datta, J., Kataria, N., & Hubballi, N. (2015). Network traffic classification in encrypted environment: A case study of google hangout. *2015 twenty first national conference on communications (NCC)*, 1–6.

Dehghani, F., Movahhedinia, N., Khayyambashi, M. R., & Kianian, S. (2010). Real-time traffic classification based on statistical and payload content features. *2010 2nd international workshop on intelligent systems and applications*, 1–4.

Deng, Z., Zhu, X., Cheng, D., Zong, M., & Zhang, S. (2016). Efficient kNN classification algorithm for big data. *Neurocomputing*, *195*, 143–148.

Dewey, J. (1933). *How we think: A restatement of the relation of reflective thinking to the educative process*. Houghton Mifflin.

Diehl, E. (2012). *Securing digital video: Techniques for drm and content protection.* Springer Science & Business Media.

Dierks, T., & Allen, C. (1999, January). The TLS Protocol Version 1.0 [Accessed: 2021-03-15]. https://www.ietf.org/rfc/rfc2246.txt

Dierks, T., & Rescorla, E. (2006, April). The Transport Layer Security (TLS) Protocol Version 1.1 [Accessed: 2021-03-15]. https://datatracker.ietf.org/doc/html/rfc4346

Dierks, T., & Rescorla, E. (2008, August). The Transport Layer Security (TLS) Protocol Version 1.2 [Accessed: 2021-03-15]. https://datatracker.ietf.org/doc/html/rfc5246

Dingledine, R., & Mathewson, N. (2022). *Tor protocol specification.* Retrieved August 1, 2010, from https://github.com/torproject/torspec/blob/main/tor-spec.txt

Dingledine, R., Mathewson, N., & Syverson, P. (2004). *Tor: The Second-Generation Onion Router* (tech. rep.). Naval Research Lab Washington DC.

Dong, G., & Liu, H. (2018). *Feature engineering for machine learning and data analytics.* CRC Press.

Draper-Gil, G., Lashkari, A. H., Mamun, M. S. I., & Ghorbani, A. A. (2016). Characterization of encrypted and vpn traffic using time-related. *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, 407–414.

Dulock, H. L. (1993). Research design: Descriptive research. *Journal of Pediatric Oncology Nursing*, *10*(4), 154–157.

Dutta, N., Jadav, N., Tanwar, S., Sarma, H. K. D., & Pricop, E. (2022a). Being hidden and anonymous. In *Cyber security: Issues and current trends* (pp. 17–36). Springer.

Dutta, N., Jadav, N., Tanwar, S., Sarma, H. K. D., & Pricop, E. (2022b). Tor—the onion router. In *Cyber security: Issues and current trends* (pp. 37–55). Springer.

Eskin, E., Arnold, A., Prerau, M., Portnoy, L., & Stolfo, S. (2002). A geometric framework for unsupervised anomaly detection. In *Applications of data mining in computer security* (pp. 77–101). Springer.

Faizan, M., & Khan, R. A. (2019). Exploring and analyzing the dark web: A new alchemy. *First Monday.*

Falck, F., Marstaller, J., Stoehr, N., Maucher, S., Ren, J., Thalhammer, A., Rettinger, A., & Studer, R. (2020). Measuring proximity between newspapers and political parties: The sentiment political compass. *Policy & internet*, *12*(3), 367–399.

Fall, K. R., & Stevens, W. R. (2012). *TCP/IP illustrated, volume 1: The protocols* (2nd ed.). addison-Wesley.

Field, A. (2013). *Discovering statistics using ibm spss statistics*. sage.

Fisher, M. J., & Marshall, A. P. (2009). Understanding descriptive statistics. *Australian critical care*, *22*(2), 93–97.

Fletcher, A. J. (2017). Applying critical realism in qualitative research: Methodology meets method. *International journal of social research methodology*, *20*(2), 181–194.

Fu, X., Ling, Z., Luo, J., Yu, W., Jia, W., & Zhao, W. (2009). One cell is enough to break Tor's anonymity. *Proceedings of Black Hat Technical Security Conference*, 578–589.

Gancarczyk, G., Dąbrowska-Boruch, A., & Wiatr, K. (2011). Statistics in cyphertext detection. *Prace Instytutu Elektrotechniki*, 67–85.

Géron, A. (2022). *Hands-on machine learning with scikit-learn, keras, and tensorflow*. " O'Reilly Media, Inc."

Gjorgjievska Perusheska, M., Dimitrova, V., Popovska-Mitrovikj, A., & Andonov, S. (2021). Application of machine learning in cryptanalysis concerning algorithms from symmetric cryptography. In *Intelligent computing* (pp. 885–903). Springer.

Gogoi, P., Bhuyan, M. H., Bhattacharyya, D., & Kalita, J. (2012). Packet and flow based network intrusion dataset. *IC3*.

Gómez, S. E., Hernández-Callejo, L., Martínez, B. C., & Sánchez-Esguevillas, A. J. (2019). Exploratory study on class imbalance and solutions for network traffic classification. *Neurocomputing*, *343*, 100–119.

Goralski, W. (2017). *The illustrated network: How tcp/ip works in a modern network* (2nd ed.). Morgan Kaufmann.

Granerud, A. O. (2010). *Identifying TLS abnormalities in Tor* [Master's thesis, Gjøvik University College].

Greven, A., Keller, G., & Warnecke, G. (2014). *Entropy* (Vol. 47). Princeton University Press.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. (2018). A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, *51*(5), 1–42.

Guitton, C. (2013). A review of the available content on Tor hidden services: The case against further development. *Comput. Hum. Behav.*, *29*, 2805–2815.

Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003). KNN model-based approach in classification. *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, 986–996.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The weka data mining software: An update. *ACM SIGKDD explorations newsletter*, *11*(1), 10–18.

Heninger, N. (2022). Rsa, dh, and dsa in the wild. *Cryptology ePrint Archive*.

Hettwer, B., Gehrer, S., & Güneysu, T. (2020). Applications of machine learning techniques in side-channel attacks: A survey. *Journal of Cryptographic Engineering*, *10*(2), 135–162.

Hintze, J. L., & Nelson, R. D. (1998). Violin plots: A box plot-density trace synergism. *The American Statistician*, *52*(2), 181–184.

Hjelmvik, E., & John, W. (2009). Statistical protocol identification with spid: Preliminary results. *Swedish National Computer Networking Workshop*, *9*, 4–5.

Hjelmvik, E., & John, W. (2010). Breaking and improving protocol obfuscation. *Chalmers University of Technology, Tech. Rep*, *123751*.

Hodo, E., Bellekens, X., Iorkyase, E., Hamilton, A., Tachtatzis, C., & Atkinson, R. (2017). Machine learning approach for detection of nonTor traffic. *Proceedings of the 12th International Conference on Availability, Reliability and Security*, 1–6.

Hofmann, T. (2013). Probabilistic latent semantic analysis. *arXiv preprint arXiv:1301.6705*.

Holmes, G., Donkin, A., & Witten, I. H. (1994). Weka: A machine learning workbench. *Proceedings of ANZIIS'94-Australian New Zealnd Intelligent Information Systems Conference*, 357–361.

Hopper, N. (2014). Challenges in protecting tor hidden services from botnet abuse. *Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers 18*, 316–325.

Hopper, N., Vasserman, E. Y., & Chan-Tin, E. (2010). How much anonymity does network latency leak? *ACM Transactions on Information and System Security (TISSEC)*, *13*(2), 1–28.

Hoy, W. K., & Adams, C. M. (2015). *Quantitative research in education: A primer*. Sage Publications.

Hunt, C. (2002). *TCP/IP network administration* (Vol. 3). O'Reilly Media, Inc.

Ismailaj, K., Camelo, M., & Latré, S. (2021). When deep learning may not be the right tool for traffic classification. *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 884–889.

Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, *31*(3), 685–695.

Johnson, L. (2019). *Security controls evaluation, testing, and assessment handbook*. Academic Press.

Kahn, D. (1973). *The codebreakers: The story of secret writing* (1st ed.). The Macmillan Company.

Karagiannis, T., Broido, A., Brownlee, N., Claffy, K., & Faloutsos, M. (2003). File-sharing in the internet: A characterization of P2P traffic in the backbone. *University of California, Riverside, USA, Tech. Rep.*

Karagiannis, T., Broido, A., Brownlee, N., kc claffy, & Faloutsos, M. (2004). Is P2P dying or just hiding?

Karagiannis, T., Broido, A., Faloutsos, M., & Claffy, K. (2004). Transport layer identification of P2P traffic. *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, 121–134. https://doi.org/10.1145/1028788.1028804

Karagiannis, T., Papagiannaki, K., & Faloutsos, M. (2005). Blinc: Multilevel traffic classification in the dark. *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, 229–240.

Karunanayake, I., Ahmed, N., Malaney, R., Islam, R., & Jha, S. (2020). Anonymity with Tor: A survey on Tor attacks. *arXiv preprint arXiv:2009.13018*.

Karunanayake, I., Ahmed, N., Malaney, R., Islam, R., & Jha, S. K. (2021). De-anonymisation attacks on tor: A survey. *IEEE Communications Surveys & Tutorials*, *23*(4), 2324–2350.

Katz, J., & Lindell, Y. (2020). *Introduction to modern cryptography*. CRC press.

Kerlinger, F. N. (1966). Foundations of behavioral research.

Keum, J., Kornelsen, K. C., Leach, J. M., & Coulibaly, P. (2017). Entropy applications to water monitoring network design: A review. *Entropy*, *19*(11), 613.

Kim, M., & Anpalagan, A. (2018). Tor Traffic Classification from Raw Packet Header using Convolutional Neural Network. *2018 1st IEEE International Conference on Knowledge Innovation and Invention (ICKII)*, 187–190.

Koch, R. (2019). *The list of countries that have banned vpns*. Retrieved July 15, 2021, from https://protonvpn.com/blog/are-vpns-illegal/

Kota, C. M., & Aissi, C. (2022). Implementation of the rsa algorithm and its cryptanalysis. *2002 GSW*.

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text classification algorithms: A survey. *Information*, *10*(4), 150.

Krishna, N. R. (2019). Classifying classic ciphers using machine learning.

Kuhn, M., Johnson, K., et al. (2013). *Applied predictive modeling* (Vol. 26). Springer.

Kwon, A., AlSabah, M., Lazar, D., Dacier, M., & Devadas, S. (2015). Circuit fingerprinting attacks: Passive deanonymization of Tor hidden services. *24th {USENIX} Security Symposium ({USENIX} Security 15)*, 287–302.

Lapshichyov, V., & Makarevich, O. (2019). TLS certificate as a sign of establishing a connection with the network Tor. *Proceedings of the 12th International Conference on Security of Information and Networks*, 1–6.

Lashkari, A. H., Draper-Gil, G., Mamun, M. S. I., & Ghorbani, A. A. (2017). Characterization of Tor traffic using time based features. *ICISSp*, 253–262.

Lee, L., Fifield, D., Malkin, N., Iyer, G., Egelman, S., & Wagner, D. (2016). *Tor's usability for censorship circumvention* [Doctoral dissertation, Master's thesis, EECS Department, University of California, Berkeley].

Leiner, B. M., Cerf, V. G., Clark, D. D., Kahn, R. E., Kleinrock, L., Lynch, D. C., Postel, J., Roberts, L. G., & Wolff, S. S. (1997). The past and future history of the internet. *Communications of the ACM*, *40*(2), 102–108.

Lemons, D. S. (2013). *A student's guide to entropy*. Cambridge University Press.

Li, B., Erdin, E., Gunes, M. H., Bebis, G., & Shipley, T. (2013). An overview of anonymity technology usage. *Computer Communications*, *36*(12), 1269–1283.

Liang, H., Sun, X., Sun, Y., & Gao, Y. (2017). Text feature extraction based on deep learning: A review. *EURASIP journal on wireless communications and networking*, *2017*(1), 1–12.

Liggett, R., Lee, J. R., Roddy, A. L., & Wallin, M. A. (2020). The dark web as a platform for crime: An exploration of illicit drug, firearm, csam, and cybercrime markets. *The Palgrave handbook of international cybercrime and cyberdeviance*, 91–116.

Ling, Z., Luo, J., Yu, W., & Fu, X. (2011). Equal-sized cells mean equal-sized packets in Tor? *2011 IEEE International Conference on Communications (ICC)*, 1–6.

Ling, Z., Luo, J., Yu, W., Fu, X., Xuan, D., & Jia, W. (2012). A new cell-counting-based attack against Tor. *IEEE/ACM Transactions On Networking*, *20*(4), 1245–1261.

Lingyu, J., Yang, L., Bailing, W., Hongri, L., & Guodong, X. (2017). A hierarchical classification approach for Tor anonymous traffic. *2017 IEEE 9th International conference on communication software and networks (ICCSN)*, 239–243. https://doi.org/10.1109/ICCSN.2017.8230113

Liu, B. (2011). *Web data mining: Exploring hyperlinks, contents, and usage data* (Vol. 1). Springer.

Liu, H., & Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. *applied sciences*, *9*(20), 4396.

Liu, R., & Yu, X. (2020). A survey on encrypted traffic identification. *Proceedings of the 2020 International Conference on Cyberspace Innovation of Advanced Technologies*, 159–163.

Liu, Y., Wang, Y., & Zhang, J. (2012). New machine learning algorithm: Random forest. *International Conference on Information Computing and Applications*, 246–252.

Lotfollahi, M., Jafari Siavoshani, M., Shirali Hossein Zade, R., & Saberian, M. (2020). Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing*, *24*(3), 1999–2012.

Loyola-Gonzalez, O. (2019). Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view. *IEEE Access*, *7*, 154096–154113.

Luo, X. (2021). Efficient english text classification using selected machine learning techniques. *Alexandria Engineering Journal*, *60*(3), 3401–3409.

Lyda, R., & Hamrock, J. (2007). Using entropy analysis to find encrypted and packed malware. *IEEE Security & Privacy*, *5*(2), 40–45.

Madhukar, A., & Williamson, C. (2006). A longitudinal study of P2P traffic classification. *14th IEEE International Symposium on Modeling, Analysis, and Simulation*, 179–188. https://doi.org/10.1109/MASCOTS.2006.6

Maghrebi, H., Portigliatti, T., & Prouff, E. (2016). Breaking cryptographic implementations using deep learning techniques. *Security, Privacy, and Applied Cryptography Engineering: 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings 6*, 3–26.

Mahdavi, E., Hassannejad, H., et al. (2018). Classification of encrypted traffic for applications based on statistical features. *ISeCure-The ISC International Journal of Information Security*, *10*(1), 29–43.

Mamun, M. S. I., Ghorbani, A. A., & Stakhanova, N. (2015). An entropy based encrypted traffic classifier. *International Conference on Information and Communications Security*, 282–294.

Martin, K. M. (2017, July). Everyday cryptography: Fundamental principles and applications. https://doi.org/10.1093/oso/9780198788003.001.0001

Mathew, A., Amudha, P., & Sivakumari, S. (2020). Deep learning techniques: An overview. *International conference on advanced machine learning technologies and applications*, 599–608.

Mayank, P., & Singh, A. (2017). Tor traffic identification. *2017 7th International Conference on Communication Systems and Network Technologies (CSNT)*, 85–91.

McDonald, N. G. (2015). Past, present, and future methods of cryptography and data encryption. *Department of Electrical and Computer Engineering University of Utah*.

Melicher, W., Ur, B., Segreti, S. M., Komanduri, S., Bauer, L., Christin, N., & Cranor, L. F. (2016). Fast, lean, and accurate: Modeling password guessability using neural networks. *USENIX Security Symposium*, 175–191.

Menzies, T., Majumder, S., Balaji, N., Brey, K., & Fu, W. (2018). 500+ times faster than deep learning:(a case study exploring faster methods for text mining stackoverflow). *2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)*, 554–563.

Mercaldo, F., & Martinelli, F. (2017). Tor traffic analysis and identification. *2017 AEIT International Annual Conference*, 1–6.

Mirea, M., Wang, V., & Jung, J. (2018). The not so dark side of the darknet: A qualitative study [12 month embargo]. *Security Journal*. https://doi.org/10.1057/s41284-018-0150-5

Mishra, P., Pandey, C. M., Singh, U., Gupta, A., Sahu, C., & Keshri, A. (2019). Descriptive statistics and normality tests for statistical data. *Annals of cardiac anaesthesia*, *22*(1), 67.

Mitchell, T. M. (1997). *Machine learning* (1st ed.). McGraw-Hill, Inc.

Modadugu, N., & Rescorla, E. (2004). The Design and Implementation of Datagram TLS. *NDSS*.

Moldagulova, A., & Sulaiman, R. B. (2017). Using KNN algorithm for classification of textual documents. *2017 8th international conference on information technology (ICIT)*, 665–671.

Monk, B., Mitchell, J., Frank, R., & Davies, G. (2018). Uncovering Tor: An examination of the network structure. *Security and communication networks*, *2018*.

Moore, A., Zuev, D., & Crogan, M. (2013). *Discriminators for use in flow-based classification* (tech. rep.). Queen Mary University of London. https://qmro.qmul.ac.uk/xmlui/bitstream/handle/123456789/5050/RR-05-13.pdf

Moore, A. W., & Papagiannaki, K. (2005). Toward the accurate identification of network applications. *International workshop on passive and active network measurement*, 41–54.

Mousa, A., Faragallah, O. S., El-Rabaie, S., & Nigm, E. (2013). Security analysis of reverse encryption algorithm for databases. *International Journal of Computer Applications*, *66*(14).

Mrphs. (2016, August). *Breaking through censorship barriers, even when Tor is blocked.* Retrieved March 12, 2021, from https://blog.torproject.org/breaking-through-censorship-barriers-even-when-tor-blocked

Mueller, J. P., & Massaron, L. (2019). *Deep learning for dummies.* John Wiley & Sons.

Müller, A. C., & Guido, S. (2018). *Introduction to machine learning with python: A guide for data scientists.* " O'Reilly Media, Inc."

Murdoch, S. J., & Zieliński, P. (2007). Sampled traffic analysis by internet-exchange-level adversaries. *International workshop on privacy enhancing technologies*, 167–183.

Mushtaq, M. F., Jamel, S., Disina, A. H., Pindar, Z. A., Shakir, N. S. A., & Deris, M. M. (2017). A survey on the cryptographic encryption algorithms. *International Journal of Advanced Computer Science and Applications*, *8*(11).

Nachar, N., et al. (2008). The mann-whitney u: A test for assessing whether two independent samples come from the same distribution. *Tutorials in quantitative Methods for Psychology*, *4*(1), 13–20.

Nagar, S. (2018). Ipython. In *Introduction to python for engineers and scientists* (pp. 31–45). Springer.

Natal, J., Ávila, I., Tsukahara, V. B., Pinheiro, M., & Maciel, C. D. (2021). Entropy: From thermodynamics to information processing. *Entropy*, *23*(10), 1340.

Nguyen, T. T., & Armitage, G. (2008). A survey of techniques for internet traffic classification using machine learning. *IEEE communications surveys & tutorials*, *10*(4), 56–76.

Niranjana, R., Kumar, V. A., & Sheen, S. (2020). Darknet traffic analysis and classification using numerical agm and mean shift clustering algorithm. *SN Computer Science*, *1*(1), 1–10.

Nistico, A., Markudova, D., Trevisan, M., Meo, M., & Carofiglio, G. (2020). A comparative study of rtc applications. *2020 IEEE International Symposium on Multimedia (ISM)*, 1–8.

Ott, R. L., & Longnecker, M. T. (2015). *An introduction to statistical methods and data analysis*. Cengage Learning.

Owen, G., & Savage, N. (2015). *The Tor dark net* (tech. rep.). Centre for International Governance Innovation; the Royal Institute of International Affairs. https://www.cigionline.org/sites/default/files/no20_0.pdf

Pagano, R. R. (2012). *Understanding statistics in the behavioral sciences*. Cengage Learning.

Patil, P., Narayankar, P., Narayan, D., & Meena, S. M. (2016). A comprehensive evaluation of cryptographic algorithms: DES, 3DES, AES, RSA and Blowfish. *Procedia Computer Science*, *78*, 617–624.

Pektaş, A., & Acarman, T. (2019). A deep learning method to detect network intrusion through flow-based features. *International Journal of Network Management*, *29*(3), e2050.

Perkel, J. M. (2018). Why jupyter is data scientists' computational notebook of choice. *Nature*, *563*(7732), 145–147.

Platzer, F., Schäfer, M., & Steinebach, M. (2020). Critical traffic analysis on the tor network. *Proceedings of the 15th International Conference on Availability, Reliability and Security*, 1–10.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, *1*(1), 81–106.

Quinlan, J. R. (2014). *C4. 5: Programs for machine learning*. Elsevier.

Radovanović, M., & Ivanović, M. (2008). Text mining: Approaches and applications. *Novi Sad J. Math*, *38*(3), 227–234.

Raigoza, J., & Jituri, K. (2016). Evaluating performance of symmetric encryption algorithms. *2016 international conference on computational science and computational intelligence (CSCI)*, 1378–1379.

Rao, F.-Y. (2017). On the security of a variant of elgamal encryption scheme. *IEEE Transactions on Dependable and Secure Computing*, *16*(4), 725–728.

Rescorla, E. (2018, August). The Transport Layer Security (TLS) Protocol Version 1.3 [Accessed: 2021-03-15]. https://datatracker.ietf.org/doc/html/rfc8446

Reynolds, J., & Postel, J. (1992). Assigned numbers. *STD 2, RFC 1700, USC/INFORMATION SCIENCES INSTITUTE*.

Rezaei, S., & Liu, X. (2019). Deep learning for encrypted traffic classification: An overview. *IEEE communications magazine*, *57*(5), 76–81.

Ristic, I. (2015). *Bulletproof SSL and TLS: Understanding and deploying ssl/tls and pki to secure servers and web applications*. Feisty Duck Limited.

Robertazzi, T. G., & Shi, L. (2020). Machine learning in networking. In *Networking and computation: Technology, modeling and performance* (pp. 151–190). Springer International Publishing. https://doi.org/10.1007/978-3-030-36704-6_7

Rokach, L., & Maimon, O. (2015). Data mining with decision trees–theory and applications 2nd edition. *Series Mach. Percept. Artif. Intell*, *81*, 328.

Saad, S. E., & Yang, J. (2019). Twitter sentiment analysis based on ordinal regression. *IEEE Access*, *7*, 163677–163685.

Salman, O., Elhajj, I. H., Kayssi, A., & Chehab, A. (2020). A review on machine learning–based approaches for internet traffic classification. *Annals of Telecommunications*, *75*, 673–710.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, *3*(3), 210–229.

Sanap, S., & More, V. (2021). Analysis of encryption techniques for secure communication. *2021 International Conference on Emerging Smart Computing and Informatics (ESCI)*, 290–294.

Sanders, C. (2017). *Practical packet analysis: Using wireshark to solve real-world network problems* (3rd). No Starch Press.

Santos-González, I., Rivero-García, A., Molina-Gil, J., & Caballero-Gil, P. (2017). Implementation and analysis of real-time streaming protocols. *Sensors*, *17*(4), 846.

Saputra, F. A., Nadhori, I. U., & Barry, B. F. (2016). Detecting and blocking onion router traffic using deep packet inspection. *2016 International Electronics Symposium (IES)*, 283–288.

Sarkar, D., Vinod, P., & Yerima, S. Y. (2020). Detection of Tor traffic using deep learning. *2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA)*, 1–8.

Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, *2*(3), 1–21.

Sarwar, M. B., Hanif, M. K., Talib, R., Younas, M., & Sarwar, M. U. (2021). Darkdetect: Darknet traffic detection and categorization using modified convolution-long short-term memory. *IEEE Access*, *9*, 113705–113713.

Satapathy, A., Livingston, J., et al. (2016). A comprehensive survey on ssl/tls and their vulnerabilities. *International Journal of Computer Applications*, *153*(5), 31–38.

Schatzmann, D., Mühlbauer, W., Spyropoulos, T., & Dimitropoulos, X. (2010). Digging into https: Flow-based classification of webmail traffic. *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 322–327.

Schrödel, T. (2008). Breaking short vigenère ciphers. *Cryptologia*, *32*(4), 334–347.

Shafiq, M., Yu, X., Laghari, A. A., Yao, L., Karn, N. K., & Abdessamia, F. (2016). Network traffic classification techniques and comparative analysis using machine learning algorithms. *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, 2451–2455.

Shahbar, K., & Zincir-Heywood, A. N. (2014). Benchmarking two techniques for Tor classification: Flow level and circuit level classification. *2014 IEEE Symposium on Computational Intelligence in Cyber Security (CICS)*, 1–8.

Shahbar, K., & Zincir-Heywood, A. N. (2015). Traffic flow analysis of Tor pluggable transports. *2015 11th International Conference on Network and Service Management (CNSM)*, 178–181.

Shahbar, K., & Zincir-Heywood, A. N. (2017). Anon17: Network traffic dataset of anonymity services. *Faculty of Computer Science Dalhousie University, Tech. Rep.*

Shaheen, F., Verma, B., & Asafuddoula, M. (2016). Impact of automatic feature extraction in deep learning architecture. *2016 International conference on digital image computing: techniques and applications (DICTA)*, 1–8.

Shaik, A. B., & Srinivasan, S. (2019). A brief survey on random forest ensembles in classification model. *International Conference on Innovative Computing and Communications*, 253–260.

Shamrat, F. J. M., Ranjan, R., Md, K., Hasib, A. Y., & Siddique, A. H. (2021). Performance Evaluation among ID3, C4. 5, and CART Decision Tree Algorithms. *Pervasive Computing and Social Networking: Proceedings of ICPCSN 2021*, *317*, 127.

Shannon, C. E. (1951). Prediction and entropy of printed english. *Bell system technical journal*, *30*(1), 50–64.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, *27*(3), 379–423.

Shavers, B., & Bair, J. (2016). Chapter 2 - the Tor browser. In B. Shavers & J. Bair (Eds.), *Hiding behind the keyboard* (pp. 11–34). Syngress. https://doi.org/https://doi.org/10.1016/B978-0-12-803340-1.00002-1

Sherry, J., Lan, C., Popa, R. A., & Ratnasamy, S. (2015). Blindbox: Deep packet inspection over encrypted traffic. *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 213–226.

Sheu, Y.-h. (2020). Illuminating the black box: Interpreting deep neural network models for psychiatric research. *Frontiers in Psychiatry*, *11*, 551299.

Shinde, P. P., & Shah, S. (2018). A review of machine learning and deep learning applications. *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*, 1–6.

Singh, M., Jakhar, A. K., & Pandey, S. (2021). Sentiment analysis on the impact of coronavirus in social life using the bert model. *Social Network Analysis and Mining*, *11*(1), 1–11.

Soleimani, M. H. M., Mansoorizadeh, M., & Nassiri, M. (2018). Real-time identification of three Tor pluggable transports using machine learning techniques. *The Journal of Supercomputing*, *74*(10), 4910–4927.

Song, Y.-Y., & Ying, L. (2015). Decision tree methods: Applications for classification and prediction. *Shanghai archives of psychiatry*, *27*(2), 130.

Sun, Y., Edmundson, A., Vanbever, L., Li, O., Rexford, J., Chiang, M., & Mittal, P. (2015). {Raptor}: Routing attacks on privacy in Tor. *24th {USENIX} Security Symposium ({USENIX} Security 15)*, 271–286.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Syarif, I., Prugel-Bennett, A., & Wills, G. (2012). Unsupervised clustering approach for network anomaly detection. *International conference on networked digital technologies*, 135–145.

Tails. (2022). *How tails works*. https://tails.boum.org/about/index.en.html

Tanenbaum, A. S., Wetherall, D. J., et al. (2021). *Computer networks* (6th ed.). Pearson Education Limited.

Tang, Z., Zeng, X., & Sheng, Y. (2019). Entropy-based feature extraction algorithm for encrypted and non-encrypted compressed traffic classification. *International Journal of ICIC*, *15*(3), 845.

Tanha, J., van Someren, M., & Afsarmanesh, H. (2017). Semi-supervised self-training for decision tree classifiers. *International Journal of Machine Learning and Cybernetics*, *8*(1), 355–370.

The Tor Project. (2021a). *Tor metrics*. Retrieved March 12, 2023, from https://metrics.torproject.org/

The Tor Project. (2021b). *Why is it called tor?* Retrieved March 12, 2021, from https://support.torproject.org/

Timon, B. (2019). Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 107–131.

Tong, V., Tran, H. A., Souihi, S., & Mellouk, A. (2018). A novel quic traffic classifier based on convolutional neural networks. *2018 IEEE Global Communications Conference (GLOBECOM)*, 1–6.

Tran, N. P., Nguyen, D. T., Le, H. H., Nguyen, N. T., & Nguyen, N. B. (2020). An efficient algorithm to extract control flow-based features for iot malware detection. *The Computer Journal*.

Tsiounis, Y., & Yung, M. (1998). On the security of elgamal based encryption. *International Workshop on Public Key Cryptography*, 117–134.

Vápeník, R., Michalko, M., Janitor, J., & Jakab, F. (2014). Secured web oriented videoconferencing system for educational purposes using webrtc technology. *2014 IEEE 12th IEEE International Conference on Emerging eLearning Technologies and Applications (ICETA)*, 495–500.

Velan, P., Čermák, M., Čeleda, P., & Drašar, M. (2015). A survey of methods for encrypted traffic classification and analysis. *International Journal of Network Management*, *25*(5), 355–374.

Verdu, S. (1998). Fifty years of shannon theory. *IEEE Transactions on information theory*, *44*(6), 2057–2078.

Vinayakumar, R., Soman, K., & Poornachandran, P. (2017). Secure shell (ssh) traffic analysis with flow based features using shallow and deep networks. *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2026–2032.

Vint, C., & Kahn, R. (1974). A protocol for packet network interconnection. *IEEE Transactions of Communications*, *22*(5), 637–48.

Wang, B., Spencer, B., Ling, C. X., & Zhang, H. (2008). Semi-supervised self-training for sentence subjectivity classification. *Conference of the Canadian Society for Computational Studies of Intelligence*, 344–355.

Wang, L., Mei, H., & Sheng, V. S. (2020). Multilevel identification and classification analysis of Tor on mobile and pc platforms. *IEEE Transactions on Industrial Informatics*, *17*(2), 1079–1088.

Wang, W., Shang, Y., He, Y., Li, Y., & Liu, J. (2020). Botmark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors. *Information Sciences*, *511*, 284–296.

Wang, X., & Cronin, B. (2014). TCP/IP reassembly in network intrusion detection and prevention systems. *International Journal of Information Security and Privacy (IJISP)*, *8*(3), 63–76.

Weidman, S. (2019). *Deep learning from scratch: Building with python from first principles*. O'Reilly Media.

Weir, G. (2007). The posit text profiling toolset. *Proceedings of the 12th Conference of Pan-Pacific Association of Applied Linguistics*.

Weir, G. (2009). Corpus profiling with the posit tools. *Proceedings of the 5th Corpus Linguistics Conference. University of Liverpool*.

Weir, G., Owoeye, K., Oberacker, A., & Alshahrani, H. (2018). Cloud-based textual analysis as a basis for document classification. *2018 International Conference on High Performance Computing & Simulation (HPCS)*, 672–676.

Wiek, A., & Lang, D. J. (2016). Transformational sustainability research methodology. In *Sustainability science* (pp. 31–41). Springer.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2017). *Data mining: Practical machine learning tools and techniques* (Fourth Edition). Morgan Kaufmann. https://www.sciencedirect.com/book/9780128042915/data-mining

Wood, C. A., & Uzun, E. (2014). Flexible end-to-end content security in ccn. *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*, 858–865.

Yaga, D., Mell, P., Roby, N., & Scarfone, K. (2019). Blockchain technology overview. *arXiv preprint arXiv:1906.11078*.

Yeo, M., Koo, Y., Yoon, Y., Hwang, T., Ryu, J., Song, J., & Park, C. (2018). Flow-based malware detection using convolutional neural network. *2018 International Conference on Information Networking (ICOIN)*, 910–913.

Yetter, R. B. (2015). *Darknets, cybercrime & the onion router: Anonymity & security in cyberspace* [Doctoral dissertation, Utica College].

Yin, R. K. (2009). *Case study research: Design and methods* (Vol. 5). sage.

Ying, X. (2019). An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, *1168*(2), 022022.

Zhang, B., Kreitz, G., Isaksson, M., Ubillos, J., Urdaneta, G., Pouwelse, J. A., & Epema, D. (2013). Understanding user behavior in spotify. *2013 Proceedings IEEE INFOCOM*, 220–224.

Zhang, H., Lu, G., Qassrawi, M. T., Zhang, Y., & Yu, X. (2012). Feature selection for optimizing traffic classification. *Computer Communications*, *35*(12), 1457–1471.

Zhang, Q., Yang, L. T., Chen, Z., & Li, P. (2018). A survey on deep learning for big data. *Information Fusion*, *42*, 146–157.

Zhang, S., Cheng, D., Deng, Z., Zong, M., & Deng, X. (2018). A novel kNN algorithm with data-driven k parameter computation. *Pattern Recognition Letters*, *109*, 44–54.

Zhang, S., Li, X., Zong, M., Zhu, X., & Cheng, D. (2017). Learning k for kNN classification. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *8*(3), 1–19.

Zhou, D., Yan, Z., Fu, Y., & Yao, Z. (2018). A survey on network data collection. *Journal of Network and Computer Applications*, *116*, 9–23.

Zhou, K., Wang, W., Wu, C., & Hu, T. (2020). Practical evaluation of encrypted traffic classification based on a combined method of entropy estimation and neural networks. *ETRI Journal*, *42*(3), 311–323.

Ziegler, A., & König, I. R. (2014). Mining data with random forests: Current options for real-world applications. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *4*(1), 55–63.

Appendix A

Python Scripts

A.1 charstat.py

```
1  import os
2  import pandas as pd
3  import collections
4  from scipy.stats import entropy
5
6  # Define network type for working directory and label
7  Tor = 'y'
8
9  # Function to process pcap files
10 def process_pcap(pcap):
11     pcap = pcap.lower()
12     # Conditions to process pcap files based on different app types
13     if (Tor == 'y' and 'torg' in pcap):
14         conditions = [
15         (('audio' in pcap), 'tshark -r %s -Y "(tls.app_data &&
           ↪  !tls.handshake && !_ws.expert)" -Tfields -e frame.number -e
           ↪  ip.src -e ip.dst -e tcp.payload', 'audio', 'tcp.payload'),
16         (('browsing' in pcap), 'tshark -r %s -Y "(tls.app_data &&
           ↪  !tls.handshake && !_ws.expert)" -Tfields -e frame.number -e
           ↪  ip.src -e ip.dst -e tls.app_data', 'browsing',
           ↪  'tls_app_data'),
17         (('chat' in pcap), 'tshark -r %s -Y "(tls.app_data &&
           ↪  !tls.handshake && !_ws.expert)" -Tfields -e frame.number -e
           ↪  ip.src -e ip.dst -e tcp.payload', 'chat', 'tcp.payload'),
18         (('transfer' in pcap), 'tshark -r %s -Y "(tls.app_data &&
           ↪  !tls.handshake && !_ws.expert)" -Tfields -e frame.number -e
           ↪  ip.src -e ip.dst -e tcp.payload', 'ftp', 'tcp.payload'),
```

```
19          (('mail' in pcap), 'tshark -r %s -Y "(tls.app_data &&
            ↪  !tls.handshake && !_ws.expert)" -Tfields -e frame.number -e
            ↪  ip.src -e ip.dst -e tcp.payload', 'email', 'tcp.payload'),
20          (('p2p' in pcap), 'tshark -r %s -Y "(tls.app_data &&
            ↪  !tls.handshake && !_ws.expert)" -Tfields -e frame.number -e
            ↪  ip.src -e ip.dst -e tcp.payload', 'p2p', 'tcp.payload'),
21          (('video' in pcap), 'tshark -r %s -Y "(tls.app_data &&
            ↪  !tls.handshake && !_ws.expert)" -Tfields -e frame.number -e
            ↪  ip.src -e ip.dst -e tls.app_data', 'vdo', 'tls.app_data'),
22          (('voip' in pcap or 'ssl' in pcap), 'tshark -r %s -Y
            ↪  "(tls.app_data && !tls.handshake && !_ws.expert)" -Tfields
            ↪ -e frame.number -e ip.src -e ip.dst -e tls.app_data', 'voip',
            ↪  'tls_app_data'),
23          (('private' in pcap), 'tshark -r %s -Y "(tls.app_data &&
            ↪  !tls.handshake && !_ws.expert)" -Tfields -e frame.number -e
            ↪  ip.src -e ip.dst -e tls.app_data', 'private', 'tls_app_data')
            ↪  ]
24      else:
25          conditions = [
26          (('aim' in pcap or 'icq' in pcap), 'tshark -r %s -Y
            ↪  "(tcp.payload && !tls && !_ws.expert)" -Tfields -e
            ↪  frame.number -e ip.src -e ip.dst -e tcp.payload', 'chat',
            ↪  'tcp.payload'),
27          (('facebook' in pcap or 'hangout_chat' in pcap or 'hangoutchat'
            ↪  in pcap), 'tshark -r %s -Y "(tls.app_data && !tls.handshake
            ↪  && !_ws.expert)" -Tfields -e frame.number -e ip.src -e
            ↪  ip.dst -e tls.app_data', 'chat', 'tls_app_data'),
28          (('skype_chat' in pcap or 'skypechat' in pcap), 'tshark -r %s -Y
            ↪  "(tcp.payload && !tls && !_ws.expert)" -Tfields -e
            ↪  frame.number -e ip.src -e ip.dst -e tcp.payload', 'chat',
            ↪  'tcp.payload'),
29          (('spotify' in pcap), 'tshark -r %s -Y "(tcp.payload && !http &&
            ↪  !tls && !_ws.expert)" -Tfields -e frame.number -e ip.src -e
            ↪  ip.dst -e tcp.payload', 'audio', 'tcp.payload'),
30          (('skype_audio' in pcap or 'skype_voice' in pcap or
            ↪  'skype_transfer' in pcap), 'tshark -r %s -Y "(tcp.payload &&
            ↪  !http && !tls && !_ws.expert)" -Tfields -e frame.number -e
            ↪  ip.src -e ip.dst -e tcp.payload', 'audio', 'tcp.payload'),
31          (('p2p' in pcap), 'tshark -r %s -Y "(tcp.payload && !bittorrent
            ↪  && !tls && !_ws.expert)" -Tfields -e frame.number -e ip.src
            ↪ -e ip.dst -e tcp.payload', 'p2p', 'tcp.payload'),
32          (('sftp' in pcap), 'tshark -r %s -Y "(ssh.encrypted_packet &&
            ↪  !_ws.expert)" -Tfields -e frame.number -e ip.src -e ip.dst
            ↪ -e tls.app_data', 'ftp', 'tls.app_data'),
```

```
33          (('browsing' in pcap or 'ssl' in pcap), 'tshark -r %s -Y
            ↪  "(tls.app_data && !tls.handshake && !_ws.expert)" -Tfields
            ↪  -e frame.number -e ip.src -e ip.dst -e tls.app_data',
            ↪  'browsing', 'tls_app_data'),
34          (('email' in pcap or 'thunderbird' in pcap), 'tshark -r %s -Y
            ↪  "(tls.app_data && !tls.handshake && !_ws.expert)" -Tfields
            ↪  -e frame.number -e ip.src -e ip.dst -e tls.app_data',
            ↪  'email', 'tls_app_data'),
35          (('vimeo' in pcap or 'youtube' in pcap), 'tshark -r %s -Y
            ↪  "(tls.app_data && !tls.handshake && !_ws.expert)" -Tfields
            ↪  -e frame.number -e ip.src -e ip.dst -e tls.app_data', 'vdo',
            ↪  'tls_app_data'),
36          (('hangout_audio' in pcap), 'tshark -r %s -Y "(tcp.payload &&
            ↪  !tls && !_ws.expert)" -Tfields -e frame.number -e ip.src -e
            ↪  ip.dst -e tcp.payload', 'audio', 'tcp.payload')
37          ]
38
39      # Iterate through conditions and return the command, app_type and
        ↪  field_name if condition is met
40      for condition, command, app_type, field_name in conditions:
41          if condition:
42              return command, app_type, field_name
43      print(app_type)
44      return None, None, None
45
46  # Function to estimate the Shannon entropy of a sequence
47  def estimate_shannon_entropy(dna_sequence):
48      bases = collections.Counter([tmp_base for tmp_base in dna_sequence])
49      dist = [x/sum(bases.values()) for x in bases.values()]
50      entropy_value = entropy(dist, base=2)
51      return entropy_value
52
53  directory = os.getcwd()
54
55  # Define the headers for the output CSV file
56  headers = ['F_{:x}'.format(i) for i in range(16)] + ['R_{:x}'.format(i)
    ↪  for i in range(16)] + ['t', 'e']
57
58  #Define the network type (Tor or Non-Tor)
59  network_type = 'tor' if Tor == 'y' else 'nontor'
60
61  # Iterate through pcap files in the directory
62  for pcap in os.listdir(directory):
63      if pcap.endswith(".pcap"):
64          pcap_path = os.path.join(directory, pcap)
```

```
65          csv_name = 'features_' + pcap.replace(".pcap", "") + '.csv'
66          csv_path = os.path.join(directory, csv_name)
67          # Process the pcap file and get the command, app_type, and
            ↪  field_name
68          command, app_type, field_name = process_pcap(pcap)
69          if command: # Run the tshark command
70              print(f"Running tshark command: {command}")  # Add this line
                ↪  to print the tshark command
71              os.system((command % pcap_path) + f" > {csv_path}")
72
73          # Read the generated CSV file containing the payloads
74          df = pd.read_csv(csv_path, sep='\t', header=None,
            ↪  names=['frame_number', 'ip_src', 'ip_dst', field_name])
75
76          # Prepare a list for storing the extracted features
77          extracted_features = []
78
79          # Iterate through each row (payload) in the DataFrame
80          for _, row in df.iterrows():
81              payload = row[field_name]
82
83              # Check if payload is a string before processing
84              if isinstance(payload, str):
85                  # Split the payload by comma
86                  payloads = payload.split(',')
87
88                  # Iterate through payload parts
89                  for payload_part in payloads:
90                      hex_freq = collections.Counter(payload_part)
91                      hex_ratio = {k: v / len(payload_part) * 100 for k, v
                        ↪  in hex_freq.items()}
92                      total_chars = len(payload_part)
93                      shannon_entropy =
                        ↪  estimate_shannon_entropy(payload_part)
94
95                      # Combine the features into a single list
96                      features = [hex_freq.get(k, 0) for k in
                        ↪  '0123456789abcdef'] + \
97                                 [hex_ratio.get(k, 0.0) for k in
                                   ↪  '0123456789abcdef'] + \
98                                 [total_chars, shannon_entropy]
99                      extracted_features.append(features)
100             else:
101                 print(f"Skipped non-string payload: {payload}")  # Add
                    ↪  this line to print non-string payloads
```

```
102
103         # Convert the extracted features list into a DataFrame and save
            ↪  it as a CSV file
104         features_df = pd.DataFrame(extracted_features, columns=headers)
105
106         # Assign the label by combining network_type and app_type
107         label = f"{network_type}-{app_type}"
108         print(f"Assigned label: {label}")  # Add this line to print the
            ↪  assigned label
109         features_df['class'] = label
110
111         features_df.to_csv(csv_path, index=False)
```

A.2 website_lists.py

```
1  import web-browser
2
3  #website_list_1
4  webbrowser.open("https://www.google.com")
5  webbrowser.open("https://www.apple.com")
6  webbrowser.open("https://www.cloudflare.com")
7  webbrowser.open("https://www.microsoft.com")
8  webbrowser.open("https://www.blogger.com")
9  webbrowser.open("https://wordpress.org/")
10 webbrowser.open("https://www.mozilla.org")
11 webbrowser.open("https://en.wikipedia.org/")
12 webbrowser.open("https://www.linkedin.com/")
13 webbrowser.open("https://europa.eu/")
14 webbrowser.open("https://www.adobe.com/")
15 webbrowser.open("https://vimeo.com/")
16 webbrowser.open("https://telegram.org/")
17 webbrowser.open("https://github.com/")
18 webbrowser.open("https://vk.com/")
19 webbrowser.open("https://www.amazon.com/")
20 webbrowser.open("https://www.istockphoto.com/")
21 webbrowser.open("https://www.uol.com.br/")
22 webbrowser.open("https://www.whatsapp.com/")
23 webbrowser.open("https://www.bbc.co.uk/")
24
25 #website_list_2
26 webbrowser.open("https://www.facebook.com/")
27 #webbrowser.open("https://www.netvibes.com/")
28 webbrowser.open("https://www.nytimes.com/")
29 webbrowser.open("https://www.terra.com.br/")
```

```
30 #webbrowser.open("https://www.paypal.com/")
31 webbrowser.open("https://www.nih.gov/")
32 webbrowser.open("https://www.hugedomains.com/")
33 webbrowser.open("https://www.imdb.com/")
34 webbrowser.open("https://www.slideshare.net/")
35 webbrowser.open("https://issuu.com/")
36 webbrowser.open("https://www.globo.com/")
37 webbrowser.open("https://www.w3.org/")
38 webbrowser.open("https://www.brandbucket.com/")
39 webbrowser.open("https://edition.cnn.com/")
40 #webbrowser.open("https://www.weebly.com/")
41 webbrowser.open("https://creativecommons.org/")
42 webbrowser.open("https://www.forbes.com/")
43 webbrowser.open("https://www.theguardian.com/")
44 webbrowser.open("https://www.washingtonpost.com/")
45 webbrowser.open("https://www.dropbox.com/")
46 webbrowser.open("https://mail.ru/")
47 webbrowser.open("http://gnu.org/")
48 webbrowser.open("http://gizmodo.com/")
49
50 #website_list_3
51 webbrowser.open("https://myspace.com/")
52 #webbrowser.open("https://www.reuters.com/")
53 webbrowser.open("https://www.live.com/")
54 webbrowser.open("https://medium.com/")
55 webbrowser.open("https://www.msn.com")
56 webbrowser.open("https://www.opera.com/")
57 webbrowser.open("https://uk.yahoo.com/")
58 webbrowser.open("https://www.dailymotion.com/gb")
59 webbrowser.open("https://line.me/en/")
60 webbrowser.open("https://www.who.int/")
61 webbrowser.open("http://dailymail.co.uk/")
62 #webbrowser.open("http://rakuten.co.jp/")
63 webbrowser.open("http://telegraph.co.uk/")
64 webbrowser.open("http://wired.com/")
65 webbrowser.open("http://ft.com/")
66 webbrowser.open("https://twitter.com/")
67 webbrowser.open("https://www.office.com/")
68 webbrowser.open("http://cpanel.com/")
69 webbrowser.open("https://elpais.com/")
70 webbrowser.open("http://fandom.com/")
71 webbrowser.open("http://loc.gov/")
72 webbrowser.open("https://www.washington.edu/")
73
74 #website_list_4
```

```
75  webbrowser.open("https://www.privacyshield.gov/")
76  webbrowser.open("http://independent.co.uk/")
77  webbrowser.open("http://mediafire.com/")
78  webbrowser.open("https://www.thesun.co.uk/")
79  webbrowser.open("https://www.aol.com/")
80  webbrowser.open("http://huffingtonpost.com/")
81  webbrowser.open("http://booking.com/")
82  webbrowser.open("http://wsj.com/")
83  webbrowser.open("http://huffpost.com/")
84  webbrowser.open("https://bitly.com/")
85  webbrowser.open("http://android.com/")
86  webbrowser.open("http://steampowered.com/")
87  webbrowser.open("http://bloomberg.com/")
88  webbrowser.open("http://ebay.com/")
89  webbrowser.open("http://soundcloud.com/")
90  webbrowser.open("https://ok.ru/")
91  webbrowser.open("http://amazon.de/")
92  webbrowser.open("http://aliexpress.com/")
93  webbrowser.open("http://tinyurl.com/")
94  webbrowser.open("http://foxnews.com/")
95
96  #website_list_5
97  #webbrowser.open("http://samsung.com/")
98  #webbrowser.open("http://nasa.gov/")
99  #webbrowser.open("http://aboutads.info/")
100 webbrowser.open("http://businessinsider.com/")
101 webbrowser.open("http://time.com/")
102 webbrowser.open("http://cnet.com/")
103 webbrowser.open("http://abcnews.go.com/")
104 webbrowser.open("https://en.gravatar.com/")
105 webbrowser.open("http://usatoday.com/")
106 webbrowser.open("http://scribd.com/")
107 webbrowser.open("http://www.gov.uk/")
108 webbrowser.open("http://archive.org/")
109 webbrowser.open("http://mirror.co.uk/")
110 webbrowser.open("http://cbsnews.com/")
111 webbrowser.open("http://dan.com/")
112 webbrowser.open("https://www.plesk.com/")
113 webbrowser.open("https://www.cdc.gov/")
114 webbrowser.open("https://discord.com/")
115 webbrowser.open("http://berkeley.edu/")
116 #webbrowser.open("http://yelp.com/")
117 #webbrowser.open("http://quora.com/")
118 webbrowser.open("http://hollywoodreporter.com/")
119 webbrowser.open("https://ria.ru/")
```

```
120 webbrowser.open("https://www.theguardian.com/")
121 webbrowser.open("https://www.netflix.com/")
```

Appendix B

Statistics

B.1 Descriptive Statistics of Audio

Table B.1: Descriptive Statistics of Audio

Mean	Tor	nonTor	SD	Tor	nonTor	Min	Tor	nonTor	Max	Tor	nonTor
F_0	70.48	153.43	**F_0**	18.04	48.30	**F_0**	2	0	**F_0**	190	557
F_1	71.24	153.38	**F_1**	18.02	47.92	**F_1**	1	0	**F_1**	185	229
F_2	70.49	153.48	**F_2**	18.10	48.34	**F_2**	1	0	**F_2**	199	407
F_3	70.50	153.93	**F_3**	17.99	49.50	**F_3**	1	0	**F_3**	203	714
F_4	70.09	153.41	**F_4**	17.79	48.10	**F_4**	2	0	**F_4**	196	247
F_5	70.85	153.23	**F_5**	17.89	48.06	**F_5**	1	0	**F_5**	209	232
F_6	71.32	153.90	**F_6**	18.03	49.92	**F_6**	1	0	**F_6**	203	761
F_7	70.43	153.26	**F_7**	17.97	48.02	**F_7**	2	0	**F_7**	198	233
F_8	70.25	153.32	**F_8**	17.96	48.12	**F_8**	1	0	**F_8**	196	232
F_9	70.13	153.16	**F_9**	18.04	48.08	**F_9**	2	0	**F_9**	195	237
F_a	69.98	153.15	**F_a**	18.07	48.09	**F_a**	1	0	**F_a**	186	240
F_b	72.14	153.08	**F_b**	17.87	48.19	**F_b**	1	0	**F_b**	189	235
F_c	70.99	153.22	**F_c**	18.07	48.27	**F_c**	0	0	**F_c**	193	241
F_d	70.23	153.02	**F_d**	18.11	48.03	**F_d**	4	0	**F_d**	200	240
F_e	70.62	152.94	**F_e**	18.08	48.00	**F_e**	0	0	**F_e**	196	236
F_f	71.67	153.09	**F_f**	18.01	48.21	**F_f**	4	0	**F_f**	179	281
R_0	6.23	6.28	**R_0**	0.74	1.35	**R_0**	3.03	0.00	**R_0**	15.00	90.00
R_1	6.30	6.26	**R_1**	0.75	0.74	**R_1**	1.52	0.00	**R_1**	15.63	22.73
R_2	6.23	6.25	**R_2**	0.75	0.72	**R_2**	1.52	0.00	**R_2**	13.54	18.18
R_3	6.23	6.28	**R_3**	0.73	0.87	**R_3**	1.04	0.00	**R_3**	9.85	25.22
R_4	6.19	6.26	**R_4**	0.72	0.74	**R_4**	1.79	0.00	**R_4**	10.64	18.18
R_5	6.26	6.25	**R_5**	0.73	0.75	**R_5**	1.56	0.00	**R_5**	9.39	33.33
R_6	6.30	6.27	**R_6**	0.75	0.94	**R_6**	1.04	0.00	**R_6**	12.12	31.25
R_7	6.22	6.25	**R_7**	0.73	0.75	**R_7**	2.63	0.00	**R_7**	9.11	20.31
R_8	6.21	6.25	**R_8**	0.74	0.70	**R_8**	1.72	0.00	**R_8**	12.12	18.18
R_9	6.20	6.24	**R_9**	0.74	0.70	**R_9**	2.08	0.00	**R_9**	9.68	20.83
R_a	6.18	6.24	**R_a**	0.74	0.74	**R_a**	1.56	0.00	**R_a**	12.12	22.73
R_b	6.38	6.23	**R_b**	0.75	0.74	**R_b**	1.67	0.00	**R_b**	10.61	18.52
R_c	6.28	6.25	**R_c**	0.73	0.75	**R_c**	0.00	0.00	**R_c**	11.02	18.18
R_d	6.21	6.23	**R_d**	0.73	0.71	**R_d**	3.05	0.00	**R_d**	13.64	18.18
R_e	6.24	6.23	**R_e**	0.74	0.77	**R_e**	0.00	0.00	**R_e**	9.29	22.73
R_f	6.34	6.23	**R_f**	0.75	0.77	**R_f**	3.79	0.00	**R_f**	16.46	31.82
Total	1131.41	2453	**Total**	256.96	743.65	**Total**	58	12	**Total**	2816	2920
Entropy	3.9898	3.9884	**Entropy**	0.0073	0.0704	**Entropy**	3.6629	0.5690	**Entropy**	3.9992	3.9994

B.2 Descriptive Statistics of Browsing

Table B.2: Descriptive Statistics of Browsing

Mean	Tor	nonTor	SD	Tor	nonTor	Min	Tor	nonTor	Max	Tor	nonTor
F_0	79.88	50.86	F_0	27.35	52.82	F_0	2	0	F_0	187	230
F_1	80.08	41.88	F_1	27.52	52.80	F_1	1	0	F_1	182	228
F_2	79.89	41.82	F_2	27.46	52.78	F_2	0	0	F_2	183	230
F_3	79.56	41.88	F_3	27.31	52.91	F_3	2	0	F_3	184	223
F_4	79.77	41.83	F_4	27.37	52.82	F_4	0	0	F_4	184	220
F_5	80.19	41.82	F_5	27.43	52.90	F_5	0	0	F_5	192	222
F_6	80.10	41.87	F_6	27.40	52.93	F_6	2	0	F_6	181	223
F_7	80.78	41.80	F_7	27.63	52.87	F_7	3	0	F_7	191	221
F_8	79.70	41.80	F_8	27.57	52.83	F_8	2	0	F_8	179	232
F_9	79.73	41.81	F_9	27.45	52.86	F_9	1	0	F_9	195	225
F_a	79.88	41.80	F_a	27.34	52.88	F_a	1	0	F_a	184	219
F_b	80.23	41.78	F_b	27.36	52.87	F_b	1	0	F_b	195	224
F_c	80.18	41.77	F_c	27.51	52.81	F_c	2	0	F_c	187	220
F_d	79.59	41.77	F_d	27.49	52.83	F_d	2	0	F_d	176	223
F_e	79.63	41.73	F_e	27.48	52.85	F_e	2	0	F_e	181	229
F_f	79.53	41.74	F_f	27.30	52.82	F_f	2	0	F_f	183	218
R_0	6.25	12.98	R_0	0.70	12.98	R_0	1.56	0.00	R_0	11.72	37.50
R_1	6.26	5.88	R_1	0.71	5.88	R_1	0.78	0.00	R_1	12.12	18.75
R_2	6.25	5.84	R_2	0.71	5.84	R_2	0.00	0.00	R_2	11.49	18.75
R_3	6.22	5.81	R_3	0.70	5.81	R_3	1.56	0.00	R_3	11.31	20.83
R_4	6.24	5.81	R_4	0.71	5.81	R_4	0.00	0.00	R_4	11.54	16.67
R_5	6.27	5.81	R_5	0.71	5.81	R_5	0.00	0.00	R_5	10.94	18.75
R_6	6.27	5.81	R_6	0.70	5.81	R_6	1.19	0.00	R_6	11.72	16.67
R_7	6.32	5.80	R_7	0.72	5.80	R_7	1.92	0.00	R_7	14.06	20.83
R_8	6.23	5.80	R_8	0.70	5.80	R_8	2.34	0.00	R_8	12.02	17.86
R_9	6.23	5.80	R_9	0.71	5.80	R_9	1.43	0.00	R_9	11.06	18.06
R_a	6.25	5.79	R_a	0.71	5.79	R_a	0.78	0.00	R_a	15.52	18.75
R_b	6.28	5.78	R_b	0.71	5.78	R_b	1.56	0.00	R_b	12.50	19.44
R_c	6.27	5.78	R_c	0.70	5.78	R_c	2.34	0.00	R_c	13.28	18.06
R_d	6.22	5.78	R_d	0.71	5.78	R_d	2.98	0.00	R_d	13.64	17.19
R_e	6.23	5.76	R_e	0.71	5.76	R_e	1.52	0.00	R_e	10.16	22.92
R_f	6.22	5.77	R_f	0.71	5.77	R_f	1.56	0.00	R_f	10.94	18.06
Total	1278.70	677.96	Total	416.65	839.19	Total	58	48	Total	2748	2910
Entropy	3.9898	3.9884	Entropy	0.0073	0.0704	Entropy	3.6629	0.5690	Entropy	3.9992	3.9994

B.3 Descriptive Statistics of Chat

Table B.3: Descriptive Statistics of Chat

Mean	Tor	nonTor	SD	Tor	nonTor	Min	Tor	nonTor	Max	Tor	nonTor
F_0	70.48	153.43	F_0	18.04	48.30	F_0	2	0	F_0	190	557
F_1	71.24	153.38	F_1	18.02	47.92	F_1	1	0	F_1	185	229
F_2	70.49	153.48	F_2	18.10	48.34	F_2	1	0	F_2	199	407
F_3	70.50	153.93	F_3	17.99	49.50	F_3	1	0	F_3	203	714
F_4	70.09	153.41	F_4	17.79	48.10	F_4	2	0	F_4	196	247
F_5	70.85	153.23	F_5	17.89	48.06	F_5	1	0	F_5	209	232
F_6	71.32	153.90	F_6	18.03	49.92	F_6	1	0	F_6	203	761
F_7	70.43	153.26	F_7	17.97	48.02	F_7	2	0	F_7	198	233
F_8	70.25	153.32	F_8	17.96	48.12	F_8	1	0	F_8	196	232
F_9	70.13	153.16	F_9	18.04	48.08	F_9	2	0	F_9	195	237
F_a	69.98	153.15	F_a	18.07	48.09	F_a	1	0	F_a	186	240
F_b	72.14	153.08	F_b	17.87	48.19	F_b	1	0	F_b	189	235
F_c	70.99	153.22	F_c	18.07	48.27	F_c	0	0	F_c	193	241
F_d	70.23	153.02	F_d	18.11	48.03	F_d	4	0	F_d	200	240
F_e	70.62	152.94	F_e	18.08	48.00	F_e	0	0	F_e	196	236
F_f	71.67	153.09	F_f	18.01	48.21	F_f	4	0	F_f	179	281
R_0	6.23	6.28	R_0	0.74	1.35	R_0	3.03	0.00	R_0	15.00	90.00
R_1	6.30	6.26	R_1	0.75	0.74	R_1	1.52	0.00	R_1	15.63	22.73
R_2	6.23	6.25	R_2	0.75	0.72	R_2	1.52	0.00	R_2	13.54	18.18
R_3	6.23	6.28	R_3	0.73	0.87	R_3	1.04	0.00	R_3	9.85	25.22
R_4	6.19	6.26	R_4	0.72	0.74	R_4	1.79	0.00	R_4	10.64	18.18
R_5	6.26	6.25	R_5	0.73	0.75	R_5	1.56	0.00	R_5	9.39	33.33
R_6	6.30	6.27	R_6	0.75	0.94	R_6	1.04	0.00	R_6	12.12	31.25
R_7	6.22	6.25	R_7	0.73	0.75	R_7	2.63	0.00	R_7	9.11	20.31
R_8	6.21	6.25	R_8	0.74	0.70	R_8	1.72	0.00	R_8	12.12	18.18
R_9	6.20	6.24	R_9	0.74	0.70	R_9	2.08	0.00	R_9	9.68	20.83
R_a	6.18	6.24	R_a	0.74	0.74	R_a	1.56	0.00	R_a	12.12	22.73
R_b	6.38	6.23	R_b	0.75	0.74	R_b	1.67	0.00	R_b	10.61	18.52
R_c	6.28	6.25	R_c	0.73	0.75	R_c	0.00	0.00	R_c	11.02	18.18
R_d	6.21	6.23	R_d	0.73	0.71	R_d	3.05	0.00	R_d	13.64	18.18
R_e	6.24	6.23	R_e	0.74	0.77	R_e	0.00	0.00	R_e	9.29	22.73
R_f	6.34	6.23	R_f	0.75	0.77	R_f	3.79	0.00	R_f	16.46	31.82
Total	1131.41	2453	Total	256.965	743.659	Total	58	12	Total	2816	2920
Entropy	3.9898	3.9884	Entropy	0.0073	0.0704	Entropy	3.6629	0.5690	Entropy	3.9992	3.9994

B.4 Descriptive Statistics of Email

Table B.4: Descriptive Statistics of Email

Mean	Tor	nonTor	SD	Tor	nonTor	Min	Tor	nonTor	Max	Tor	nonTor
F_0	66.44	43.32	F_0	27.73	54.10	F_0	3	5	F_0	197	234
F_1	66.89	29.92	F_1	27.53	54.71	F_1	3	0	F_1	199	213
F_2	66.90	29.77	F_2	27.84	54.64	F_2	3	0	F_2	202	215
F_3	67.66	29.64	F_3	27.88	54.41	F_3	3	0	F_3	208	211
F_4	66.34	29.81	F_4	27.78	54.61	F_4	3	0	F_4	198	216
F_5	66.79	29.75	F_5	27.87	54.78	F_5	2	0	F_5	193	215
F_6	66.82	29.82	F_6	27.76	54.72	F_6	3	0	F_6	199	211
F_7	67.07	29.69	F_7	27.61	54.50	F_7	4	0	F_7	199	211
F_8	66.39	29.85	F_8	27.79	54.88	F_8	3	0	F_8	191	214
F_9	67.18	29.73	F_9	27.75	54.63	F_9	4	0	F_9	194	224
F_a	67.48	29.71	F_a	27.52	54.72	F_a	4	0	F_a	200	211
F_b	67.63	29.72	F_b	27.77	54.74	F_b	4	0	F_b	213	222
F_c	66.99	29.76	F_c	27.63	54.82	F_c	1	0	F_c	188	216
F_d	67.04	29.75	F_d	27.83	54.75	F_d	2	0	F_d	207	220
F_e	66.75	29.66	F_e	27.70	54.64	F_e	2	0	F_e	207	213
F_f	66.40	29.75	F_f	27.56	54.71	F_f	2	0	F_f	192	215
R_0	6.19	17.46	R_0	0.83	7.28	R_0	2.34	5.23	R_0	10.16	40.00
R_1	6.25	5.67	R_1	0.81	2.15	R_1	2.34	0.00	R_1	11.14	17.57
R_2	6.25	5.54	R_2	0.81	2.07	R_2	1.79	0.00	R_2	10.60	16.67
R_3	6.32	5.50	R_3	0.83	2.11	R_3	1.44	0.00	R_3	10.76	15.71
R_4	6.18	5.58	R_4	0.82	2.12	R_4	2.34	0.00	R_4	10.10	18.75
R_5	6.23	5.46	R_5	0.83	2.14	R_5	1.56	0.00	R_5	12.02	16.67
R_6	6.23	5.53	R_6	0.83	2.12	R_6	1.44	0.00	R_6	11.72	16.18
R_7	6.27	5.51	R_7	0.82	2.14	R_7	2.45	0.00	R_7	10.94	16.67
R_8	6.19	5.52	R_8	0.82	2.14	R_8	2.34	0.00	R_8	11.06	15.00
R_9	6.28	5.48	R_9	0.84	2.13	R_9	2.99	0.00	R_9	12.50	16.28
R_a	6.34	5.43	R_a	0.86	2.11	R_a	2.08	0.00	R_a	12.50	15.71
R_b	6.31	5.45	R_b	0.83	2.08	R_b	2.40	0.00	R_b	11.96	16.67
R_c	6.26	5.43	R_c	0.83	2.09	R_c	0.78	0.00	R_c	12.50	17.19
R_d	6.26	5.49	R_d	0.83	2.10	R_d	1.56	0.00	R_d	11.06	15.48
R_e	6.23	5.45	R_e	0.80	2.16	R_e	1.56	0.00	R_e	11.72	15.71
R_f	6.19	5.49	R_f	0.82	2.09	R_f	1.56	0.00	R_f	11.11	14.71
Total	1070.77	489.66	Total	425.08	870.27	Total	128	50	Total	2904	2822
Entropy	3.9873	3.7707	Entropy	0.0104	0.1801	Entropy	3.8500	3.0200	Entropy	3.9988	3.9995

B.5 Descriptive Statistics of FTP

Table B.5: Descriptive Statistics of FTP

Mean	Tor	nonTor	SD	Tor	nonTor	Min	Tor	nonTor	Max	Tor	nonTor
F_0	61.63	159.85	F_0	34.61	52.26	F_0	0	0	F_0	223	550
F_1	62.17	159.81	F_1	34.52	52.25	F_1	1	0	F_1	222	253
F_2	61.90	159.87	F_2	34.69	52.29	F_2	0	0	F_2	221	425
F_3	62.66	159.85	F_3	34.59	52.53	F_3	1	0	F_3	215	864
F_4	62.80	159.82	F_4	34.42	52.25	F_4	1	0	F_4	220	278
F_5	61.62	159.84	F_5	34.59	52.24	F_5	0	0	F_5	223	246
F_6	63.06	159.93	F_6	34.47	52.64	F_6	2	0	F_6	226	799
F_7	62.50	159.84	F_7	34.54	52.25	F_7	1	0	F_7	220	242
F_8	61.61	159.83	F_8	34.61	52.28	F_8	0	0	F_8	217	275
F_9	62.22	159.81	F_9	34.50	52.25	F_9	1	0	F_9	226	239
F_a	62.62	159.81	F_a	34.59	52.30	F_a	1	0	F_a	233	271
F_b	62.55	159.79	F_b	34.55	52.28	F_b	1	0	F_b	222	247
F_c	62.10	159.77	F_c	34.71	52.28	F_c	0	0	F_c	220	243
F_d	61.94	159.79	F_d	34.67	52.27	F_d	0	0	F_d	221	240
F_e	62.49	159.79	F_e	34.55	52.26	F_e	1	0	F_e	223	246
F_f	62.73	159.84	F_f	34.88	52.31	F_f	1	0	F_f	216	404
R_0	6.15	6.25	R_0	0.91	1.03	R_0	0.00	0.00	R_0	15.00	90.00
R_1	6.25	6.25	R_1	0.91	0.96	R_1	0.78	0.00	R_1	14.84	50.00
R_2	6.17	6.25	R_2	0.92	0.95	R_2	0.00	0.00	R_2	14.84	50.00
R_3	6.30	6.25	R_3	0.91	0.97	R_3	0.78	0.00	R_3	14.84	50.00
R_4	6.37	6.25	R_4	0.93	0.94	R_4	1.52	0.00	R_4	16.41	33.33
R_5	6.15	6.25	R_5	0.91	0.94	R_5	0.00	0.00	R_5	14.06	37.50
R_6	6.39	6.25	R_6	0.93	0.98	R_6	1.56	0.00	R_6	16.67	50.00
R_7	6.29	6.25	R_7	0.91	0.95	R_7	0.78	0.00	R_7	16.41	37.50
R_8	6.15	6.25	R_8	0.91	0.95	R_8	0.00	0.00	R_8	13.28	50.00
R_9	6.26	6.25	R_9	0.91	0.94	R_9	0.78	0.00	R_9	14.84	50.00
R_a	6.30	6.25	R_a	0.91	0.95	R_a	0.78	0.00	R_a	17.19	50.00
R_b	6.30	6.25	R_b	0.92	0.95	R_b	0.78	0.00	R_b	15.63	37.50
R_c	6.20	6.25	R_c	0.92	0.97	R_c	0.00	0.00	R_c	13.28	50.00
R_d	6.18	6.25	R_d	0.91	0.95	R_d	0.00	0.00	R_d	14.84	37.50
R_e	6.29	6.25	R_e	0.91	0.95	R_e	0.78	0.00	R_e	15.00	50.00
R_f	6.25	6.25	R_f	0.92	0.95	R_f	0.78	0.00	R_f	13.28	50.00
Total	996.60	2557.25	Total	539.90	812.71	Total	60	0.00	Total	2904	2920
Entropy	3.9843	3.9823	Entropy	0.0143	0.0783	Entropy	3.5988	0.568996	Entropy	3.9993	3.9996

B.6 Descriptive Statistics of P2P

Table B.6: Descriptive Statistics of P2P

Mean	Tor	nonTor	SD	Tor	nonTor	Min	Tor	nonTor	Max	Tor	nonTor
F_0	72.17	149.27	F_0	20.36	101.27	F_0	1	0	F_0	200	2871
F_1	72.68	144.64	F_1	20.40	62.37	F_1	0	0	F_1	209	858
F_2	72.78	144.17	F_2	20.29	62.36	F_2	0	0	F_2	202	809
F_3	71.88	145.66	F_3	20.25	62.96	F_3	1	0	F_3	196	824
F_4	72.05	144.20	F_4	20.37	62.50	F_4	1	0	F_4	206	1071
F_5	71.87	144.76	F_5	20.44	62.52	F_5	1	0	F_5	217	889
F_6	71.51	144.99	F_6	20.35	63.49	F_6	0	0	F_6	204	1080
F_7	71.97	146.63	F_7	20.39	63.60	F_7	1	0	F_7	199	809
F_8	72.79	144.80	F_8	20.21	62.85	F_8	0	0	F_8	200	824
F_9	71.64	144.90	F_9	20.37	62.55	F_9	0	0	F_9	206	829
F_a	72.39	144.39	F_a	20.33	62.69	F_a	1	0	F_a	202	808
F_b	72.84	145.68	F_b	20.36	63.03	F_b	2	0	F_b	202	474
F_c	71.55	145.45	F_c	20.35	62.68	F_c	1	0	F_c	213	501
F_d	71.94	145.62	F_d	20.32	62.95	F_d	0	0	F_d	211	366
F_e	72.02	146.51	F_e	20.40	63.55	F_e	1	0	F_e	197	415
F_f	72.60	149.28	F_f	20.34	69.95	F_f	1	0	F_f	207	2792
R_0	6.25	7.36	R_0	0.73	8.92	R_0	1.52	0.00	R_0	12.82	98.32
R_1	6.30	6.15	R_1	0.74	1.38	R_1	0.00	0.00	R_1	14.06	50.00
R_2	6.31	6.13	R_2	0.73	1.40	R_2	0.00	0.00	R_2	14.06	50.00
R_3	6.22	6.18	R_3	0.73	1.41	R_3	0.78	0.00	R_3	12.68	50.00
R_4	6.24	6.16	R_4	0.73	1.31	R_4	0.78	0.00	R_4	15.15	50.00
R_5	6.22	6.14	R_5	0.73	1.44	R_5	1.47	0.00	R_5	13.28	50.00
R_6	6.19	6.18	R_6	0.73	1.42	R_6	0.00	0.00	R_6	13.28	50.00
R_7	6.23	6.21	R_7	0.73	1.45	R_7	1.04	0.00	R_7	13.64	50.00
R_8	6.31	6.15	R_8	0.75	1.43	R_8	0.00	0.00	R_8	14.84	50.00
R_9	6.20	6.15	R_9	0.74	1.42	R_9	0.00	0.00	R_9	15.38	50.00
R_a	6.27	6.12	R_a	0.73	1.46	R_a	1.14	0.00	R_a	12.12	50.00
R_b	6.31	6.17	R_b	0.73	1.42	R_b	1.47	0.00	R_b	13.28	50.00
R_c	6.19	6.17	R_c	0.73	1.41	R_c	1.04	0.00	R_c	15.00	50.00
R_d	6.23	6.21	R_d	0.73	1.34	R_d	0.00	0.00	R_d	12.50	50.00
R_e	6.24	6.20	R_e	0.73	1.45	R_e	0.78	0.00	R_e	14.84	100.00
R_f	6.29	6.32	R_f	0.73	2.27	R_f	1.44	0.00	R_f	14.06	100.00
Total	1154.70	2330.96	Total	297.86	976.01	Total	52	2	Total	2830	2920
Entropy	3.9900	3.9262	Entropy	0.0066	0.3543	Entropy	3.6465	0.0000	Entropy	3.9993	3.9999

B.7 Descriptive Statistics of VDO

Table B.7: Descriptive Statistics of VDO

Mean	Tor	nonTor	SD	Tor	nonTor	Min	Tor	nonTor	Max	Tor	nonTor
F_0	71.29	51.61	F_0	19.06	60.10	F_0	4	0	F_0	172	227
F_1	71.60	47.68	F_1	18.99	61.35	F_1	3	0	F_1	175	227
F_2	71.00	47.60	F_2	18.99	61.26	F_2	3	0	F_2	175	221
F_3	71.81	47.68	F_3	19.14	61.41	F_3	3	0	F_3	173	219
F_4	71.40	47.58	F_4	19.04	61.27	F_4	3	0	F_4	175	225
F_5	71.81	47.60	F_5	19.04	61.26	F_5	4	0	F_5	171	229
F_6	71.24	47.56	F_6	18.94	61.25	F_6	2	0	F_6	170	228
F_7	71.81	47.72	F_7	19.05	61.51	F_7	2	0	F_7	174	223
F_8	70.87	47.63	F_8	18.92	61.37	F_8	3	0	F_8	173	224
F_9	72.27	47.64	F_9	19.06	61.52	F_9	4	0	F_9	169	223
F_a	71.17	47.54	F_a	19.13	61.19	F_a	4	0	F_a	173	221
F_b	71.06	47.60	F_b	18.94	61.32	F_b	3	0	F_b	167	226
F_c	70.72	47.51	F_c	19.04	61.29	F_c	3	0	F_c	177	222
F_d	71.29	47.56	F_d	19.06	61.34	F_d	3	0	F_d	168	223
F_e	72.81	47.69	F_e	18.99	61.38	F_e	3	0	F_e	171	223
F_f	72.08	47.55	F_f	19.00	61.36	F_f	3	0	F_f	166	223
R_0	6.23	9.41	R_0	0.73	6.41	R_0	2.38	0.00	R_0	12.50	36.36
R_1	6.26	6.09	R_1	0.74	1.78	R_1	1.79	0.00	R_1	9.39	16.41
R_2	6.21	6.06	R_2	0.75	1.77	R_2	1.79	0.00	R_2	11.31	14.38
R_3	6.27	6.07	R_3	0.74	1.79	R_3	1.79	0.00	R_3	12.50	15.63
R_4	6.24	6.05	R_4	0.74	1.77	R_4	1.79	0.00	R_4	13.69	15.85
R_5	6.28	6.05	R_5	0.74	1.78	R_5	2.38	0.00	R_5	11.31	18.18
R_6	6.23	6.05	R_6	0.73	1.79	R_6	1.19	0.00	R_6	11.31	15.63
R_7	6.28	6.02	R_7	0.75	1.78	R_7	1.19	0.00	R_7	14.88	16.67
R_8	6.19	6.05	R_8	0.72	1.79	R_8	2.34	0.00	R_8	9.52	15.63
R_9	6.32	6.01	R_9	0.73	1.79	R_9	2.38	0.00	R_9	10.94	16.67
R_a	6.22	6.02	R_a	0.74	1.79	R_a	2.98	0.00	R_a	12.50	16.67
R_b	6.21	6.03	R_b	0.74	1.78	R_b	1.79	0.00	R_b	12.50	17.19
R_c	6.18	6.00	R_c	0.74	1.78	R_c	1.79	0.00	R_c	11.90	15.15
R_d	6.23	6.02	R_d	0.74	1.79	R_d	2.34	0.00	R_d	10.71	17.71
R_e	6.37	6.07	R_e	0.74	1.80	R_e	1.79	0.00	R_e	10.94	19.51
R_f	6.30	6.01	R_f	0.73	1.78	R_f	1.79	0.00	R_f	10.12	15.85
Total	1144.22	765.76	Total	274.64	973.98	Total	128	48	Total	2224	2880
Entropy	3.9899	3.9021	Entropy	0.0060	0.1251	Entropy	3.8286	3.2440	Entropy	3.9993	3.9996

B.8 Descriptive Statistics of VoIP

Table B.8: Descriptive Statistics of VoIP

Mean	Tor	nonTor	SD	Tor	nonTor	Min	Tor	nonTor	Max	Tor	nonTor
F_0	68.22	25.07	F_0	11.99	28.20	F_0	2	0	F_0	175	669
F_1	68.21	15.64	F_1	11.93	14.64	F_1	1	0	F_1	177	247
F_2	68.69	15.56	F_2	12.17	14.25	F_2	1	0	F_2	177	251
F_3	68.16	14.97	F_3	11.94	14.05	F_3	1	0	F_3	181	227
F_4	68.85	15.56	F_4	11.99	15.11	F_4	1	0	F_4	180	283
F_5	68.82	14.37	F_5	12.04	13.43	F_5	2	0	F_5	178	215
F_6	68.08	15.05	F_6	11.92	14.38	F_6	1	0	F_6	178	325
F_7	69.13	14.72	F_7	11.89	14.55	F_7	2	0	F_7	176	257
F_8	69.27	14.42	F_8	11.90	13.10	F_8	1	0	F_8	175	210
F_9	68.24	15.44	F_9	11.95	14.75	F_9	2	0	F_9	172	254
F_a	68.09	14.23	F_a	11.99	13.57	F_a	2	0	F_a	178	220
F_b	68.77	14.37	F_b	11.98	13.93	F_b	2	0	F_b	175	216
F_c	68.33	13.82	F_c	11.95	12.85	F_c	2	0	F_c	175	208
F_d	68.84	14.11	F_d	11.94	13.66	F_d	2	0	F_d	179	212
F_e	67.83	14.68	F_e	11.93	13.96	F_e	2	0	F_e	175	232
F_f	68.18	13.64	F_f	11.94	12.65	F_f	2	0	F_f	178	206
R_0	6.23	10.42	R_0	0.73	6.72	R_0	2.34	0.00	R_0	10.94	100.00
R_1	6.23	6.37	R_1	0.73	1.99	R_1	0.78	0.00	R_1	10.94	50.00
R_2	6.27	6.40	R_2	0.74	2.00	R_2	0.78	0.00	R_2	16.67	62.50
R_3	6.22	6.06	R_3	0.73	1.98	R_3	1.52	0.00	R_3	11.72	50.00
R_4	6.28	6.29	R_4	0.73	2.24	R_4	1.52	0.00	R_4	11.06	50.00
R_5	6.28	5.85	R_5	0.74	2.02	R_5	2.34	0.00	R_5	10.50	50.00
R_6	6.21	6.08	R_6	0.73	2.04	R_6	0.78	0.00	R_6	10.94	66.67
R_7	6.31	5.87	R_7	0.74	2.09	R_7	2.34	0.00	R_7	13.28	50.00
R_8	6.32	5.89	R_8	0.74	2.03	R_8	1.52	0.00	R_8	12.50	62.50
R_9	6.23	6.30	R_9	0.74	2.11	R_9	2.34	0.00	R_9	11.72	40.00
R_a	6.21	5.74	R_a	0.73	2.10	R_a	2.34	0.00	R_a	10.94	50.00
R_b	6.28	5.80	R_b	0.73	2.09	R_b	1.56	0.00	R_b	13.28	50.00
R_c	6.24	5.67	R_c	0.73	2.12	R_c	2.34	0.00	R_c	10.94	50.00
R_d	6.28	5.72	R_d	0.73	2.14	R_d	1.56	0.00	R_d	12.50	50.00
R_e	6.19	5.95	R_e	0.73	2.04	R_e	1.56	0.00	R_e	13.64	50.00
R_f	6.22	5.58	R_f	0.73	2.13	R_f	1.56	0.00	R_f	12.50	50.00
Total	1095.69	1095.69	Total	142.34	224.11	Total	66	2	Total	2264	2920
Entropy	3.9900	3.9900	Entropy	0.0039	0.1748	Entropy	3.7344	0.0000	Entropy	3.9992	3.9985

B.9 Descriptive Statistics of Private Browsing

Table B.9: Descriptive Statistics of Private Browsing

Mean	Tor	nonTor	SD	Tor	nonTor	Min	Tor	nonTor	Max	Tor	nonTor
F_0	75.54	47.92	F_0	29.42	64.64	F_0	5.00	0	F_0	549	1110
F_1	75.38	46.38	F_1	29.38	64.65	F_1	4.00	0	F_1	536	1059
F_2	75.35	46.45	F_2	29.31	64.71	F_2	4.00	0	F_2	569	1093
F_3	75.40	46.33	F_3	29.45	64.54	F_3	6.00	0	F_3	527	1055
F_4	75.58	46.25	F_4	29.59	64.48	F_4	4.00	0	F_4	555	1063
F_5	75.50	46.40	F_5	29.44	64.68	F_5	4.00	0	F_5	567	1005
F_6	75.60	46.48	F_6	29.57	64.94	F_6	1.00	0	F_6	555	1089
F_7	75.52	46.34	F_7	29.56	64.69	F_7	3.00	0	F_7	552	1038
F_8	75.48	46.29	F_8	29.84	64.59	F_8	4.00	0	F_8	571	1049
F_9	75.50	46.34	F_9	29.60	64.45	F_9	4.00	0	F_9	555	1006
F_a	75.50	46.25	F_a	29.42	64.47	F_a	5.00	0	F_a	548	1107
F_b	75.38	46.37	F_b	29.33	64.45	F_b	5.00	0	F_b	549	1089
F_c	75.38	46.39	F_c	29.46	64.70	F_c	4.00	0	F_c	523	1103
F_d	75.58	46.34	F_d	29.41	64.61	F_d	5.00	0	F_d	564	1045
F_e	75.50	46.38	F_e	29.51	64.75	F_e	3.00	0	F_e	566	1057
F_f	75.50	46.35	F_f	29.67	64.63	F_f	4.00	0	F_f	563	1076
R_0	6.26	6.93	R_0	0.72	3.55	R_0	3.09	0.00	R_0	10.49	34.85
R_1	6.24	6.19	R_1	0.72	2.10	R_1	2.47	0.00	R_1	9.26	21.05
R_2	6.24	6.23	R_2	0.73	2.13	R_2	2.47	0.00	R_2	11.11	22.06
R_3	6.24	6.23	R_3	0.72	2.14	R_3	3.58	0.00	R_3	9.88	23.68
R_4	6.26	6.17	R_4	0.72	2.14	R_4	2.47	0.00	R_4	9.88	21.05
R_5	6.25	6.18	R_5	0.72	2.09	R_5	2.47	0.00	R_5	10.49	21.15
R_6	6.26	6.19	R_6	0.72	2.11	R_6	0.62	0.00	R_6	11.11	23.68
R_7	6.25	6.21	R_7	0.73	2.13	R_7	1.85	0.00	R_7	12.96	21.21
R_8	6.25	6.16	R_8	0.73	2.09	R_8	2.47	0.00	R_8	9.89	26.32
R_9	6.25	6.23	R_9	0.72	2.09	R_9	2.47	0.00	R_9	10.49	21.15
R_a	6.25	6.20	R_a	0.72	2.10	R_a	3.09	0.00	R_a	9.88	21.15
R_b	6.24	6.24	R_b	0.73	2.11	R_b	3.09	0.00	R_b	11.11	21.05
R_c	6.24	6.22	R_c	0.72	2.12	R_c	2.47	0.00	R_c	9.88	23.68
R_d	6.26	6.21	R_d	0.72	2.09	R_d	3.09	0.00	R_d	10.49	21.05
R_e	6.25	6.19	R_e	0.72	2.10	R_e	1.85	0.00	R_e	11.73	26.32
R_f	6.25	6.21	R_f	0.73	2.15	R_f	2.47	0.00	R_f	9.88	23.68
Total	1207.69	743.25	Total	452.31	1028.49	Total	162	36	Total	8130	17044
Entropy	3.9903	3.9053	Entropy	0.0049	0.1136	Entropy	3.8757	3.2332	Entropy	3.9994	3.9995

B.10 Hex Characters Frequency (F_0-F_f)

B.10.1 Frequency

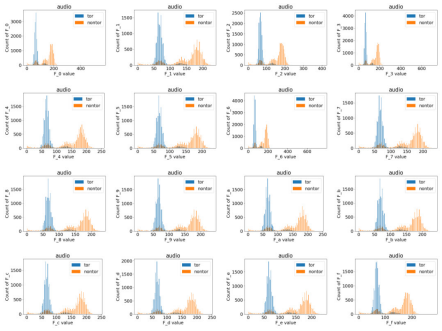


Figure B.1: Histogram of Hex Characters Frequency of Audio

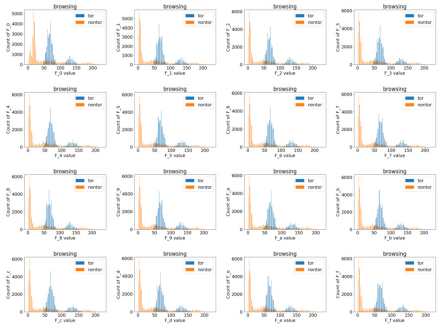


Figure B.2: Histogram of Hex Characters Frequency of Browsing

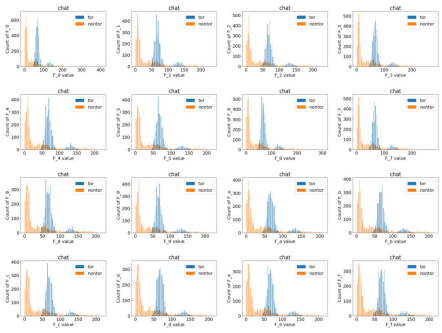


Figure B.3: Histogram of Hex Characters Frequency of Chat

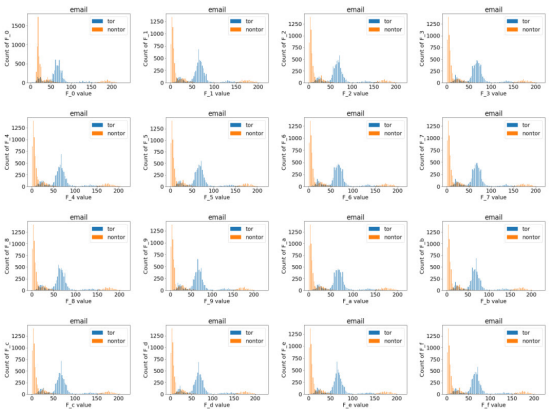


Figure B.4: Histogram of Hex Characters Frequency of Email

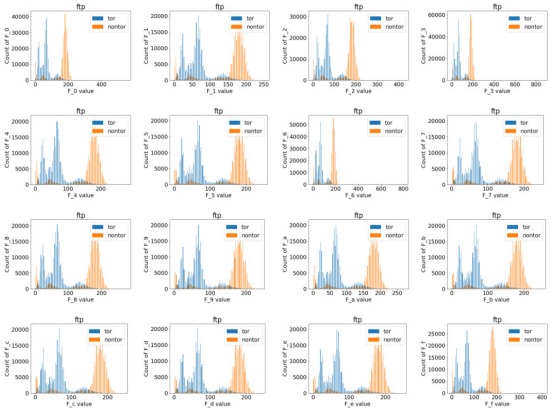


Figure B.5: Histogram of Hex Characters Frequency of FTP

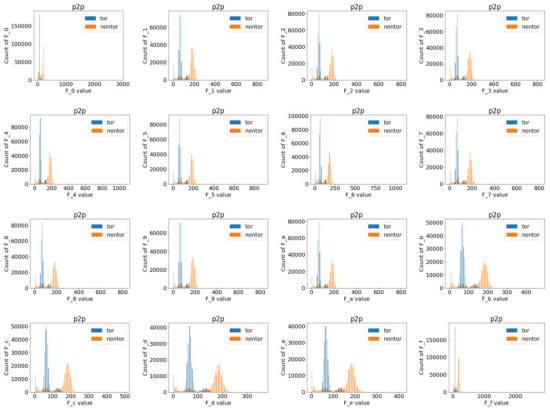


Figure B.6: Histogram of Hex Characters Frequency of P2P

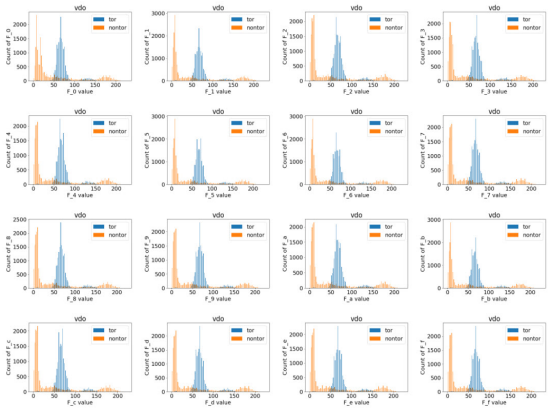


Figure B.7: Histogram of Hex Characters Frequency of VDO

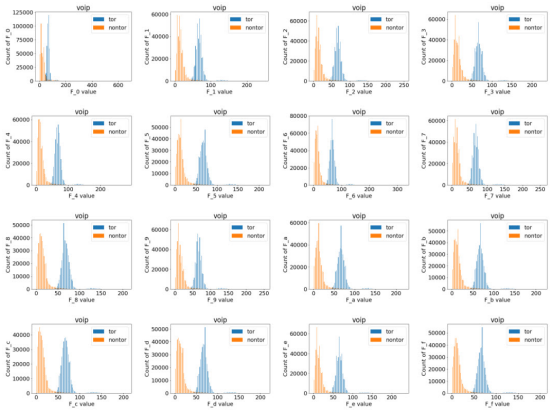


Figure B.8: Histogram of Hex Characters Frequency of VoIP

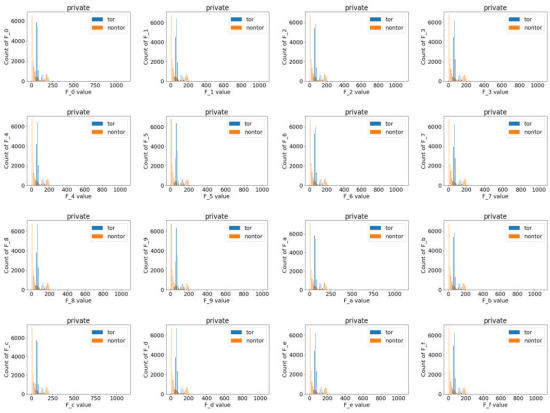


Figure B.9: Histogram of Hex Characters Frequency of Private Browsing

B.10.2 Mean, SD, Min and Max

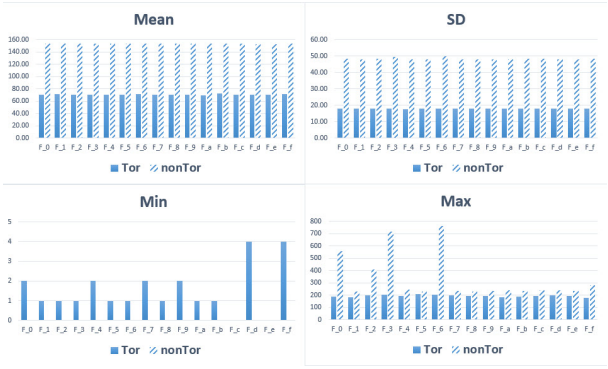


Figure B.10: Descriptive Statistics of Hex Characters Frequency of Audio

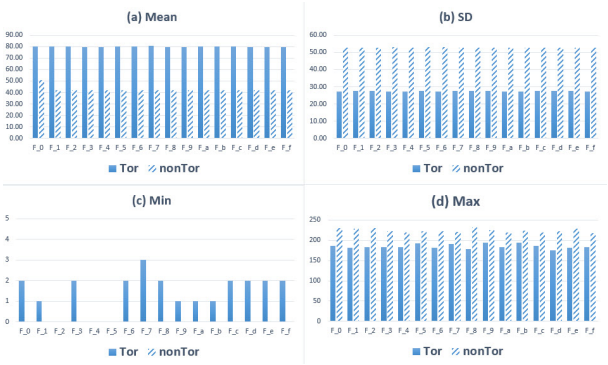


Figure B.11: Descriptive Statistics of Hex Characters Frequency of Browsing

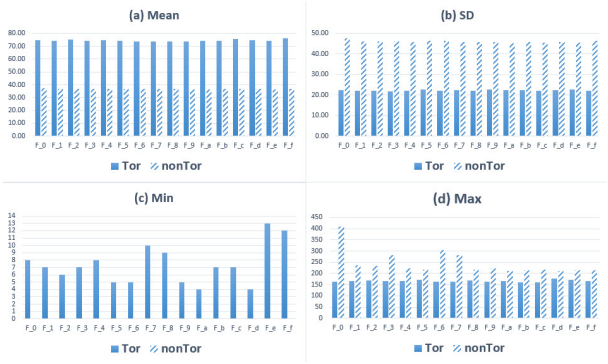


Figure B.12: Descriptive Statistics of Hex Characters Frequency of Chat

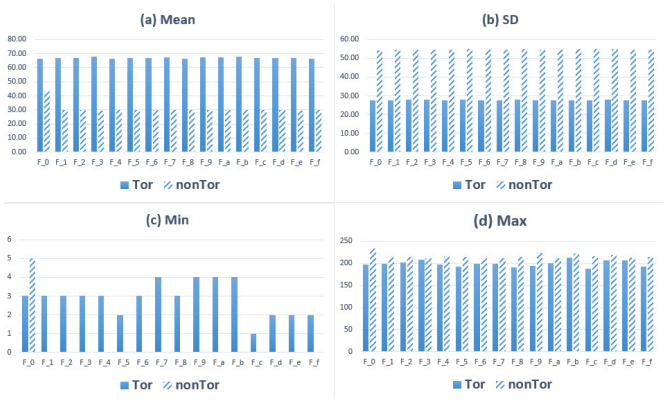


Figure B.13: Descriptive Statistics of Hex Characters Frequency of Email

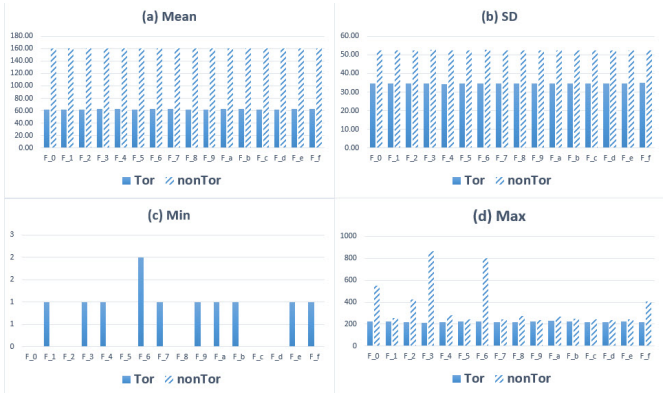


Figure B.14: Descriptive Statistics of Hex Characters Frequency of FTP

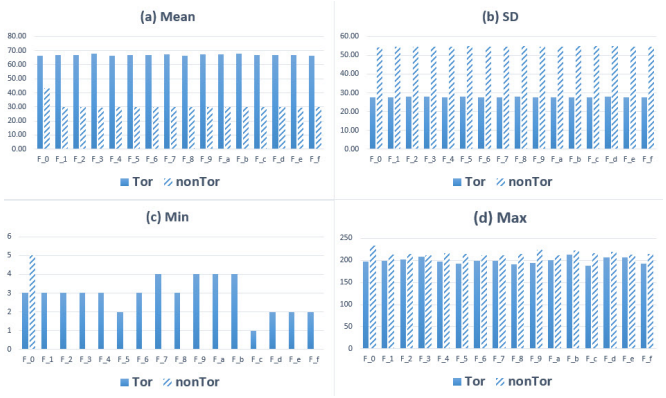


Figure B.15: Descriptive Statistics of Hex Characters Frequency of P2P

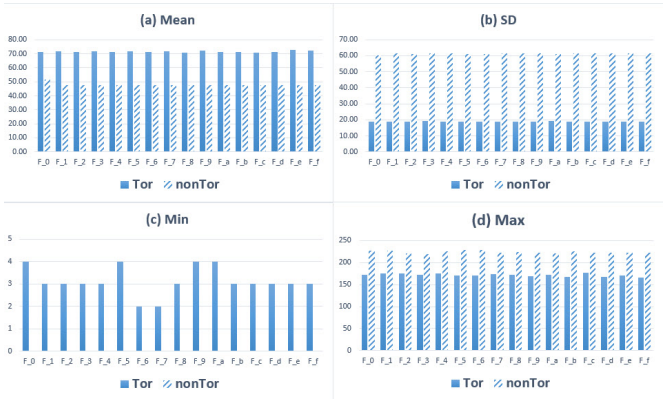


Figure B.16: Descriptive Statistics of Hex Characters Frequency of VDO

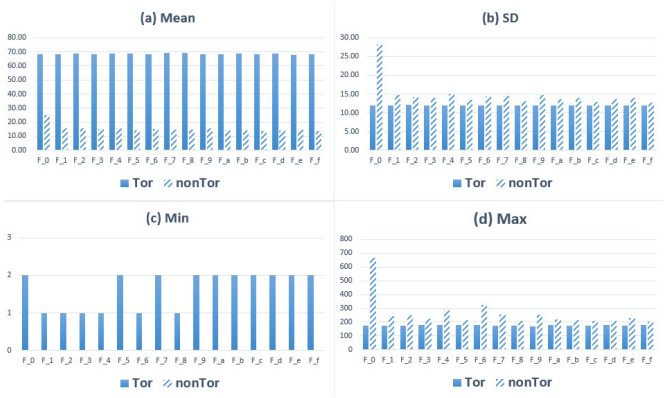


Figure B.17: Descriptive Statistics of Hex Characters Frequency of VoIP

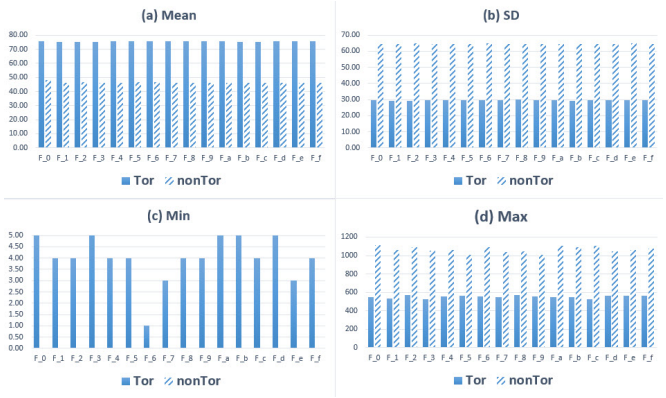


Figure B.18: Descriptive Statistics of Hex Characters Frequency of Private Browsing

B.11 Hex Characters Frequency Ratio (R_0-R_f)

B.11.1 Frequency

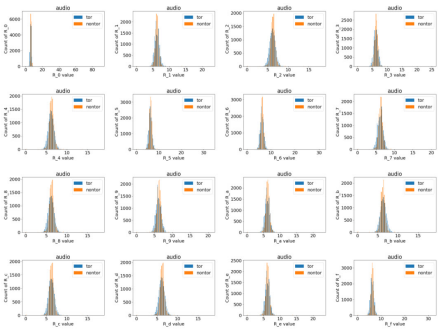


Figure B.19: Histogram of Hex Characters Frequency Ratio of Audio

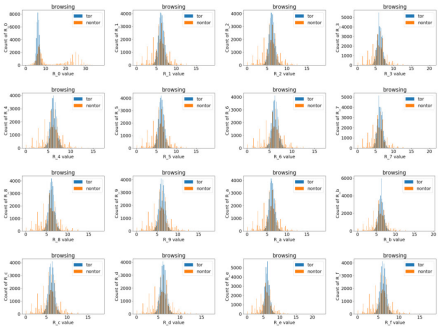


Figure B.20: Histogram of Hex Characters Frequency Ratio of Browsing

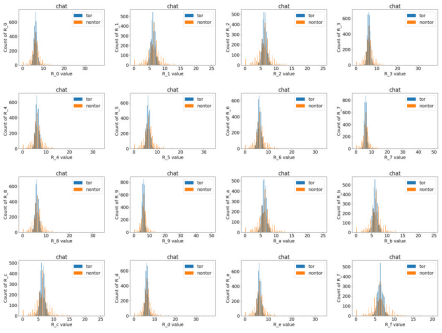


Figure B.21: Histogram of Hex Characters Frequency Ratio of Chat

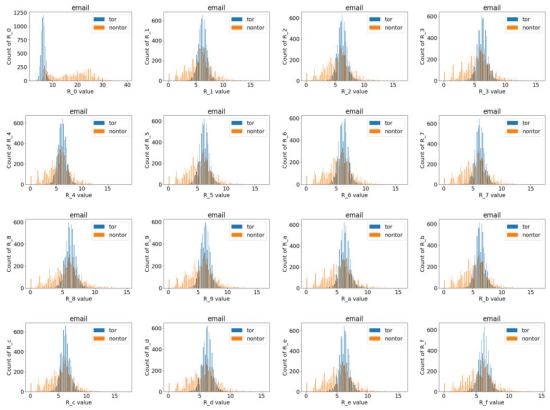


Figure B.22: Histogram of Hex Characters Frequency Ratio of Email

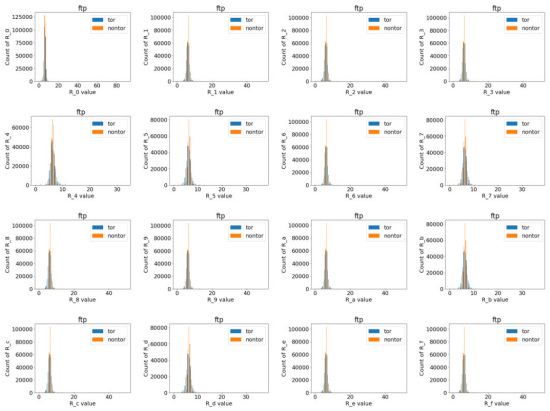


Figure B.23: Histogram of Hex Characters Frequency Ratio of FTP

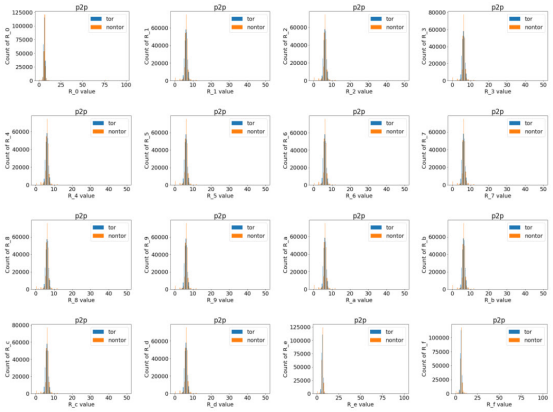


Figure B.24: Histogram of Hex Characters Frequency Ratio of P2P

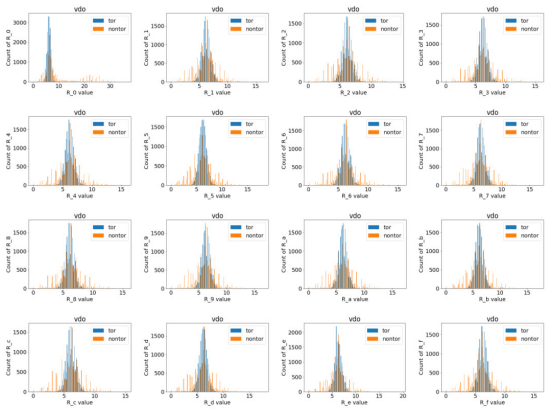


Figure B.25: Histogram of Hex Characters Frequency Ratio of VDO

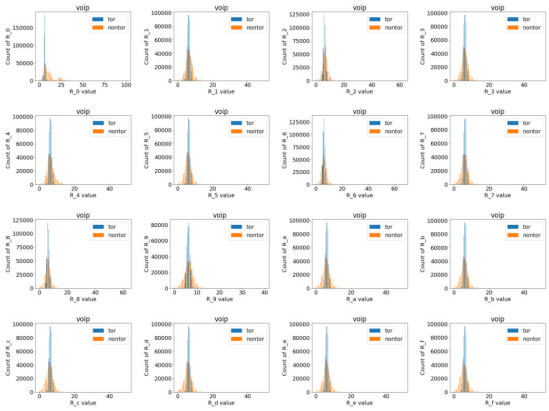


Figure B.26: Histogram of Hex Characters Frequency Ratio of VoIP

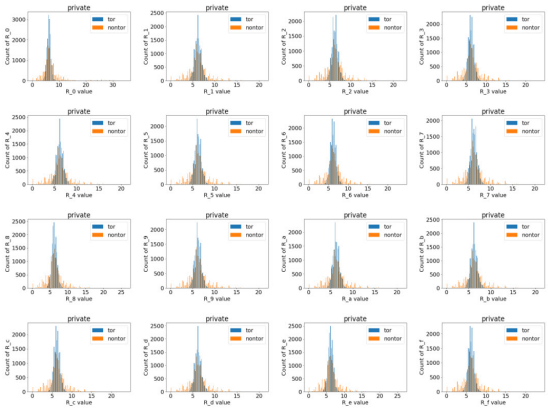


Figure B.27: Histogram of Hex Characters Frequency Ratio of Private Browsing

B.11.2 Mean, SD, Min and Max

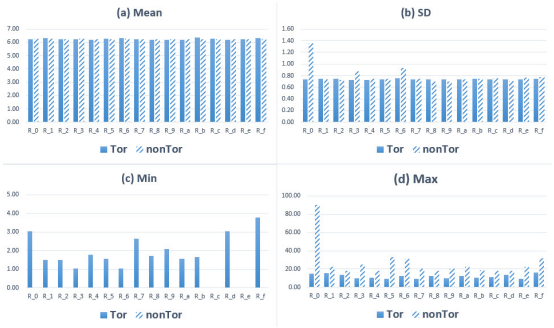


Figure B.28: Descriptive Statistics of Hex Characters Frequency Ratio of Audio

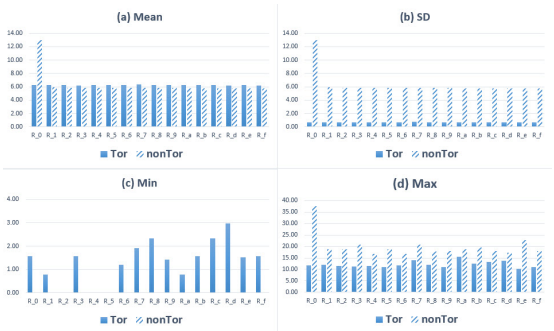


Figure B.29: Descriptive Statistics of Hex Characters Frequency Ratio of Browsing

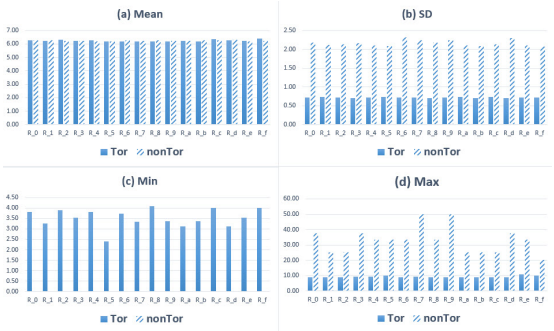


Figure B.30: Descriptive Statistics of Hex Characters Frequency Ratio of Chat

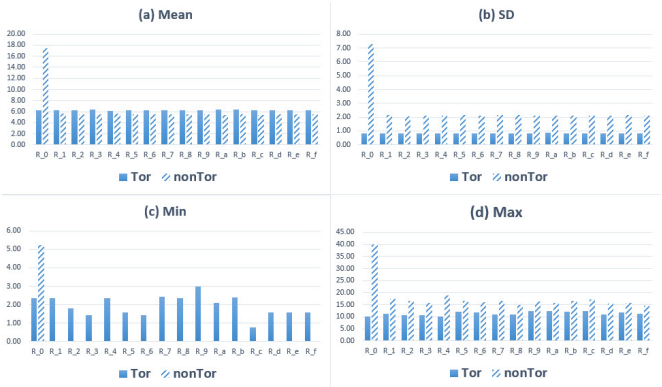


Figure B.31: Descriptive Statistics of Hex Characters Frequency Ratio of Email

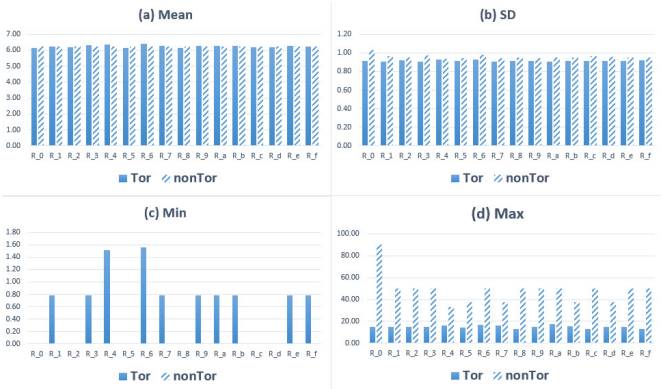


Figure B.32: Descriptive Statistics of Hex Characters Frequency Ratio of FTP

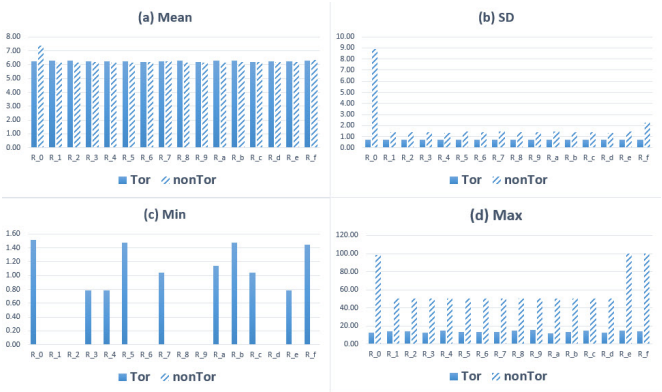


Figure B.33: Descriptive Statistics of Hex Characters Frequency Ratio of P2P

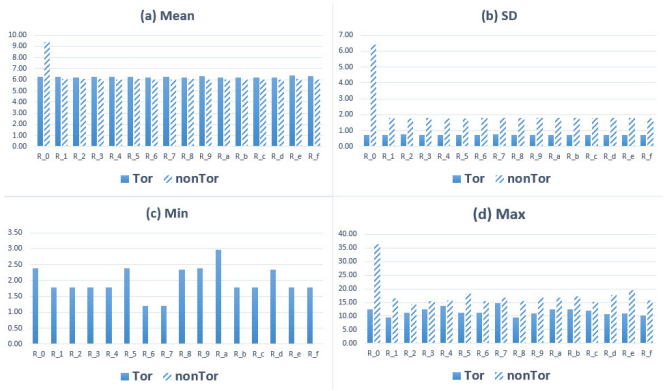


Figure B.34: Descriptive Statistics of Hex Characters Frequency Ratio of VDO

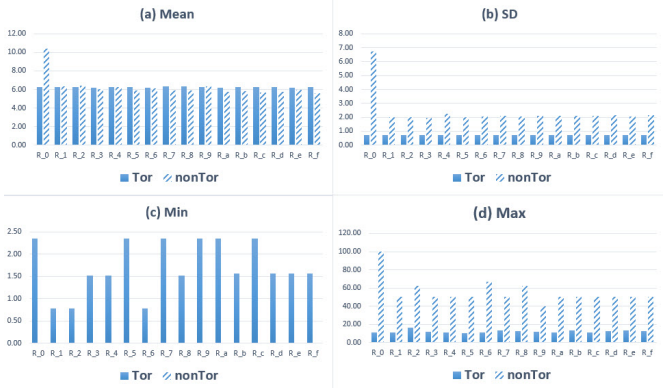


Figure B.35: Descriptive Statistics of Hex Characters Frequency Ratio of VoIP

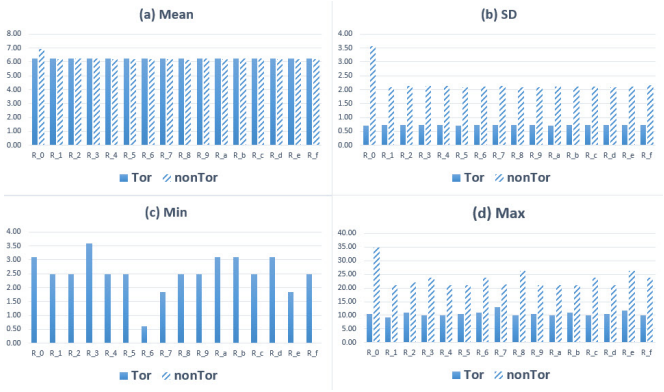


Figure B.36: Descriptive Statistics of Hex Characters Frequency Ratio of Private-Browsing

B.12 Total Characters

B.12.1 Frequency

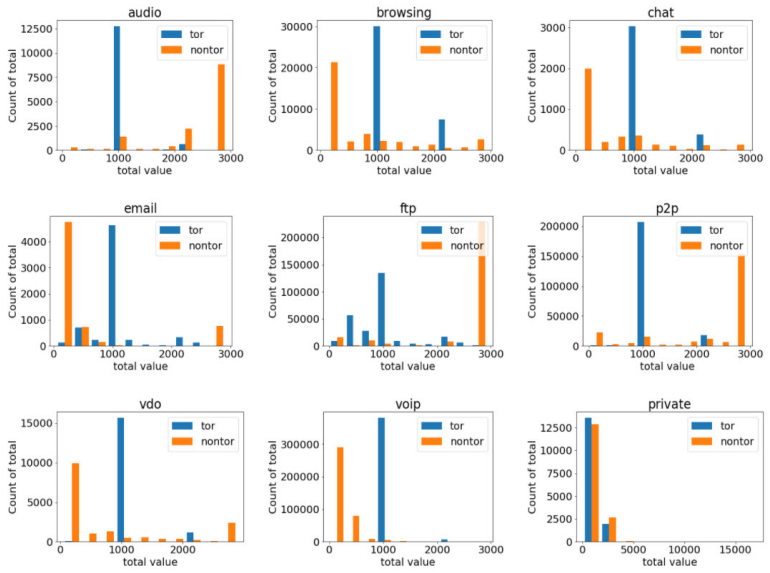


Figure B.37: Histogram of Total Character of each application

B.12.2 Mean, SD, Min and Max

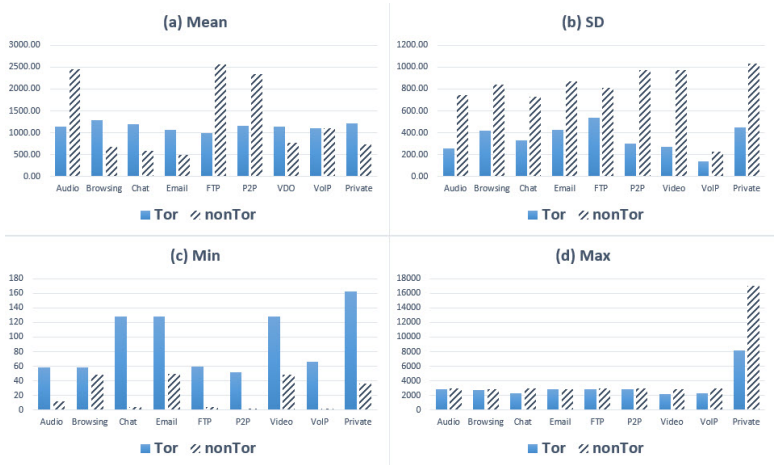


Figure B.38: Descriptive Statistics of Total Character of each application

B.13 Entropy

B.13.1 Frequency

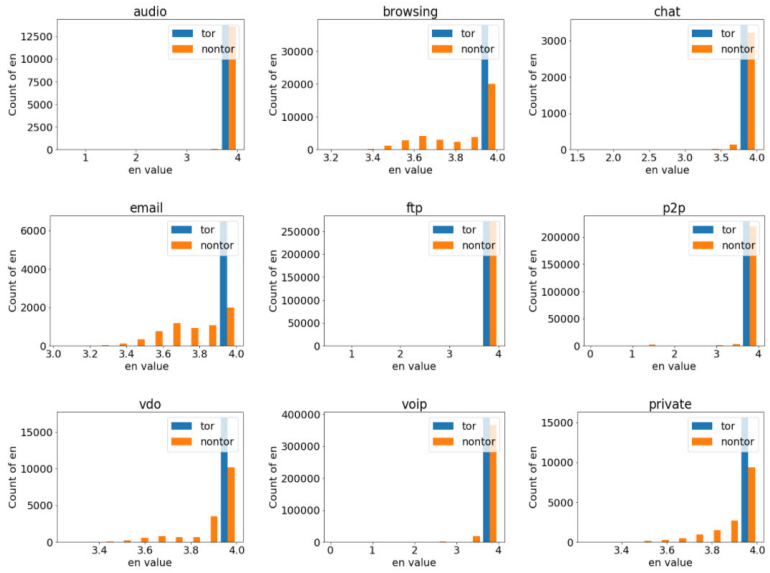


Figure B.39: Histogram of Entropy of each application

B.13.2 Mean, SD, Min and Max

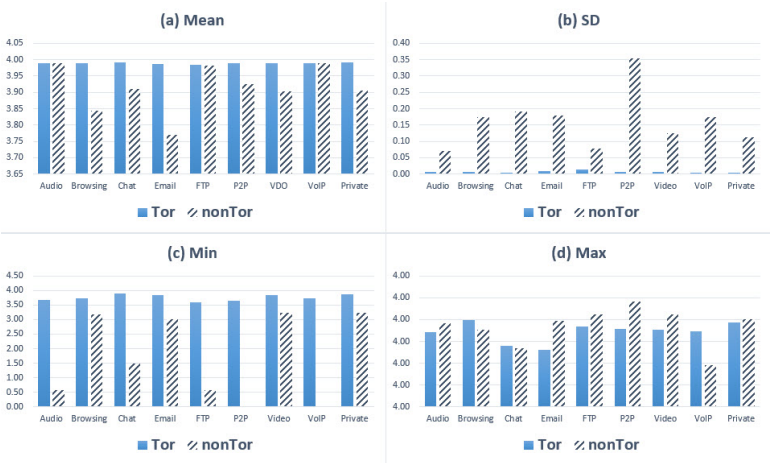


Figure B.40: Descriptive Statistics of Entropy of each application

B.14 Normality Test

Tests of Normality

	Class	Kolmogorov-Smirnov[a] Statistic	df	Sig.
Value_0	Tor	.208	13727	.000
	nonTor	.188	13727	.000
Value_1	Tor	.211	13727	.000
	nonTor	.190	13727	.000
Value_2	Tor	.209	13727	.000
	nonTor	.188	13727	.000
Value_3	Tor	.211	13727	.000
	nonTor	.184	13727	.000
Value_4	Tor	.211	13727	.000
	nonTor	.191	13727	.000
Value_5	Tor	.210	13727	.000
	nonTor	.187	13727	.000
Value_6	Tor	.205	13727	.000
	nonTor	.186	13727	.000
Value_7	Tor	.210	13727	.000
	nonTor	.191	13727	.000
Value_8	Tor	.207	13727	.000
	nonTor	.187	13727	.000
Value_9	Tor	.210	13727	.000
	nonTor	.190	13727	.000
Value_a	Tor	.209	13727	.000
	nonTor	.189	13727	.000
Value_b	Tor	.203	13727	.000
	nonTor	.188	13727	.000
Value_c	Tor	.211	13727	.000
	nonTor	.188	13727	.000
Value_d	Tor	.213	13727	.000
	nonTor	.189	13727	.000
Value_e	Tor	.211	13727	.000
	nonTor	.189	13727	.000
Value_f	Tor	.208	13727	.000
	nonTor	.186	13727	.000
Value_total_char	Tor	.520	13727	.000
	nonTor	.367	13727	.000
Value_entropy	Tor	.164	13727	.000
	nonTor	.439	13727	.000
Value_R_0	Tor	.033	13727	<.001
	nonTor	.224	13727	.000
Value_R_1	Tor	.038	13727	<.001
	nonTor	.100	13727	.000
Value_R_2	Tor	.031	13727	<.001
	nonTor	.088	13727	<.001
Value_R_3	Tor	.031	13727	<.001
	nonTor	.138	13727	.000
Value_R_4	Tor	.034	13727	<.001
	nonTor	.095	13727	.000
Value_R_5	Tor	.028	13727	<.001
	nonTor	.094	13727	.000
Value_R_6	Tor	.031	13727	<.001
	nonTor	.151	13727	.000
Value_R_7	Tor	.026	13727	<.001
	nonTor	.096	13727	.000
Value_R_8	Tor	.033	13727	<.001
	nonTor	.082	13727	<.001
Value_R_9	Tor	.032	13727	<.001
	nonTor	.081	13727	<.001
Value_R_a	Tor	.030	13727	<.001
	nonTor	.094	13727	.000
Value_R_b	Tor	.033	13727	<.001
	nonTor	.101	13727	.000
Value_R_c	Tor	.034	13727	<.001
	nonTor	.100	13727	.000
Value_R_d	Tor	.037	13727	<.001
	nonTor	.088	13727	<.001
Value_R_e	Tor	.031	13727	<.001
	nonTor	.104	13727	.000
Value_R_f	Tor	.035	13727	<.001
	nonTor	.103	13727	.000

a. Lilliefors Significance Correction

Tests of Normality

	Class	Kolmogorov-Smirnov[a] Statistic	df	Sig.
Value_0	Tor	.263	37868	.000
	nonTor	.247	37868	.000
Value_1	Tor	.264	37868	.000
	nonTor	.244	37868	.000
Value_2	Tor	.262	37868	.000
	nonTor	.244	37868	.000
Value_3	Tor	.262	37868	.000
	nonTor	.244	37868	.000
Value_4	Tor	.262	37868	.000
	nonTor	.245	37868	.000
Value_5	Tor	.264	37868	.000
	nonTor	.245	37868	.000
Value_6	Tor	.263	37868	.000
	nonTor	.244	37868	.000
Value_7	Tor	.261	37868	.000
	nonTor	.244	37868	.000
Value_8	Tor	.264	37868	.000
	nonTor	.243	37868	.000
Value_9	Tor	.263	37868	.000
	nonTor	.244	37868	.000
Value_a	Tor	.263	37868	.000
	nonTor	.244	37868	.000
Value_b	Tor	.261	37868	.000
	nonTor	.245	37868	.000
Value_c	Tor	.263	37868	.000
	nonTor	.243	37868	.000
Value_d	Tor	.261	37868	.000
	nonTor	.244	37868	.000
Value_e	Tor	.263	37868	.000
	nonTor	.243	37868	.000
Value_f	Tor	.261	37868	.000
	nonTor	.245	37868	.000
Value_total_char	Tor	.484	37868	.000
	nonTor	.258	37868	.000
Value_entropy	Tor	.122	37868	.000
	nonTor	.194	37868	.000
Value_R_0	Tor	.032	37868	<.001
	nonTor	.256	37868	.000
Value_R_1	Tor	.030	37868	<.001
	nonTor	.100	37868	.000
Value_R_2	Tor	.031	37868	<.001
	nonTor	.102	37868	.000
Value_R_3	Tor	.030	37868	<.001
	nonTor	.102	37868	.000
Value_R_4	Tor	.031	37868	<.001
	nonTor	.101	37868	.000
Value_R_5	Tor	.034	37868	<.001
	nonTor	.100	37868	.000
Value_R_6	Tor	.031	37868	<.001
	nonTor	.104	37868	.000
Value_R_7	Tor	.031	37868	<.001
	nonTor	.103	37868	.000
Value_R_8	Tor	.031	37868	<.001
	nonTor	.103	37868	.000
Value_R_9	Tor	.032	37868	<.001
	nonTor	.104	37868	.000
Value_R_a	Tor	.032	37868	<.001
	nonTor	.104	37868	.000
Value_R_b	Tor	.032	37868	<.001
	nonTor	.104	37868	.000
Value_R_c	Tor	.032	37868	<.001
	nonTor	.104	37868	.000
Value_R_d	Tor	.030	37868	<.001
	nonTor	.105	37868	.000
Value_R_e	Tor	.030	37868	<.001
	nonTor	.103	37868	.000
Value_R_f	Tor	.031	37868	<.001
	nonTor	.104	37868	.000

a. Lilliefors Significance Correction

Figure B.41: Normaility Test of Audio and Browsing

Tests of Normality

	Class	Kolmogorov-Smirnov[a] Statistic	df	Sig.
Value_0	Tor	.257	3423	.000
	nonTor	.229	3423	.000
Value_1	Tor	.251	3423	.000
	nonTor	.228	3423	.000
Value_2	Tor	.250	3423	.000
	nonTor	.231	3423	.000
Value_3	Tor	.255	3423	.000
	nonTor	.233	3423	.000
Value_4	Tor	.253	3423	.000
	nonTor	.235	3423	.000
Value_5	Tor	.258	3423	.000
	nonTor	.231	3423	.000
Value_6	Tor	.250	3423	.000
	nonTor	.226	3423	.000
Value_7	Tor	.254	3423	.000
	nonTor	.230	3423	.000
Value_8	Tor	.252	3423	.000
	nonTor	.231	3423	.000
Value_9	Tor	.261	3423	.000
	nonTor	.228	3423	.000
Value_a	Tor	.253	3423	.000
	nonTor	.229	3423	.000
Value_b	Tor	.259	3423	.000
	nonTor	.232	3423	.000
Value_c	Tor	.246	3423	.000
	nonTor	.230	3423	.000
Value_d	Tor	.259	3423	.000
	nonTor	.229	3423	.000
Value_e	Tor	.258	3423	.000
	nonTor	.228	3423	.000
Value_f	Tor	.250	3423	.000
	nonTor	.232	3423	.000
Value_total_char	Tor	.521	3423	.000
	nonTor	.247	3423	.000
Value_entropy	Tor	.063	3423	<.001
	nonTor	.320	3423	.000
Value_R_0	Tor	.035	3423	<.001
	nonTor	.109	3423	<.001
Value_R_1	Tor	.035	3423	<.001
	nonTor	.104	3423	<.001
Value_R_2	Tor	.035	3423	<.001
	nonTor	.104	3423	<.001
Value_R_3	Tor	.035	3423	<.001
	nonTor	.114	3423	<.001
Value_R_4	Tor	.030	3423	<.001
	nonTor	.113	3423	<.001
Value_R_5	Tor	.025	3423	<.001
	nonTor	.097	3423	<.001
Value_R_6	Tor	.029	3423	<.001
	nonTor	.121	3423	<.001
Value_R_7	Tor	.035	3423	<.001
	nonTor	.115	3423	<.001
Value_R_8	Tor	.032	3423	<.001
	nonTor	.122	3423	<.001
Value_R_9	Tor	.032	3423	<.001
	nonTor	.124	3423	<.001
Value_R_a	Tor	.038	3423	<.001
	nonTor	.106	3423	<.001
Value_R_b	Tor	.035	3423	<.001
	nonTor	.111	3423	<.001
Value_R_c	Tor	.034	3423	<.001
	nonTor	.104	3423	<.001
Value_R_d	Tor	.027	3423	<.001
	nonTor	.129	3423	<.001
Value_R_e	Tor	.033	3423	<.001
	nonTor	.115	3423	<.001
Value_R_f	Tor	.030	3423	<.001
	nonTor	.099	3423	<.001

a. Lilliefors Significance Correction

Tests of Normality

	Class	Kolmogorov-Smirnov[a] Statistic	df	Sig.
Value_0	Tor	.178	6474	.000
	nonTor	.334	6474	.000
Value_1	Tor	.179	6474	.000
	nonTor	.336	6474	.000
Value_2	Tor	.181	6474	.000
	nonTor	.335	6474	.000
Value_3	Tor	.176	6474	.000
	nonTor	.335	6474	.000
Value_4	Tor	.176	6474	.000
	nonTor	.337	6474	.000
Value_5	Tor	.185	6474	.000
	nonTor	.334	6474	.000
Value_6	Tor	.180	6474	.000
	nonTor	.333	6474	.000
Value_7	Tor	.179	6474	.000
	nonTor	.332	6474	.000
Value_8	Tor	.180	6474	.000
	nonTor	.334	6474	.000
Value_9	Tor	.177	6474	.000
	nonTor	.335	6474	.000
Value_a	Tor	.182	6474	.000
	nonTor	.333	6474	.000
Value_b	Tor	.179	6474	.000
	nonTor	.336	6474	.000
Value_c	Tor	.179	6474	.000
	nonTor	.337	6474	.000
Value_d	Tor	.177	6474	.000
	nonTor	.337	6474	.000
Value_e	Tor	.177	6474	.000
	nonTor	.335	6474	.000
Value_f	Tor	.177	6474	.000
	nonTor	.334	6474	.000
Value_total_char	Tor	.349	6474	.000
	nonTor	.353	6474	.000
Value_entropy	Tor	.202	6474	.000
	nonTor	.102	6474	<.001
Value_R_0	Tor	.034	6474	<.001
	nonTor	.090	6474	<.001
Value_R_1	Tor	.036	6474	<.001
	nonTor	.059	6474	<.001
Value_R_2	Tor	.032	6474	<.001
	nonTor	.057	6474	<.001
Value_R_3	Tor	.036	6474	<.001
	nonTor	.055	6474	<.001
Value_R_4	Tor	.031	6474	<.001
	nonTor	.056	6474	<.001
Value_R_5	Tor	.032	6474	<.001
	nonTor	.048	6474	<.001
Value_R_6	Tor	.037	6474	<.001
	nonTor	.056	6474	<.001
Value_R_7	Tor	.038	6474	<.001
	nonTor	.056	6474	<.001
Value_R_8	Tor	.036	6474	<.001
	nonTor	.052	6474	<.001
Value_R_9	Tor	.037	6474	<.001
	nonTor	.045	6474	<.001
Value_R_a	Tor	.048	6474	<.001
	nonTor	.056	6474	<.001
Value_R_b	Tor	.035	6474	<.001
	nonTor	.052	6474	<.001
Value_R_c	Tor	.040	6474	<.001
	nonTor	.057	6474	<.001
Value_R_d	Tor	.035	6474	<.001
	nonTor	.053	6474	<.001
Value_R_e	Tor	.036	6474	<.001
	nonTor	.050	6474	<.001
Value_R_f	Tor	.035	6474	<.001
	nonTor	.052	6474	<.001

a. Lilliefors Significance Correction

Figure B.42: Normaility Test of Chat and Email

Tests of Normality

	Class	Kolmogorov-Smirnov[a] Statistic	df	Sig.
Value_0	Tor	.142	271027	.000
	nonTor	.298	271027	.000
Value_1	Tor	.145	271027	.000
	nonTor	.300	271027	.000
Value_2	Tor	.139	271027	.000
	nonTor	.299	271027	.000
Value_3	Tor	.142	271027	.000
	nonTor	.299	271027	.000
Value_4	Tor	.147	271027	.000
	nonTor	.299	271027	.000
Value_5	Tor	.143	271027	.000
	nonTor	.299	271027	.000
Value_6	Tor	.146	271027	.000
	nonTor	.299	271027	.000
Value_7	Tor	.143	271027	.000
	nonTor	.299	271027	.000
Value_8	Tor	.142	271027	.000
	nonTor	.299	271027	.000
Value_9	Tor	.145	271027	.000
	nonTor	.299	271027	.000
Value_a	Tor	.143	271027	.000
	nonTor	.299	271027	.000
Value_b	Tor	.144	271027	.000
	nonTor	.299	271027	.000
Value_c	Tor	.139	271027	.000
	nonTor	.299	271027	.000
Value_d	Tor	.141	271027	.000
	nonTor	.299	271027	.000
Value_e	Tor	.144	271027	.000
	nonTor	.298	271027	.000
Value_f	Tor	.135	271027	.000
	nonTor	.298	271027	.000
Value_total_char	Tor	.276	271027	.000
	nonTor	.493	271027	.000
Value_entropy	Tor	.189	271027	.000
	nonTor	.433	271027	.000
Value_R_0	Tor	.041	271027	.000
	nonTor	.160	271027	.000
Value_R_1	Tor	.044	271027	.000
	nonTor	.146	271027	.000
Value_R_2	Tor	.042	271027	.000
	nonTor	.143	271027	.000
Value_R_3	Tor	.044	271027	.000
	nonTor	.149	271027	.000
Value_R_4	Tor	.055	271027	.000
	nonTor	.140	271027	.000
Value_R_5	Tor	.040	271027	.000
	nonTor	.142	271027	.000
Value_R_6	Tor	.055	271027	.000
	nonTor	.151	271027	.000
Value_R_7	Tor	.045	271027	.000
	nonTor	.142	271027	.000
Value_R_8	Tor	.040	271027	.000
	nonTor	.143	271027	.000
Value_R_9	Tor	.045	271027	.000
	nonTor	.141	271027	.000
Value_R_a	Tor	.044	271027	.000
	nonTor	.143	271027	.000
Value_R_b	Tor	.047	271027	.000
	nonTor	.143	271027	.000
Value_R_c	Tor	.043	271027	.000
	nonTor	.147	271027	.000
Value_R_d	Tor	.042	271027	.000
	nonTor	.143	271027	.000
Value_R_e	Tor	.046	271027	.000
	nonTor	.143	271027	.000
Value_R_f	Tor	.044	271027	.000
	nonTor	.143	271027	.000

a. Lilliefors Significance Correction

Tests of Normality

	Class	Kolmogorov-Smirnov[a] Statistic	df	Sig.
Value_0	Tor	.232	228300	.000
	nonTor	.236	228300	.000
Value_1	Tor	.231	228300	.000
	nonTor	.237	228300	.000
Value_2	Tor	.231	228300	.000
	nonTor	.231	228300	.000
Value_3	Tor	.230	228300	.000
	nonTor	.244	228300	.000
Value_4	Tor	.232	228300	.000
	nonTor	.231	228300	.000
Value_5	Tor	.234	228300	.000
	nonTor	.236	228300	.000
Value_6	Tor	.233	228300	.000
	nonTor	.235	228300	.000
Value_7	Tor	.233	228300	.000
	nonTor	.239	228300	.000
Value_8	Tor	.226	228300	.000
	nonTor	.236	228300	.000
Value_9	Tor	.231	228300	.000
	nonTor	.240	228300	.000
Value_a	Tor	.232	228300	.000
	nonTor	.236	228300	.000
Value_b	Tor	.233	228300	.000
	nonTor	.241	228300	.000
Value_c	Tor	.232	228300	.000
	nonTor	.246	228300	.000
Value_d	Tor	.231	228300	.000
	nonTor	.240	228300	.000
Value_e	Tor	.232	228300	.000
	nonTor	.239	228300	.000
Value_f	Tor	.232	228300	.000
	nonTor	.211	228300	.000
Value_total_char	Tor	.518	228300	.000
	nonTor	.374	228300	.000
Value_entropy	Tor	.139	228300	.000
	nonTor	.425	228300	.000
Value_R_0	Tor	.032	228300	.000
	nonTor	.421	228300	.000
Value_R_1	Tor	.032	228300	.000
	nonTor	.195	228300	.000
Value_R_2	Tor	.032	228300	.000
	nonTor	.192	228300	.000
Value_R_3	Tor	.031	228300	.000
	nonTor	.206	228300	.000
Value_R_4	Tor	.031	228300	.000
	nonTor	.174	228300	.000
Value_R_5	Tor	.030	228300	.000
	nonTor	.202	228300	.000
Value_R_6	Tor	.030	228300	.000
	nonTor	.194	228300	.000
Value_R_7	Tor	.031	228300	.000
	nonTor	.203	228300	.000
Value_R_8	Tor	.031	228300	.000
	nonTor	.199	228300	.000
Value_R_9	Tor	.031	228300	.000
	nonTor	.204	228300	.000
Value_R_a	Tor	.031	228300	.000
	nonTor	.206	228300	.000
Value_R_b	Tor	.033	228300	.000
	nonTor	.207	228300	.000
Value_R_c	Tor	.030	228300	.000
	nonTor	.208	228300	.000
Value_R_d	Tor	.030	228300	.000
	nonTor	.187	228300	.000
Value_R_e	Tor	.032	228300	.000
	nonTor	.206	228300	.000
Value_R_f	Tor	.031	228300	.000
	nonTor	.243	228300	.000

a. Lilliefors Significance Correction

Tests of Normality

	Class	Kolmogorov-Smirnov[a] Statistic	df	Sig.
Value_0	Tor	.228	16923	.000
	nonTor	.246	16923	.000
Value_1	Tor	.221	16923	.000
	nonTor	.268	16923	.000
Value_2	Tor	.219	16923	.000
	nonTor	.269	16923	.000
Value_3	Tor	.222	16923	.000
	nonTor	.269	16923	.000
Value_4	Tor	.221	16923	.000
	nonTor	.268	16923	.000
Value_5	Tor	.224	16923	.000
	nonTor	.268	16923	.000
Value_6	Tor	.221	16923	.000
	nonTor	.268	16923	.000
Value_7	Tor	.220	16923	.000
	nonTor	.268	16923	.000
Value_8	Tor	.227	16923	.000
	nonTor	.269	16923	.000
Value_9	Tor	.227	16923	.000
	nonTor	.268	16923	.000
Value_a	Tor	.225	16923	.000
	nonTor	.269	16923	.000
Value_b	Tor	.223	16923	.000
	nonTor	.269	16923	.000
Value_c	Tor	.224	16923	.000
	nonTor	.269	16923	.000
Value_d	Tor	.224	16923	.000
	nonTor	.268	16923	.000
Value_e	Tor	.224	16923	.000
	nonTor	.269	16923	.000
Value_f	Tor	.224	16923	.000
	nonTor	.269	16923	.000
Value_total_char	Tor	.527	16923	.000
	nonTor	.277	16923	.000
Value_entropy	Tor	.122	16923	.000
	nonTor	.219	16923	.000
Value_R_0	Tor	.029	16923	<.001
	nonTor	.278	16923	.000
Value_R_1	Tor	.032	16923	<.001
	nonTor	.081	16923	<.001
Value_R_2	Tor	.033	16923	<.001
	nonTor	.084	16923	<.001
Value_R_3	Tor	.034	16923	<.001
	nonTor	.084	16923	<.001
Value_R_4	Tor	.034	16923	<.001
	nonTor	.086	16923	.000
Value_R_5	Tor	.027	16923	<.001
	nonTor	.084	16923	.000
Value_R_6	Tor	.035	16923	<.001
	nonTor	.084	16923	.000
Value_R_7	Tor	.035	16923	<.001
	nonTor	.088	16923	.000
Value_R_8	Tor	.032	16923	<.001
	nonTor	.088	16923	.000
Value_R_9	Tor	.027	16923	<.001
	nonTor	.085	16923	.000
Value_R_a	Tor	.034	16923	<.001
	nonTor	.088	16923	.000
Value_R_b	Tor	.034	16923	<.001
	nonTor	.085	16923	.000
Value_R_c	Tor	.032	16923	<.001
	nonTor	.087	16923	.000
Value_R_d	Tor	.031	16923	<.001
	nonTor	.085	16923	.000
Value_R_e	Tor	.031	16923	<.001
	nonTor	.083	16923	<.001
Value_R_f	Tor	.034	16923	<.001
	nonTor	.085	16923	.000

a. Lilliefors Significance Correction

Figure B.43: Normaility Test of FTP, P2P and VDO

Tests of Normality

	Class	Kolmogorov-Smirnov[a] Statistic	df	Sig.
Value_0	Tor	.129	388096	.000
	nonTor	.198	388096	.000
Value_1	Tor	.127	388096	.000
	nonTor	.181	388096	.000
Value_2	Tor	.130	388096	.000
	nonTor	.186	388096	.000
Value_3	Tor	.127	388096	.000
	nonTor	.183	388096	.000
Value_4	Tor	.128	388096	.000
	nonTor	.194	388096	.000
Value_5	Tor	.128	388096	.000
	nonTor	.191	388096	.000
Value_6	Tor	.128	388096	.000
	nonTor	.187	388096	.000
Value_7	Tor	.125	388096	.000
	nonTor	.191	388096	.000
Value_8	Tor	.125	388096	.000
	nonTor	.174	388096	.000
Value_9	Tor	.126	388096	.000
	nonTor	.170	388096	.000
Value_a	Tor	.128	388096	.000
	nonTor	.182	388096	.000
Value_b	Tor	.128	388096	.000
	nonTor	.183	388096	.000
Value_c	Tor	.127	388096	.000
	nonTor	.176	388096	.000
Value_d	Tor	.127	388096	.000
	nonTor	.183	388096	.000
Value_e	Tor	.127	388096	.000
	nonTor	.179	388096	.000
Value_f	Tor	.127	388096	.000
	nonTor	.181	388096	.000
Value_total_char	Tor	.536	388096	.000
	nonTor	.213	388096	.000
Value_entropy	Tor	.050	388096	.000
	nonTor	.285	388096	.000
Value_R_0	Tor	.032	388096	.000
	nonTor	.158	388096	.000
Value_R_1	Tor	.032	388096	.000
	nonTor	.048	388096	.000
Value_R_2	Tor	.032	388096	.000
	nonTor	.049	388096	.000
Value_R_3	Tor	.032	388096	.000
	nonTor	.048	388096	.000
Value_R_4	Tor	.033	388096	.000
	nonTor	.071	388096	.000
Value_R_5	Tor	.031	388096	.000
	nonTor	.048	388096	.000
Value_R_6	Tor	.032	388096	.000
	nonTor	.051	388096	.000
Value_R_7	Tor	.032	388096	.000
	nonTor	.043	388096	.000
Value_R_8	Tor	.033	388096	.000
	nonTor	.049	388096	.000
Value_R_9	Tor	.032	388096	.000
	nonTor	.041	388096	.000
Value_R_a	Tor	.032	388096	.000
	nonTor	.043	388096	.000
Value_R_b	Tor	.033	388096	.000
	nonTor	.044	388096	.000
Value_R_c	Tor	.032	388096	.000
	nonTor	.042	388096	.000
Value_R_d	Tor	.032	388096	.000
	nonTor	.044	388096	.000
Value_R_e	Tor	.032	388096	.000
	nonTor	.047	388096	.000
Value_R_f	Tor	.032	388096	.000
	nonTor	.040	388096	.000

a. Lilliefors Significance Correction

Tests of Normality

	Class	Kolmogorov-Smirnov[a] Statistic	df	Sig.
Value_0	1	.280	15579	.000
	2	.248	15579	.000
Value_1	1	.277	15579	.000
	2	.259	15579	.000
Value_2	1	.275	15579	.000
	2	.258	15579	.000
Value_3	1	.278	15579	.000
	2	.260	15579	.000
Value_4	1	.279	15579	.000
	2	.260	15579	.000
Value_5	1	.281	15579	.000
	2	.260	15579	.000
Value_6	1	.278	15579	.000
	2	.259	15579	.000
Value_7	1	.278	15579	.000
	2	.260	15579	.000
Value_8	1	.276	15579	.000
	2	.260	15579	.000
Value_9	1	.280	15579	.000
	2	.258	15579	.000
Value_a	1	.278	15579	.000
	2	.258	15579	.000
Value_b	1	.276	15579	.000
	2	.260	15579	.000
Value_c	1	.279	15579	.000
	2	.261	15579	.000
Value_d	1	.279	15579	.000
	2	.259	15579	.000
Value_e	1	.281	15579	.000
	2	.259	15579	.000
Value_f	1	.279	15579	.000
	2	.257	15579	.000
Value_total_char	1	.491	15579	.000
	2	.270	15579	.000
Value_entropy	1	.071	15579	<.001
	2	.204	15579	.000
Value_R_0	1	.030	15579	<.001
	2	.193	15579	.000
Value_R_1	1	.032	15579	<.001
	2	.093	15579	.000
Value_R_2	1	.031	15579	<.001
	2	.091	15579	.000
Value_R_3	1	.031	15579	<.001
	2	.096	15579	.000
Value_R_4	1	.028	15579	<.001
	2	.092	15579	.000
Value_R_5	1	.027	15579	<.001
	2	.091	15579	.000
Value_R_6	1	.031	15579	<.001
	2	.091	15579	.000
Value_R_7	1	.031	15579	<.001
	2	.092	15579	.000
Value_R_8	1	.030	15579	<.001
	2	.089	15579	.000
Value_R_9	1	.026	15579	<.001
	2	.090	15579	.000
Value_R_a	1	.030	15579	<.001
	2	.092	15579	.000
Value_R_b	1	.030	15579	<.001
	2	.092	15579	.000
Value_R_c	1	.029	15579	<.001
	2	.091	15579	.000
Value_R_d	1	.036	15579	<.001
	2	.092	15579	.000
Value_R_e	1	.028	15579	<.001
	2	.089	15579	.000
Value_R_f	1	.029	15579	<.001
	2	.092	15579	.000

a. Lilliefors Significance Correction

Figure B.44: Normaility Test of VoIP and Private Browsing

B.15 Mann Whitney Test

Test Statistics[a]

	Value_0	Value_1	Value_2	Value_3
Mann-Whitney U	18807948.50	19084245.50	18769996.00	18755345.50
Wilcoxon W	113030076.5	113306373.5	112992124.0	112977473.5
Z	-114.859	-114.438	-114.917	-114.939
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_R_2	Value_R_3	Value_R_4	Value_R_5
Mann-Whitney U	92020141.50	91386639.50	88889350.50	93054794.50
Wilcoxon W	186242269.5	185608767.5	183111478.5	187276922.5
Z	-3.343	-4.308	-8.112	-1.768
Asymp. Sig. (2-tailed)	<.001	<.001	<.001	.077

Test Statistics[a]

	Value_4	Value_5	Value_6	Value_7
Mann-Whitney U	18414692.00	19060339.50	19047364.50	18581392.50
Wilcoxon W	112636820.0	113282467.5	113269492.5	112803520.5
Z	-115.458	-114.474	-114.494	-115.204
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_R_6	Value_R_7	Value_R_8	Value_R_9
Mann-Whitney U	90568623.00	91977901.50	89900909.50	89565786.00
Wilcoxon W	184790751.0	186200029.5	184123037.5	183787914.0
Z	-5.554	-3.408	-6.571	-7.082
Asymp. Sig. (2-tailed)	<.001	<.001	<.001	<.001

Test Statistics[a]

	Value_8	Value_9	Value_a	Value_b
Mann-Whitney U	18513082.50	18545011.00	18533334.50	19988254.00
Wilcoxon W	112735210.5	112767139.0	112755462.5	114210382.0
Z	-115.308	-115.259	-115.277	-113.061
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_R_a	Value_R_b	Value_R_c	Value_R_d
Mann-Whitney U	88407409.50	83702788.00	93009203.00	90818941.00
Wilcoxon W	182629537.5	177924916.0	187231331.0	185041069.0
Z	-8.846	-16.012	-1.837	-5.173
Asymp. Sig. (2-tailed)	<.001	<.001	.066	<.001

Test Statistics[a]

	Value_c	Value_d	Value_e	Value_f
Mann-Whitney U	19355398.50	18711359.50	18867906.50	19683512.00
Wilcoxon W	113577526.5	112933487.5	113090034.5	113905640.0
Z	-114.025	-115.006	-114.768	-113.525
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_R_e	Value_R_f
Mann-Whitney U	94051293.50	86948069.50
Wilcoxon W	188273421.5	181170197.5
Z	-.250	-11.069
Asymp. Sig. (2-tailed)	.803	<.001

a. Grouping Variable: Class

Test Statistics[a]

	Value_total_char	Value_entropy	Value_R_0	Value_R_1
Mann-Whitney U	16677032.50	28669646.00	91970222.50	91424307.00
Wilcoxon W	110899160.5	122891774.0	186192350.5	185646435.0
Z	-126.613	-99.827	-3.420	-4.251
Asymp. Sig. (2-tailed)	.000	.000	<.001	<.001

Figure B.45: Mann Whitney Test of Audio

Test Statistics[a]

	Value_0	Value_1	Value_2	Value_3
Mann-Whitney U	346417594.5	302123137.5	302947148.5	304660165.0
Wilcoxon W	1063429241	1019134784	1019958795	1021671811
Z	-123.197	-137.928	-137.655	-137.085
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_R_2	Value_R_3	Value_R_4	Value_R_5
Mann-Whitney U	607091381.0	613093109.5	603891302.5	591416636.0
Wilcoxon W	1324103027	1330104756	1320902949	1308428282
Z	-36.534	-34.539	-37.598	-41.745
Asymp. Sig. (2-tailed)	<.001	<.001	<.001	.000

Test Statistics[a]

	Value_4	Value_5	Value_6	Value_7
Mann-Whitney U	303421138.5	301084200.0	301630012.5	298759106.0
Wilcoxon W	1020432785	1018095846	1018641659	1015770752
Z	-137.497	-138.274	-138.092	-139.047
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_R_6	Value_R_7	Value_R_8	Value_R_9
Mann-Whitney U	597583141.0	573111548.0	605662628.0	604431090.5
Wilcoxon W	1314594787	1290123194	1322674274	1321442737
Z	-39.695	-47.830	-37.009	-37.419
Asymp. Sig. (2-tailed)	.000	.000	<.001	<.001

Test Statistics[a]

	Value_8	Value_9	Value_a	Value_b
Mann-Whitney U	304072356.0	303018247.5	302768946.0	301347782.5
Wilcoxon W	1021084002	1020029894	1019780592	1018359429
Z	-137.280	-137.631	-137.714	-138.187
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_R_a	Value_R_b	Value_R_c	Value_R_d
Mann-Whitney U	599356911.5	580782641.0	584003408.0	604912057.5
Wilcoxon W	1316368558	1297794287	1301015054	1321923704
Z	-39.105	-45.280	-44.209	-37.259
Asymp. Sig. (2-tailed)	.000	.000	.000	<.001

Test Statistics[a]

	Value_c	Value_d	Value_e	Value_f
Mann-Whitney U	300313670.5	304289627.5	303288295.5	304423965.5
Wilcoxon W	1017325317	1021301274	1020299942	1021435612
Z	-138.530	-137.208	-137.541	-137.164
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_R_e	Value_R_f
Mann-Whitney U	596342941.0	600803357.5
Wilcoxon W	1313354587	1317815004
Z	-40.107	-38.625
Asymp. Sig. (2-tailed)	.000	.000

a. Grouping Variable: Class

Test Statistics[a]

	Value_total_char	Value_entropy	Value_R_0	Value_R_1
Mann-Whitney U	298267649.0	287004766.0	322392904.0	612852425.0
Wilcoxon W	1015279295	1004016412	1039404550	1329864071
Z	-143.931	-142.929	-131.175	-34.619
Asymp. Sig. (2-tailed)	.000	.000	.000	<.001

Figure B.46: Mann Whitney Test of Browsing

Test Statistics[a]

	Value_0	Value_1	Value_2	Value_3	Value_4
Mann-Whitney U	2018730.500	2045760.500	1982767.000	2004602.500	1982471.500
Wilcoxon W	7878906.500	7905936.500	7842943.000	7864778.500	7842647.500
Z	-46.968	-46.637	-47.408	-47.141	-47.411
Asymp. Sig. (2-tailed)	.000	.000	.000	.000	.000

Test Statistics[a]

	Value_5	Value_6	Value_7	Value_8	Value_9
Mann-Whitney U	2039181.500	2020978.500	2047010.500	2034004.000	2025076.500
Wilcoxon W	7899357.500	7881154.500	7907186.500	7894180.000	7885252.500
Z	-46.718	-46.940	-46.622	-46.781	-46.890
Asymp. Sig. (2-tailed)	.000	.000	.000	.000	.000

Test Statistics[a]

	Value_a	Value_b	Value_c	Value_d	Value_e
Mann-Whitney U	2007309.000	2048940.000	1965097.000	1999480.500	2007124.500
Wilcoxon W	7867485.000	7909116.000	7825273.000	7859656.500	7867300.500
Z	-47.108	-46.598	-47.623	-47.203	-47.110
Asymp. Sig. (2-tailed)	.000	.000	.000	.000	.000

Test Statistics[a]

	Value_f	Value_total_char	Value_entropy	Value_R_0
Mann-Whitney U	1948490.500	1856358.000	2120872.000	5584044.500
Wilcoxon W	7808666.500	7716534.000	7981048.000	11444220.50
Z	-47.827	-51.256	-45.712	-3.357
Asymp. Sig. (2-tailed)	.000	.000	.000	<.001

Test Statistics[a]

	Value_R_1	Value_R_2	Value_R_3	Value_R_4
Mann-Whitney U	5784804.500	5601027.000	5695851.500	5538340.000
Wilcoxon W	11644980.50	11461203.00	11556027.50	11398516.00
Z	-.901	-3.149	-1.989	-3.916
Asymp. Sig. (2-tailed)	.368	.002	.047	<.001

Test Statistics[a]

	Value_R_5	Value_R_6	Value_R_7	Value_R_8
Mann-Whitney U	5742435.500	5854736.500	5831433.500	5800871.000
Wilcoxon W	11602611.50	11714912.50	11691609.50	11661047.00
Z	-1.419	-.046	-.331	-.704
Asymp. Sig. (2-tailed)	.156	.964	.741	.481

Test Statistics[a]

	Value_R_9	Value_R_a	Value_R_b	Value_R_c
Mann-Whitney U	5834574.000	5706953.000	5759203.000	5535163.000
Wilcoxon W	11694750.00	11567129.00	11619379.00	11395339.00
Z	-.292	-1.853	-1.214	-3.954
Asymp. Sig. (2-tailed)	.770	.064	.225	<.001

Test Statistics[a]

	Value_R_d	Value_R_e	Value_R_f
Mann-Whitney U	5749996.500	5698607.000	5311170.500
Wilcoxon W	11610172.50	11558783.00	11171346.50
Z	-1.327	-1.955	-6.694
Asymp. Sig. (2-tailed)	.185	.051	<.001

a. Grouping Variable: Class

Figure B.47: Mann Whitney Test of Chat

Test Statistics[a]

	Value_0	Value_1	Value_2	Value_3
Mann-Whitney U	7779217.000	5971326.500	5964800.000	5945692.000
Wilcoxon W	28738792.00	26930901.50	26924375.00	26905267.00
Z	-61.982	-70.485	-70.517	-70.606
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_4	Value_5	Value_6	Value_7
Mann-Whitney U	5995262.000	5995052.500	5993109.000	5958781.500
Wilcoxon W	26954837.00	26954627.50	26952684.00	26918356.50
Z	-70.372	-70.374	-70.383	-70.544
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_8	Value_9	Value_a	Value_b
Mann-Whitney U	5992357.000	5960102.500	5901327.500	5969804.500
Wilcoxon W	26951932.00	26919677.50	26860902.50	26929379.50
Z	-70.387	-70.539	-70.815	-70.494
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_c	Value_d	Value_e	Value_f
Mann-Whitney U	5968894.500	5968703.000	5984782.500	6007628.500
Wilcoxon W	26928469.50	26928278.00	26944357.50	26967203.50
Z	-70.498	-70.499	-70.422	-70.315
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_total_char	Value_entropy	Value_R_0	Value_R_1
Mann-Whitney U	6045505.000	5511182.500	1764788.000	16562562.50
Wilcoxon W	27005080.00	26470757.50	22724363.00	37522137.50
Z	-71.525	-72.626	-90.245	-20.661
Asymp. Sig. (2-tailed)	.000	.000	.000	<.001

Test Statistics[a]

	Value_R_2	Value_R_3	Value_R_4	Value_R_5
Mann-Whitney U	15726227.50	14700986.00	16522999.00	15299504.50
Wilcoxon W	36685802.50	35660561.00	37482574.00	36259079.50
Z	-24.594	-29.415	-20.847	-26.600
Asymp. Sig. (2-tailed)	<.001	<.001	<.001	<.001

Test Statistics[a]

	Value_R_6	Value_R_7	Value_R_8	Value_R_9
Mann-Whitney U	15873784.00	15276411.00	16145457.50	15012277.00
Wilcoxon W	36833359.00	36235986.00	37105032.50	35971852.00
Z	-23.900	-26.709	-22.622	-27.951
Asymp. Sig. (2-tailed)	<.001	<.001	<.001	<.001

Test Statistics[a]

	Value_R_a	Value_R_b	Value_R_c	Value_R_d
Mann-Whitney U	14444516.50	14583028.50	14982597.00	15350897.00
Wilcoxon W	35404091.50	35542603.50	35942172.00	36310472.00
Z	-30.621	-29.969	-28.091	-26.359
Asymp. Sig. (2-tailed)	<.001	<.001	<.001	<.001

Test Statistics[a]

	Value_R_e	Value_R_f
Mann-Whitney U	15253548.00	15894453.00
Wilcoxon W	36213123.00	36854028.00
Z	-26.817	-23.803
Asymp. Sig. (2-tailed)	<.001	<.001

a. Grouping Variable: Class

Figure B.48: Mann Whitney Test of Email

Test Statistics[a]

	Value_0	Value_1	Value_2	Value_3
Mann-Whitney U	7589090559	7661347807	7609569471	7668182819
Wilcoxon W	4.432E+10	4.439E+10	4.434E+10	4.440E+10
Z	-505.875	-504.621	-505.520	-504.502
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_4	Value_5	Value_6	Value_7
Mann-Whitney U	7704405624	7592768798	7722250065	7665431754
Wilcoxon W	4.443E+10	4.432E+10	4.445E+10	4.439E+10
Z	-503.873	-505.812	-503.564	-504.550
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_8	Value_9	Value_a	Value_b
Mann-Whitney U	7605525068	7651846035	7685381135	7683480039
Wilcoxon W	4.433E+10	4.438E+10	4.441E+10	4.441E+10
Z	-505.590	-504.786	-504.204	-504.237
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_c	Value_d	Value_e	Value_f
Mann-Whitney U	7633121059	7627951504	7672679717	7662153271
Wilcoxon W	4.436E+10	4.436E+10	4.440E+10	4.439E+10
Z	-505.111	-505.201	-504.424	-504.607
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_total_char	Value_entropy	Value_R_0	Value_R_1
Mann-Whitney U	7516829460	1.065E+10	3.401E+10	3.640E+10
Wilcoxon W	4.424E+10	4.737E+10	7.074E+10	7.313E+10
Z	-519.375	-452.802	-47.199	-5.695
Asymp. Sig. (2-tailed)	.000	.000	.000	<.001

Test Statistics[a]

	Value_R_2	Value_R_3	Value_R_4	Value_R_5
Mann-Whitney U	3.482E+10	3.557E+10	3.443E+10	3.405E+10
Wilcoxon W	7.154E+10	7.230E+10	7.116E+10	7.077E+10
Z	-33.204	-20.136	-39.894	-46.575
Asymp. Sig. (2-tailed)	<.001	<.001	.000	.000

Test Statistics[a]

	Value_R_6	Value_R_7	Value_R_8	Value_R_9
Mann-Whitney U	3.372E+10	3.610E+10	3.399E+10	3.657E+10
Wilcoxon W	7.045E+10	7.283E+10	7.072E+10	7.330E+10
Z	-52.210	-10.917	-47.461	-2.663
Asymp. Sig. (2-tailed)	.000	<.001	.000	.008

Test Statistics[a]

	Value_R_a	Value_R_b	Value_R_c	Value_R_d
Mann-Whitney U	3.568E+10	3.588E+10	3.551E+10	3.504E+10
Wilcoxon W	7.241E+10	7.261E+10	7.223E+10	7.177E+10
Z	-18.180	-14.633	-21.209	-29.254
Asymp. Sig. (2-tailed)	<.001	<.001	<.001	<.001

Test Statistics[a]

	Value_R_e	Value_R_f
Mann-Whitney U	3.612E+10	3.620E+10
Wilcoxon W	7.285E+10	7.293E+10
Z	-10.578	-9.166
Asymp. Sig. (2-tailed)	<.001	<.001

a. Grouping Variable: Class

Figure B.49: Mann Whitney Test of FTP

Test Statistics[a]

	Value_0	Value_1	Value_2	Value_3
Mann-Whitney U	9284368613	9383681924	9436324587	9243128945
Wilcoxon W	3.534E+10	3.544E+10	3.550E+10	3.530E+10
Z	-376.744	-374.514	-373.332	-377.671
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_4	Value_5	Value_6	Value_7
Mann-Whitney U	9348229312	9285498393	9253931741	9228348740
Wilcoxon W	3.541E+10	3.535E+10	3.531E+10	3.529E+10
Z	-375.311	-376.720	-377.429	-378.003
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_8	Value_9	Value_a	Value_b
Mann-Whitney U	9426350316	9263209564	9433690676	9395447121
Wilcoxon W	3.549E+10	3.532E+10	3.549E+10	3.546E+10
Z	-373.555	-377.220	-373.392	-374.251
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_c	Value_d	Value_e	Value_f
Mann-Whitney U	9225458633	9304887284	9260888115	9310920858
Wilcoxon W	3.529E+10	3.537E+10	3.532E+10	3.537E+10
Z	-378.068	-376.284	-377.272	-376.148
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_total_char	Value_entropy	Value_R_0	Value_R_1
Mann-Whitney U	9282462678	1.495E+10	2.602E+10	2.406E+10
Wilcoxon W	3.534E+10	4.102E+10	5.208E+10	5.012E+10
Z	-402.985	-249.384	-.878	-45.018
Asymp. Sig. (2-tailed)	.000	.000	.380	.000

Test Statistics[a]

	Value_R_2	Value_R_3	Value_R_4	Value_R_5
Mann-Whitney U	2.360E+10	2.578E+10	2.503E+10	2.575E+10
Wilcoxon W	4.966E+10	5.184E+10	5.109E+10	5.181E+10
Z	-55.246	-6.320	-23.054	-6.950
Asymp. Sig. (2-tailed)	.000	<.001	<.001	<.001

Test Statistics[a]

	Value_R_6	Value_R_7	Value_R_8	Value_R_9
Mann-Whitney U	2.557E+10	2.534E+10	2.389E+10	2.580E+10
Wilcoxon W	5.163E+10	5.140E+10	4.995E+10	5.186E+10
Z	-10.931	-16.245	-48.688	-5.846
Asymp. Sig. (2-tailed)	<.001	<.001	.000	<.001

Test Statistics[a]

	Value_R_a	Value_R_b	Value_R_c	Value_R_d
Mann-Whitney U	2.460E+10	2.461E+10	2.519E+10	2.577E+10
Wilcoxon W	5.066E+10	5.067E+10	5.126E+10	5.183E+10
Z	-32.735	-32.477	-19.436	-6.580
Asymp. Sig. (2-tailed)	<.001	<.001	<.001	<.001

Test Statistics[a]

	Value_R_e	Value_R_f
Mann-Whitney U	2.546E+10	2.588E+10
Wilcoxon W	5.152E+10	5.194E+10
Z	-13.495	-4.073
Asymp. Sig. (2-tailed)	<.001	<.001

a. Grouping Variable: Class

Figure B.50: Mann Whitney Test of P2P

Test Statistics[a]

	Value_0	Value_1	Value_2	Value_3
Mann-Whitney U	73776491.50	70431647.00	70852394.50	70256096.50
Wilcoxon W	216978917.5	213634073.0	214054820.5	213458522.5
Z	-77.246	-80.972	-80.504	-81.167
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_4	Value_5	Value_6	Value_7
Mann-Whitney U	70419508.00	70637160.50	70585837.00	70372269.50
Wilcoxon W	213621934.0	213839586.5	213788263.0	213574695.5
Z	-80.985	-80.743	-80.800	-81.038
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_8	Value_9	Value_a	Value_b
Mann-Whitney U	70642994.00	70057452.50	70751530.50	70763958.50
Wilcoxon W	213845420.0	213259878.5	213953956.5	213966384.5
Z	-80.737	-81.388	-80.616	-80.602
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_c	Value_d	Value_e	Value_f
Mann-Whitney U	70895341.50	70472191.00	69935553.00	70155981.00
Wilcoxon W	214097767.5	213674617.0	213137979.0	213358407.0
Z	-80.456	-80.927	-81.524	-81.279
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_total_char	Value_entropy	Value_R_0	Value_R_1
Mann-Whitney U	68847866.00	69380952.00	105306685.5	132582149.0
Wilcoxon W	212050292.0	212583378.0	248509111.5	275784575.0
Z	-87.221	-82.127	-42.159	-11.808
Asymp. Sig. (2-tailed)	.000	.000	.000	<.001

Test Statistics[a]

	Value_R_2	Value_R_3	Value_R_4	Value_R_5
Mann-Whitney U	134635858.5	130357475.5	131400661.5	129117554.5
Wilcoxon W	277838284.5	273559901.5	274603087.5	272319980.5
Z	-9.523	-14.284	-13.123	-15.664
Asymp. Sig. (2-tailed)	<.001	<.001	<.001	<.001

Test Statistics[a]

	Value_R_6	Value_R_7	Value_R_8	Value_R_9
Mann-Whitney U	132380372.5	128509261.0	135435543.5	123974254.0
Wilcoxon W	275582798.5	271711687.0	278637969.5	267176680.0
Z	-12.033	-16.341	-8.633	-21.387
Asymp. Sig. (2-tailed)	<.001	<.001	<.001	<.001

Test Statistics[a]

	Value_R_a	Value_R_b	Value_R_c	Value_R_d
Mann-Whitney U	132302688.0	133486617.5	134129097.5	131359685.0
Wilcoxon W	275505114.0	276689043.5	277331523.5	274562111.0
Z	-12.119	-10.802	-10.087	-13.169
Asymp. Sig. (2-tailed)	<.001	<.001	<.001	<.001

Test Statistics[a]

	Value_R_e	Value_R_f
Mann-Whitney U	123228083.0	124832253.0
Wilcoxon W	266430509.0	268034679.0
Z	-22.217	-20.432
Asymp. Sig. (2-tailed)	<.001	<.001

a. Grouping Variable: Class

Figure B.51: Mann Whitney Test of VDO

Test Statistics[a]

	Value_0	Value_1	Value_2	Value_3
Mann-Whitney U	7084306070	2355561093	2136721553	2043469417
Wilcoxon W	8.239E+10	7.767E+10	7.745E+10	7.735E+10
Z	-691.306	-739.261	-741.486	-742.428
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_4	Value_5	Value_6	Value_7
Mann-Whitney U	2386989372	1778840564	2190652905	2102816942
Wilcoxon W	7.770E+10	7.709E+10	7.750E+10	7.741E+10
Z	-738.944	-745.116	-740.935	-741.822
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_8	Value_9	Value_a	Value_b
Mann-Whitney U	1603865394	2170904887	1814453032	1945069820
Wilcoxon W	7.691E+10	7.748E+10	7.712E+10	7.725E+10
Z	-746.883	-741.136	-744.751	-743.427
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_c	Value_d	Value_e	Value_f
Mann-Whitney U	1557795988	1814667605	2119194948	1510345591
Wilcoxon W	7.687E+10	7.712E+10	7.743E+10	7.682E+10
Z	-747.356	-744.749	-741.665	-747.840
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_total_char	Value_entropy	Value_R_0	Value_R_1
Mann-Whitney U	1952719160	1158451644	4.413E+10	7.255E+10
Wilcoxon W	7.726E+10	7.647E+10	1.194E+11	1.479E+11
Z	-791.297	-751.246	-315.902	-27.987
Asymp. Sig. (2-tailed)	.000	.000	.000	<.001

Test Statistics[a]

	Value_R_2	Value_R_3	Value_R_4	Value_R_5
Mann-Whitney U	7.294E+10	6.857E+10	7.006E+10	5.960E+10
Wilcoxon W	1.483E+11	1.439E+11	1.454E+11	1.349E+11
Z	-23.986	-68.308	-53.162	-159.168
Asymp. Sig. (2-tailed)	<.001	.000	.000	.000

Test Statistics[a]

	Value_R_6	Value_R_7	Value_R_8	Value_R_9
Mann-Whitney U	6.872E+10	6.125E+10	5.966E+10	7.340E+10
Wilcoxon W	1.440E+11	1.366E+11	1.350E+11	1.487E+11
Z	-66.795	-142.459	-158.550	-19.352
Asymp. Sig. (2-tailed)	.000	.000	.000	<.001

Test Statistics[a]

	Value_R_a	Value_R_b	Value_R_c	Value_R_d
Mann-Whitney U	5.978E+10	6.020E+10	5.735E+10	5.806E+10
Wilcoxon W	1.351E+11	1.355E+11	1.327E+11	1.334E+11
Z	-157.331	-153.086	-181.923	-174.766
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_R_e	Value_R_f
Mann-Whitney U	6.656E+10	5.469E+10
Wilcoxon W	1.419E+11	1.300E+11
Z	-88.697	-208.886
Asymp. Sig. (2-tailed)	.000	.000

a. Grouping Variable: Class

Figure B.52: Mann Whitney Test of VoIP

Test Statistics[a]

	Value_0	Value_1	Value_2	Value_3
Mann-Whitney U	54070038.00	52351437.00	52349452.00	52373042.00
Wilcoxon W	175430448.0	173711847.0	173709862.0	173733452.0
Z	-84.766	-86.931	-86.934	-86.904
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_4	Value_5	Value_6	Value_7
Mann-Whitney U	52307030.00	52434704.50	52217737.50	52237412.00
Wilcoxon W	173667440.0	173795114.5	173578147.5	173597822.0
Z	-86.987	-86.826	-87.100	-87.075
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_8	Value_9	Value_a	Value_b
Mann-Whitney U	52441357.00	52330015.50	52212157.50	52519801.00
Wilcoxon W	173801767.0	173690425.5	173572567.5	173880211.0
Z	-86.818	-86.959	-87.107	-86.719
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_c	Value_d	Value_e	Value_f
Mann-Whitney U	52487068.00	52269229.00	52230212.50	52270465.00
Wilcoxon W	173847478.0	173629639.0	173590622.5	173630875.0
Z	-86.760	-87.035	-87.084	-87.033
Asymp. Sig. (2-tailed)	.000	.000	.000	.000

Test Statistics[a]

	Value_total_char	Value_entropy	Value_R_0	Value_R_1
Mann-Whitney U	52872578.50	52313209.00	113072855.5	115920630.5
Wilcoxon W	174232988.5	173673619.0	234433265.5	237281040.5
Z	-89.965	-86.967	-10.431	-6.843
Asymp. Sig. (2-tailed)	.000	.000	<.001	<.001

Test Statistics[a]

	Value_R_2	Value_R_3	Value_R_4	Value_R_5
Mann-Whitney U	117245103.0	116364798.0	114215210.5	114858154.5
Wilcoxon W	238605513.0	237725208.0	235575620.5	236218564.5
Z	-5.174	-6.283	-8.991	-8.181
Asymp. Sig. (2-tailed)	<.001	<.001	<.001	<.001

Test Statistics[a]

	Value_R_6	Value_R_7	Value_R_8	Value_R_9
Mann-Whitney U	115221308.5	115051140.0	114876364.0	116718718.5
Wilcoxon W	236581718.5	236411550.0	236236774.0	238079128.5
Z	-7.724	-7.938	-8.159	-5.838
Asymp. Sig. (2-tailed)	<.001	<.001	<.001	<.001

Test Statistics[a]

	Value_R_a	Value_R_b	Value_R_c	Value_R_d
Mann-Whitney U	115110087.5	117549761.5	116401807.0	115225704.0
Wilcoxon W	236470497.5	238910171.5	237762217.0	236586114.0
Z	-7.864	-4.791	-6.237	-7.718
Asymp. Sig. (2-tailed)	<.001	<.001	<.001	<.001

Test Statistics[a]

	Value_R_e	Value_R_f
Mann-Whitney U	115663929.5	116254525.5
Wilcoxon W	237024339.5	237614935.5
Z	-7.166	-6.422
Asymp. Sig. (2-tailed)	<.001	<.001

a. Grouping Variable: Class

Figure B.53: Mann Whitney Test of Private Browsing

B.16 Pearson Correlations

B.16.1 Set 1 Feature

Correlations

		Value_total_char
Value_total_char	Pearson Correlation	1
	Sig. (2-tailed)	
	N	27454
Value_0	Pearson Correlation	.980**
	Sig. (2-tailed)	.000
	N	27454
Value_1	Pearson Correlation	.982**
	Sig. (2-tailed)	.000
	N	27454
Value_2	Pearson Correlation	.981**
	Sig. (2-tailed)	.000
	N	27454
Value_3	Pearson Correlation	.973**
	Sig. (2-tailed)	.000
	N	27454
Value_4	Pearson Correlation	.983**
	Sig. (2-tailed)	.000
	N	27454
Value_5	Pearson Correlation	.982**
	Sig. (2-tailed)	.000
	N	27454
Value_6	Pearson Correlation	.969**
	Sig. (2-tailed)	.000
	N	27454
Value_7	Pearson Correlation	.982**
	Sig. (2-tailed)	.000
	N	27454
Value_8	Pearson Correlation	.981**
	Sig. (2-tailed)	.000
	N	27454
Value_9	Pearson Correlation	.982**
	Sig. (2-tailed)	.000
	N	27454
Value_a	Pearson Correlation	.981**
	Sig. (2-tailed)	.000
	N	27454
Value_b	Pearson Correlation	.980**
	Sig. (2-tailed)	.000
	N	27454
Value_c	Pearson Correlation	.981**
	Sig. (2-tailed)	.000
	N	27454
Value_d	Pearson Correlation	.982**
	Sig. (2-tailed)	.000
	N	27454
Value_e	Pearson Correlation	.981**
	Sig. (2-tailed)	.000
	N	27454
Value_f	Pearson Correlation	.980**
	Sig. (2-tailed)	.000
	N	27454

**. Correlation is significant at the 0.01 level (2-tailed).

Correlations

		Value_total_char
Value_total_char	Pearson Correlation	1
	Sig. (2-tailed)	
	N	75736
Value_0	Pearson Correlation	.977**
	Sig. (2-tailed)	.000
	N	75736
Value_1	Pearson Correlation	.986**
	Sig. (2-tailed)	.000
	N	75736
Value_2	Pearson Correlation	.986**
	Sig. (2-tailed)	.000
	N	75736
Value_3	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	75736
Value_4	Pearson Correlation	.986**
	Sig. (2-tailed)	.000
	N	75736
Value_5	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	75736
Value_6	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	75736
Value_7	Pearson Correlation	.986**
	Sig. (2-tailed)	.000
	N	75736
Value_8	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	75736
Value_9	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	75736
Value_a	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	75736
Value_b	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	75736
Value_c	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	75736
Value_d	Pearson Correlation	.986**
	Sig. (2-tailed)	.000
	N	75736
Value_e	Pearson Correlation	.986**
	Sig. (2-tailed)	.000
	N	75736
Value_f	Pearson Correlation	.986**
	Sig. (2-tailed)	.000
	N	75736

**. Correlation is significant at the 0.01 level (2-tailed).

Correlations

		Value_total_char
Value_total_char	Pearson Correlation	1
	Sig. (2-tailed)	
	N	6846
Value_0	Pearson Correlation	.977**
	Sig. (2-tailed)	.000
	N	6846
Value_1	Pearson Correlation	.984**
	Sig. (2-tailed)	.000
	N	6846
Value_2	Pearson Correlation	.984**
	Sig. (2-tailed)	.000
	N	6846
Value_3	Pearson Correlation	.983**
	Sig. (2-tailed)	.000
	N	6846
Value_4	Pearson Correlation	.984**
	Sig. (2-tailed)	.000
	N	6846
Value_5	Pearson Correlation	.984**
	Sig. (2-tailed)	.000
	N	6846
Value_6	Pearson Correlation	.983**
	Sig. (2-tailed)	.000
	N	6846
Value_7	Pearson Correlation	.984**
	Sig. (2-tailed)	.000
	N	6846
Value_8	Pearson Correlation	.984**
	Sig. (2-tailed)	.000
	N	6846
Value_9	Pearson Correlation	.983**
	Sig. (2-tailed)	.000
	N	6846
Value_a	Pearson Correlation	.983**
	Sig. (2-tailed)	.000
	N	6846
Value_b	Pearson Correlation	.983**
	Sig. (2-tailed)	.000
	N	6846
Value_c	Pearson Correlation	.983**
	Sig. (2-tailed)	.000
	N	6846
Value_d	Pearson Correlation	.984**
	Sig. (2-tailed)	.000
	N	6846
Value_e	Pearson Correlation	.984**
	Sig. (2-tailed)	.000
	N	6846
Value_f	Pearson Correlation	.984**
	Sig. (2-tailed)	.000
	N	6846

**. Correlation is significant at the 0.01 level (2-tailed).

Figure B.54: Pearson Correlations of Set 1 Feature of Audio, Browsing and Chat

Correlations

		Value_total_char
Value_total_char	Pearson Correlation	1
	Sig. (2-tailed)	
	N	12948
Value_0	Pearson Correlation	.979**
	Sig. (2-tailed)	.000
	N	12948
Value_1	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	12948
Value_2	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	12948
Value_3	Pearson Correlation	.989**
	Sig. (2-tailed)	.000
	N	12948
Value_4	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	12948
Value_5	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	12948
Value_6	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	12948
Value_7	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	12948
Value_8	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	12948
Value_9	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	12948
Value_a	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	12948
Value_b	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	12948
Value_c	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	12948
Value_d	Pearson Correlation	.989**
	Sig. (2-tailed)	.000
	N	12948
Value_e	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	12948
Value_f	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	12948

**. Correlation is significant at the 0.01 level (2-tailed).

Correlations

		Value_total_char
Value_total_char	Pearson Correlation	1
	Sig. (2-tailed)	
	N	542054
Value_0	Pearson Correlation	.988**
	Sig. (2-tailed)	.000
	N	542054
Value_1	Pearson Correlation	.988**
	Sig. (2-tailed)	.000
	N	542054
Value_2	Pearson Correlation	.988**
	Sig. (2-tailed)	.000
	N	542054
Value_3	Pearson Correlation	.986**
	Sig. (2-tailed)	.000
	N	542054
Value_4	Pearson Correlation	.988**
	Sig. (2-tailed)	.000
	N	542054
Value_5	Pearson Correlation	.988**
	Sig. (2-tailed)	.000
	N	542054
Value_6	Pearson Correlation	.986**
	Sig. (2-tailed)	.000
	N	542054
Value_7	Pearson Correlation	.988**
	Sig. (2-tailed)	.000
	N	542054
Value_8	Pearson Correlation	.988**
	Sig. (2-tailed)	.000
	N	542054
Value_9	Pearson Correlation	.988**
	Sig. (2-tailed)	.000
	N	542054
Value_a	Pearson Correlation	.988**
	Sig. (2-tailed)	.000
	N	542054
Value_b	Pearson Correlation	.988**
	Sig. (2-tailed)	.000
	N	542054
Value_c	Pearson Correlation	.988**
	Sig. (2-tailed)	.000
	N	542054
Value_d	Pearson Correlation	.988**
	Sig. (2-tailed)	.000
	N	542054
Value_e	Pearson Correlation	.988**
	Sig. (2-tailed)	.000
	N	542054
Value_f	Pearson Correlation	.988**
	Sig. (2-tailed)	.000
	N	542054

**. Correlation is significant at the 0.01 level (2-tailed).

Correlations

		Value_total_char
Value_total_char	Pearson Correlation	1
	Sig. (2-tailed)	
	N	456600
Value_0	Pearson Correlation	.718**
	Sig. (2-tailed)	.000
	N	456600
Value_1	Pearson Correlation	.979**
	Sig. (2-tailed)	.000
	N	456600
Value_2	Pearson Correlation	.977**
	Sig. (2-tailed)	.000
	N	456600
Value_3	Pearson Correlation	.979**
	Sig. (2-tailed)	.000
	N	456600
Value_4	Pearson Correlation	.976**
	Sig. (2-tailed)	.000
	N	456600
Value_5	Pearson Correlation	.979**
	Sig. (2-tailed)	.000
	N	456600
Value_6	Pearson Correlation	.971**
	Sig. (2-tailed)	.000
	N	456600
Value_7	Pearson Correlation	.978**
	Sig. (2-tailed)	.000
	N	456600
Value_8	Pearson Correlation	.976**
	Sig. (2-tailed)	.000
	N	456600
Value_9	Pearson Correlation	.979**
	Sig. (2-tailed)	.000
	N	456600
Value_a	Pearson Correlation	.976**
	Sig. (2-tailed)	.000
	N	456600
Value_b	Pearson Correlation	.979**
	Sig. (2-tailed)	.000
	N	456600
Value_c	Pearson Correlation	.981**
	Sig. (2-tailed)	.000
	N	456600
Value_d	Pearson Correlation	.979**
	Sig. (2-tailed)	.000
	N	456600
Value_e	Pearson Correlation	.978**
	Sig. (2-tailed)	.000
	N	456600
Value_f	Pearson Correlation	.935**
	Sig. (2-tailed)	.000
	N	456600

**. Correlation is significant at the 0.01 level (2-tailed).

Figure B.55: Pearson Correlations of Set 1 Feature of Email, FTP and P2P

Correlations

		Value_total_char
Value_total_char	Pearson Correlation	1
	Sig. (2-tailed)	
	N	33846
Value_0	Pearson Correlation	.982**
	Sig. (2-tailed)	.000
	N	33846
Value_1	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	33846
Value_2	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	33846
Value_3	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	33846
Value_4	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	33846
Value_5	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	33846
Value_6	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	33846
Value_7	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	33846
Value_8	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	33846
Value_9	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	33846
Value_a	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	33846
Value_b	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	33846
Value_c	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	33846
Value_d	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	33846
Value_e	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	33846
Value_f	Pearson Correlation	.987**
	Sig. (2-tailed)	.000
	N	33846

**. Correlation is significant at the 0.01 level (2-tailed).

Correlations

		Value_total_char
Value_total_char	Pearson Correlation	1
	Sig. (2-tailed)	
	N	776192
Value_0	Pearson Correlation	.884**
	Sig. (2-tailed)	.000
	N	776192
Value_1	Pearson Correlation	.977**
	Sig. (2-tailed)	.000
	N	776192
Value_2	Pearson Correlation	.977**
	Sig. (2-tailed)	.000
	N	776192
Value_3	Pearson Correlation	.978**
	Sig. (2-tailed)	.000
	N	776192
Value_4	Pearson Correlation	.976**
	Sig. (2-tailed)	.000
	N	776192
Value_5	Pearson Correlation	.977**
	Sig. (2-tailed)	.000
	N	776192
Value_6	Pearson Correlation	.977**
	Sig. (2-tailed)	.000
	N	776192
Value_7	Pearson Correlation	.978**
	Sig. (2-tailed)	.000
	N	776192
Value_8	Pearson Correlation	.977**
	Sig. (2-tailed)	.000
	N	776192
Value_9	Pearson Correlation	.976**
	Sig. (2-tailed)	.000
	N	776192
Value_a	Pearson Correlation	.977**
	Sig. (2-tailed)	.000
	N	776192
Value_b	Pearson Correlation	.977**
	Sig. (2-tailed)	.000
	N	776192
Value_c	Pearson Correlation	.975**
	Sig. (2-tailed)	.000
	N	776192
Value_d	Pearson Correlation	.976**
	Sig. (2-tailed)	.000
	N	776192
Value_e	Pearson Correlation	.977**
	Sig. (2-tailed)	.000
	N	776192
Value_f	Pearson Correlation	.975**
	Sig. (2-tailed)	.000
	N	776192

**. Correlation is significant at the 0.01 level (2-tailed).

Correlations

		Value_total_char
Value_total_char	Pearson Correlation	1
	Sig. (2-tailed)	
	N	31158
Value_0	Pearson Correlation	.988**
	Sig. (2-tailed)	.000
	N	31158
Value_1	Pearson Correlation	.989**
	Sig. (2-tailed)	.000
	N	31158
Value_2	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	31158
Value_3	Pearson Correlation	.989**
	Sig. (2-tailed)	.000
	N	31158
Value_4	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	31158
Value_5	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	31158
Value_6	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	31158
Value_7	Pearson Correlation	.989**
	Sig. (2-tailed)	.000
	N	31158
Value_8	Pearson Correlation	.989**
	Sig. (2-tailed)	.000
	N	31158
Value_9	Pearson Correlation	.989**
	Sig. (2-tailed)	.000
	N	31158
Value_a	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	31158
Value_b	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	31158
Value_c	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	31158
Value_d	Pearson Correlation	.990**
	Sig. (2-tailed)	.000
	N	31158
Value_e	Pearson Correlation	.989**
	Sig. (2-tailed)	.000
	N	31158
Value_f	Pearson Correlation	.989**
	Sig. (2-tailed)	.000
	N	31158

**. Correlation is significant at the 0.01 level (2-tailed).

Figure B.56: Pearson Correlations of Set 1 Feature of VDO, VoIP and Private Browsing

B.16.2 Set2 Feature

Correlations

		Value_total_char
Value_total_char	Pearson Correlation	1
	Sig. (2-tailed)	
	N	27454
Value_R_0	Pearson Correlation	-.012
	Sig. (2-tailed)	.050
	N	27454
Value_R_1	Pearson Correlation	-.043**
	Sig. (2-tailed)	<.001
	N	27454
Value_R_2	Pearson Correlation	.026**
	Sig. (2-tailed)	<.001
	N	27454
Value_R_3	Pearson Correlation	.018**
	Sig. (2-tailed)	.002
	N	27454
Value_R_4	Pearson Correlation	.031**
	Sig. (2-tailed)	<.001
	N	27454
Value_R_5	Pearson Correlation	-.007
	Sig. (2-tailed)	.273
	N	27454
Value_R_6	Pearson Correlation	-.015*
	Sig. (2-tailed)	.013
	N	27454
Value_R_7	Pearson Correlation	.010
	Sig. (2-tailed)	.110
	N	27454
Value_R_8	Pearson Correlation	.028**
	Sig. (2-tailed)	<.001
	N	27454
Value_R_9	Pearson Correlation	.033**
	Sig. (2-tailed)	<.001
	N	27454
Value_R_a	Pearson Correlation	.040**
	Sig. (2-tailed)	<.001
	N	27454
Value_R_b	Pearson Correlation	-.064**
	Sig. (2-tailed)	<.001
	N	27454
Value_R_c	Pearson Correlation	-.017**
	Sig. (2-tailed)	.006
	N	27454
Value_R_d	Pearson Correlation	.022**
	Sig. (2-tailed)	<.001
	N	27454
Value_R_e	Pearson Correlation	.004
	Sig. (2-tailed)	.561
	N	27454
Value_R_f	Pearson Correlation	-.045**
	Sig. (2-tailed)	<.001
	N	27454

**. Correlation is significant at the 0.01 level (2-tailed).

Correlations

		Value_total_char
Value_total_char	Pearson Correlation	1
	Sig. (2-tailed)	
	N	75736
Value_R_0	Pearson Correlation	-.579**
	Sig. (2-tailed)	.000
	N	75736
Value_R_1	Pearson Correlation	.148**
	Sig. (2-tailed)	.000
	N	75736
Value_R_2	Pearson Correlation	.162**
	Sig. (2-tailed)	.000
	N	75736
Value_R_3	Pearson Correlation	.173**
	Sig. (2-tailed)	.000
	N	75736
Value_R_4	Pearson Correlation	.176**
	Sig. (2-tailed)	.000
	N	75736
Value_R_5	Pearson Correlation	.178**
	Sig. (2-tailed)	.000
	N	75736
Value_R_6	Pearson Correlation	.177**
	Sig. (2-tailed)	.000
	N	75736
Value_R_7	Pearson Correlation	.188**
	Sig. (2-tailed)	.000
	N	75736
Value_R_8	Pearson Correlation	.181**
	Sig. (2-tailed)	.000
	N	75736
Value_R_9	Pearson Correlation	.178**
	Sig. (2-tailed)	.000
	N	75736
Value_R_a	Pearson Correlation	.183**
	Sig. (2-tailed)	.000
	N	75736
Value_R_b	Pearson Correlation	.193**
	Sig. (2-tailed)	.000
	N	75736
Value_R_c	Pearson Correlation	.192**
	Sig. (2-tailed)	.000
	N	75736
Value_R_d	Pearson Correlation	.185**
	Sig. (2-tailed)	.000
	N	75736
Value_R_e	Pearson Correlation	.193**
	Sig. (2-tailed)	.000
	N	75736
Value_R_f	Pearson Correlation	.188**
	Sig. (2-tailed)	.000
	N	75736

**. Correlation is significant at the 0.01 level (2-tailed).

Correlations

		Value_total_char
Value_total_char	Pearson Correlation	1
	Sig. (2-tailed)	
	N	6846
Value_R_0	Pearson Correlation	.020
	Sig. (2-tailed)	.100
	N	6846
Value_R_1	Pearson Correlation	-.009
	Sig. (2-tailed)	.453
	N	6846
Value_R_2	Pearson Correlation	.011
	Sig. (2-tailed)	.361
	N	6846
Value_R_3	Pearson Correlation	-.002
	Sig. (2-tailed)	.871
	N	6846
Value_R_4	Pearson Correlation	.022
	Sig. (2-tailed)	.063
	N	6846
Value_R_5	Pearson Correlation	.035**
	Sig. (2-tailed)	.004
	N	6846
Value_R_6	Pearson Correlation	-.019
	Sig. (2-tailed)	.108
	N	6846
Value_R_7	Pearson Correlation	.005
	Sig. (2-tailed)	.695
	N	6846
Value_R_8	Pearson Correlation	-.019
	Sig. (2-tailed)	.109
	N	6846
Value_R_9	Pearson Correlation	-.011
	Sig. (2-tailed)	.376
	N	6846
Value_R_a	Pearson Correlation	-.010
	Sig. (2-tailed)	.417
	N	6846
Value_R_b	Pearson Correlation	-.019
	Sig. (2-tailed)	.108
	N	6846
Value_R_c	Pearson Correlation	-.011
	Sig. (2-tailed)	.355
	N	6846
Value_R_d	Pearson Correlation	-.033**
	Sig. (2-tailed)	.006
	N	6846
Value_R_e	Pearson Correlation	.010
	Sig. (2-tailed)	.407
	N	6846
Value_R_f	Pearson Correlation	.037**
	Sig. (2-tailed)	.002
	N	6846

**. Correlation is significant at the 0.01 level (2-tailed).

Figure B.57: Pearson Correlations of Set 2 Feature of Audio, Browsing and Chat

Correlations

		Value_total_char
Value_total_char	Pearson Correlation	1
	Sig. (2-tailed)	
	N	12948
Value_R_0	Pearson Correlation	-.657**
	Sig. (2-tailed)	.000
	N	12948
Value_R_1	Pearson Correlation	.155**
	Sig. (2-tailed)	<.001
	N	12948
Value_R_2	Pearson Correlation	.194**
	Sig. (2-tailed)	<.001
	N	12948
Value_R_3	Pearson Correlation	.206**
	Sig. (2-tailed)	<.001
	N	12948
Value_R_4	Pearson Correlation	.181**
	Sig. (2-tailed)	<.001
	N	12948
Value_R_5	Pearson Correlation	.216**
	Sig. (2-tailed)	<.001
	N	12948
Value_R_6	Pearson Correlation	.198**
	Sig. (2-tailed)	<.001
	N	12948
Value_R_7	Pearson Correlation	.197**
	Sig. (2-tailed)	<.001
	N	12948
Value_R_8	Pearson Correlation	.197**
	Sig. (2-tailed)	<.001
	N	12948
Value_R_9	Pearson Correlation	.205**
	Sig. (2-tailed)	<.001
	N	12948
Value_R_a	Pearson Correlation	.214**
	Sig. (2-tailed)	<.001
	N	12948
Value_R_b	Pearson Correlation	.228**
	Sig. (2-tailed)	<.001
	N	12948
Value_R_c	Pearson Correlation	.226**
	Sig. (2-tailed)	<.001
	N	12948
Value_R_d	Pearson Correlation	.208**
	Sig. (2-tailed)	<.001
	N	12948
Value_R_e	Pearson Correlation	.213**
	Sig. (2-tailed)	<.001
	N	12948
Value_R_f	Pearson Correlation	.207**
	Sig. (2-tailed)	<.001
	N	12948

**. Correlation is significant at the 0.01 level (2-tailed).

Correlations

		Value_total_char
Value_total_char	Pearson Correlation	1
	Sig. (2-tailed)	
	N	542054
Value_R_0	Pearson Correlation	.053**
	Sig. (2-tailed)	.000
	N	542054
Value_R_1	Pearson Correlation	-.010**
	Sig. (2-tailed)	<.001
	N	542054
Value_R_2	Pearson Correlation	.051**
	Sig. (2-tailed)	.000
	N	542054
Value_R_3	Pearson Correlation	-.029**
	Sig. (2-tailed)	<.001
	N	542054
Value_R_4	Pearson Correlation	-.079**
	Sig. (2-tailed)	.000
	N	542054
Value_R_5	Pearson Correlation	.053**
	Sig. (2-tailed)	.000
	N	542054
Value_R_6	Pearson Correlation	-.084**
	Sig. (2-tailed)	.000
	N	542054
Value_R_7	Pearson Correlation	-.025**
	Sig. (2-tailed)	<.001
	N	542054
Value_R_8	Pearson Correlation	.057**
	Sig. (2-tailed)	.000
	N	542054
Value_R_9	Pearson Correlation	-.011**
	Sig. (2-tailed)	<.001
	N	542054
Value_R_a	Pearson Correlation	-.025**
	Sig. (2-tailed)	<.001
	N	542054
Value_R_b	Pearson Correlation	-.029**
	Sig. (2-tailed)	<.001
	N	542054
Value_R_c	Pearson Correlation	.037**
	Sig. (2-tailed)	<.001
	N	542054
Value_R_d	Pearson Correlation	.043**
	Sig. (2-tailed)	<.001
	N	542054
Value_R_e	Pearson Correlation	-.025**
	Sig. (2-tailed)	<.001
	N	542054
Value_R_f	Pearson Correlation	.021**
	Sig. (2-tailed)	<.001
	N	542054

**. Correlation is significant at the 0.01 level (2-tailed).

Correlations

		Value_total_char
Value_total_char	Pearson Correlation	1
	Sig. (2-tailed)	
	N	456600
Value_R_0	Pearson Correlation	-.133**
	Sig. (2-tailed)	.000
	N	456600
Value_R_1	Pearson Correlation	.022**
	Sig. (2-tailed)	<.001
	N	456600
Value_R_2	Pearson Correlation	.013**
	Sig. (2-tailed)	<.001
	N	456600
Value_R_3	Pearson Correlation	.064**
	Sig. (2-tailed)	.000
	N	456600
Value_R_4	Pearson Correlation	.006**
	Sig. (2-tailed)	<.001
	N	456600
Value_R_5	Pearson Correlation	.056**
	Sig. (2-tailed)	.000
	N	456600
Value_R_6	Pearson Correlation	.042**
	Sig. (2-tailed)	<.001
	N	456600
Value_R_7	Pearson Correlation	.079**
	Sig. (2-tailed)	.000
	N	456600
Value_R_8	Pearson Correlation	.025**
	Sig. (2-tailed)	<.001
	N	456600
Value_R_9	Pearson Correlation	.063**
	Sig. (2-tailed)	.000
	N	456600
Value_R_a	Pearson Correlation	.037**
	Sig. (2-tailed)	<.001
	N	456600
Value_R_b	Pearson Correlation	.050**
	Sig. (2-tailed)	<.001
	N	456600
Value_R_c	Pearson Correlation	.070**
	Sig. (2-tailed)	.000
	N	456600
Value_R_d	Pearson Correlation	.041**
	Sig. (2-tailed)	<.001
	N	456600
Value_R_e	Pearson Correlation	.082**
	Sig. (2-tailed)	.000
	N	456600
Value_R_f	Pearson Correlation	.064**
	Sig. (2-tailed)	.000
	N	456600

**. Correlation is significant at the 0.01 level (2-tailed).

Figure B.58: Pearson Correlations of Set 2 Feature of Email, FTP and P2P

Correlations

		Value_total_char
Value_total_char	Pearson Correlation	1
	Sig. (2-tailed)	
	N	33846
Value_R_0	Pearson Correlation	-.370**
	Sig. (2-tailed)	.000
	N	33846
Value_R_1	Pearson Correlation	.069**
	Sig. (2-tailed)	<.001
	N	33846
Value_R_2	Pearson Correlation	.074**
	Sig. (2-tailed)	<.001
	N	33846
Value_R_3	Pearson Correlation	.080**
	Sig. (2-tailed)	<.001
	N	33846
Value_R_4	Pearson Correlation	.083**
	Sig. (2-tailed)	<.001
	N	33846
Value_R_5	Pearson Correlation	.084**
	Sig. (2-tailed)	<.001
	N	33846
Value_R_6	Pearson Correlation	.079**
	Sig. (2-tailed)	<.001
	N	33846
Value_R_7	Pearson Correlation	.099**
	Sig. (2-tailed)	<.001
	N	33846
Value_R_8	Pearson Correlation	.080**
	Sig. (2-tailed)	<.001
	N	33846
Value_R_9	Pearson Correlation	.107**
	Sig. (2-tailed)	<.001
	N	33846
Value_R_a	Pearson Correlation	.090**
	Sig. (2-tailed)	<.001
	N	33846
Value_R_b	Pearson Correlation	.088**
	Sig. (2-tailed)	<.001
	N	33846
Value_R_c	Pearson Correlation	.095**
	Sig. (2-tailed)	<.001
	N	33846
Value_R_d	Pearson Correlation	.093**
	Sig. (2-tailed)	<.001
	N	33846
Value_R_e	Pearson Correlation	.087**
	Sig. (2-tailed)	<.001
	N	33846
Value_R_f	Pearson Correlation	.100**
	Sig. (2-tailed)	<.001
	N	33846

**. Correlation is significant at the 0.01 level (2-tailed).

Correlations

		Value_total_char
Value_total_char	Pearson Correlation	1
	Sig. (2-tailed)	
	N	776192
Value_R_0	Pearson Correlation	-.378**
	Sig. (2-tailed)	.000
	N	776192
Value_R_1	Pearson Correlation	-.042**
	Sig. (2-tailed)	<.001
	N	776192
Value_R_2	Pearson Correlation	-.053**
	Sig. (2-tailed)	.000
	N	776192
Value_R_3	Pearson Correlation	.055**
	Sig. (2-tailed)	.000
	N	776192
Value_R_4	Pearson Correlation	.003**
	Sig. (2-tailed)	.003
	N	776192
Value_R_5	Pearson Correlation	.126**
	Sig. (2-tailed)	.000
	N	776192
Value_R_6	Pearson Correlation	.047**
	Sig. (2-tailed)	.000
	N	776192
Value_R_7	Pearson Correlation	.146**
	Sig. (2-tailed)	.000
	N	776192
Value_R_8	Pearson Correlation	.125**
	Sig. (2-tailed)	.000
	N	776192
Value_R_9	Pearson Correlation	-.025**
	Sig. (2-tailed)	<.001
	N	776192
Value_R_a	Pearson Correlation	.146**
	Sig. (2-tailed)	.000
	N	776192
Value_R_b	Pearson Correlation	.144**
	Sig. (2-tailed)	.000
	N	776192
Value_R_c	Pearson Correlation	.152**
	Sig. (2-tailed)	.000
	N	776192
Value_R_d	Pearson Correlation	.162**
	Sig. (2-tailed)	.000
	N	776192
Value_R_e	Pearson Correlation	.078**
	Sig. (2-tailed)	.000
	N	776192
Value_R_f	Pearson Correlation	.177**
	Sig. (2-tailed)	.000
	N	776192

**. Correlation is significant at the 0.01 level (2-tailed).

Correlations

		Value_total_char
Value_total_char	Pearson Correlation	1
	Sig. (2-tailed)	
	N	31158
Value_R_0	Pearson Correlation	-.121**
	Sig. (2-tailed)	<.001
	N	31158
Value_R_1	Pearson Correlation	.018**
	Sig. (2-tailed)	.001
	N	31158
Value_R_2	Pearson Correlation	.004
	Sig. (2-tailed)	.460
	N	31158
Value_R_3	Pearson Correlation	.004
	Sig. (2-tailed)	.474
	N	31158
Value_R_4	Pearson Correlation	.022**
	Sig. (2-tailed)	<.001
	N	31158
Value_R_5	Pearson Correlation	.024**
	Sig. (2-tailed)	<.001
	N	31158
Value_R_6	Pearson Correlation	.023**
	Sig. (2-tailed)	<.001
	N	31158
Value_R_7	Pearson Correlation	.011*
	Sig. (2-tailed)	.047
	N	31158
Value_R_8	Pearson Correlation	.027**
	Sig. (2-tailed)	<.001
	N	31158
Value_R_9	Pearson Correlation	.004
	Sig. (2-tailed)	.515
	N	31158
Value_R_a	Pearson Correlation	.012*
	Sig. (2-tailed)	.031
	N	31158
Value_R_b	Pearson Correlation	-.001
	Sig. (2-tailed)	.865
	N	31158
Value_R_c	Pearson Correlation	.008
	Sig. (2-tailed)	.161
	N	31158
Value_R_d	Pearson Correlation	.010
	Sig. (2-tailed)	.082
	N	31158
Value_R_e	Pearson Correlation	.020**
	Sig. (2-tailed)	<.001
	N	31158
Value_R_f	Pearson Correlation	.011
	Sig. (2-tailed)	.058
	N	31158

**. Correlation is significant at the 0.01 level (2-tailed).

Figure B.59: Pearson Correlations of Set 2 Feature of VDO, VoIP and Private Browsing

Appendix C

Deep Learning

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.metrics import accuracy_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, Flatten, Dense

# Load and preprocess the data
data = pd.read_csv('writing/#new/browsing-b-seen.csv')
#X = MinMaxScaler().fit_transform(data.iloc[:, 0:15].values)  # Features F_0 - F_f
X = MinMaxScaler().fit_transform(data.iloc[:, 18:34].values)  # Features R_0 - R_f
y = LabelEncoder().fit_transform(data.iloc[:, -1].values)

# Splitting data
X_temp, X_test, y_temp, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_temp, y_temp, test_size=0.25, random_state=42)

# Reshape data for CNN
X_train, X_val, X_test = [x.reshape(*x.shape, 1) for x in [X_train, X_val, X_test]]

# Define and compile the 1D CNN model
model = Sequential([
    Conv1D(16, 3, activation='relu', input_shape=X_train.shape[1:]),
    Flatten(),
```

```
    Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam', loss='binary_crossentropy')

# Train the model
history = model.fit(X_train, y_train, epochs=10, batch_size=32,
↪  verbose=0, validation_data=(X_val, y_val))

# Plot loss curves
plt.figure(figsize=(12, 6))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss Curves')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

# Display training and validation loss
print("Training Loss per Epoch:", *map("{:.4f}".format,
↪  history.history['loss']), sep='\n')
print("\nValidation Loss per Epoch:", *map("{:.4f}".format,
↪  history.history['val_loss']), sep='\n')

# Make predictions
y_pred = model.predict(X_test)
y_pred = (y_pred > 0.5).astype(int).reshape(-1)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

```

Appendix D

A Thesis Data

All data for this thesis is available at http://personal.strath.ac.uk/pitpimon.choorod