TOR USER DE-ANONYMIZATION: CLIENT-SIDE ORIGINATING WATERMARK

BY
DANIEL BROWN

A THESIS SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE IN COMPUTER SCIENCE

ELECTRICAL ENGINEERING AND COMPUTER SCIENCE (EECS)
YORK UNIVERSITY
TORONTO, ONTARIO
FEBRUARY 2025

Abstract

Traditional techniques for Tor user de-anonymization through a side-channel by means of traffic-watermarks are generally implemented through utilization/modulation of server-side originating traffic (SSOW). However, the effectiveness of SSOW is often hindered by significant amounts of traffic noise that accumulates along Tor's communication pathways. In this thesis, we outline the key ideas behind our novel user de-anonymization technique that utilizes client-side originating watermarks (CSOW). We describe some potential ways this scheme could be implemented in practice while not requiring the control of any Tor node or other resource. We also demonstrate significantly superior real-world performance of our CSOW approach vs. those previously discussed in the literature. Finally, we propose the use of Long Short-Term Memory (LSTM) DNN for the purpose of more effective watermark detection. The real-world experimentations demonstrate excellent potential of our proposed LSTM-Based CSOW watermark detection system to accurately de-anonymize Tor users while keeping the number of false positives (e.g., users mistakenly accused of wrongdoing) at an absolute 0.

Publications that have come from this thesis:

- Brown, Daniel, and Natalija Vlajic. "Novel Approach to Tor User De-Anonymization: Client-Side Originating Watermark with LSTM Autoencoder Detection." *ACM Conference on Data and Application Security and Privacy*, **(IN SUBMISSION)**
- Brown, Daniel, and Natalija Vlajic. "Novel approach to Tor User De-Anonymization: Client-side generated watermark." *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2 Sept. 2024, pp. 90–95, https://doi.org/10.1109/csr61664.2024.10679508.
- Brown, Daniel, and Natalija Vlajic. "Poster: Novel client-side watermarking technique for tor user de-anonymization." *Proceedings of the 2023 ACM on Internet Measurement Conference*, 24 Oct. 2023, pp. 720–721, https://doi.org/10.1145/3618257.3624997.

Acknowledgements

Throughout the journey of my Master's Program, I have received an immense amount of support and assistance, I recognize how lucky I am to have this and I am deeply grateful for it.

I would like to express my sincere gratitude to my supervisor, Professor Natalija Vlajic, for her exceptional mentorship and support. Her insightful feedback, attention to detail, and constant encouragement have been invaluable in shaping my work but also personal growth. Professor Vlajic's dedication to excellence has inspired me to strive for higher standards, and her belief in my potential has made a lasting impact. It has been both an honour and a privilege to work under her guidance, and I will continue to forward the skills and knowledge I gained during this experience into my future endeavours.

I also wish to extend my appreciation to Professor Yan Shvartzshnaider, the second member of my supervisory committee, for his invaluable feedback during the work on this thesis, as well as my fellow colleagues at the Department of Electrical Engineering and Computer Science. I am particularly grateful to Professor Sana Maqsood and Professor Manar Jammal for their willingness to serve on my examination committee.

Summary of Main Contributions

In this thesis we cover many topics regarding Tor User De-Anonymization, to better understand how we contribute to this field we will be briefly outlining our main contributions. Firstly, we propose a methodology to overcome the limitations of existing Tor User De-Anonymization Techniques that utilize Server-Side Originating Watermarks (SSOW). Our methodology changes the direction of the watermark flow to originate from the client through the use of Post Requests, we call this our Client-Side Originating Watermark (CSOW) approach. The advantage of using our CSOW methodology is that the watermark is detected without having to pass through the Tor Network, which is key a limitation in standard SSOW methodology as the Tor Network induces lots of noise into the network traffic often altering the watermark beyond recognition. Our results sections go into further detail proving this argument and outlines the superior results of our CSOW methodology. Furthermore, we utilized a machine learning algorithm Long Short-Term Memory (LSTM) to leverage deep learning for automated and accurate recognition of traffic patterns. Our experimental results indicate that LSTMs can successfully and accurately depict our watermark from standard Tor Network traffic in more than 85% of instances while maintaining an absolute 0% false positive rate.

Table of Contents

List of Figures

List of Acronyms

AES: Advanced Encryption Standard

CCG: Criminal Chat Group

CNN: Convolutional Neural Network

CNV: Client Network Vicinity

CSOW: Client-Side Originating Watermark

DL: Deep Learning

DOS: Denial of Service

DS: Directory Server

DT: Decision Tree

DTW: Dynamic Time Warping

DNN: Deep Neural Network

EN: Exit Node

GB: Gradient Boosting

GN: Guard Node

HS: Hidden Services

LSTM: Long Short-Term Memory

MFD: Multi-Flow Detector

ML: Machine Learning

MSE: Mean Squared Error

MTU: Malicious Tor User

OC-SVM: One-Class Support Vector Machine

OP: Onion Proxy

PS: Potential Suspects

RP: Rendezvous Point

RNN: Recurrent Neural Network

RBF: Radial Basis Function

RN: Relay Node

SSOW: Server-Side Originating Watermark

SNV: Server Network Vicinity

SVM: Support Vector Machine

SMOTE: Synthetic Minority Over-sampling Technique

TLS: Transport Layer Security

UDA: User De-Anonymization

ULEA: Undercover Law Enforcement Agent

VOIP: Voice Over Internet Protocol

Chapter 1

Introduction

1.1 Chapter Overview

In this chapter we go over the motivation behind our research and the problems we have aimed to solve. We also describe a high-level application scenario to provide context and outline some of the key assumptions of our work. We conclude the chapter with an overview of the thesis structure and the content of the subsequent chapters.

1.2 Motivation

In today's era of invasive digitalization, which has infiltrated every aspect of our lives, the general public awareness about the importance of online privacy has substantially grown [1]. As a consequence of this awareness, the popularity of anonymity browsers and VPN services that offer affordable protection of online user privacy have also seen a rapid surge [2]. Unfortunately, these

services have inevitably attracted the attention of cybercriminals who seek anonymity to carry out various illegal activities online [75].

In the realm of digital world, the concept of anonymization refers to the general process of removing human/user identifying information. Identifying information exists in many different forms, and can constitute any of the following: user's name, home address, IP address, or anything the user interacts with or produces on a daily basis (such as browsing history or online purchases). Identifying information is commonly utilized and sold by corporations (or in some case by malicious actors) to target users with specialized commercials or malicious phishing schemes. Hence, to protect their privacy and avoid becoming a victim of these campaigns, many users turn to online anonymity services such as Tor. Tor (The Onion Router) is an anonymity browser whose operation is supported by a network of anonymization nodes, which are built off the concept of Onion Routing [3]. Since its launch in 2006, the Tor browser and Tor network have become one of the most popular anonymity services [4]. The graph in Figure 1 illustrates the number of daily users connecting to Tor over the last two years [4], showing a clear trend of increasing popularity. In addition to providing low-latency anonymity to end users, Tor also offers the ability to host anonymous services/servers also known as 'onion services' (often implicated in the context of the Dark Web), while remaining completely open-source and free to the general public. Tor's connection with the Dark Web, and the fact that it serves as the key enabler of various illicit activities online and beyond, is what has inspired many cyber security researchers to look for ways to break Tor's Anonymity [5]. Or, more specifically, to develop effective techniques of Tor-user de-anonymization that could potentially be used against cyber criminals. This is exactly what has motivated the work presented in this thesis.

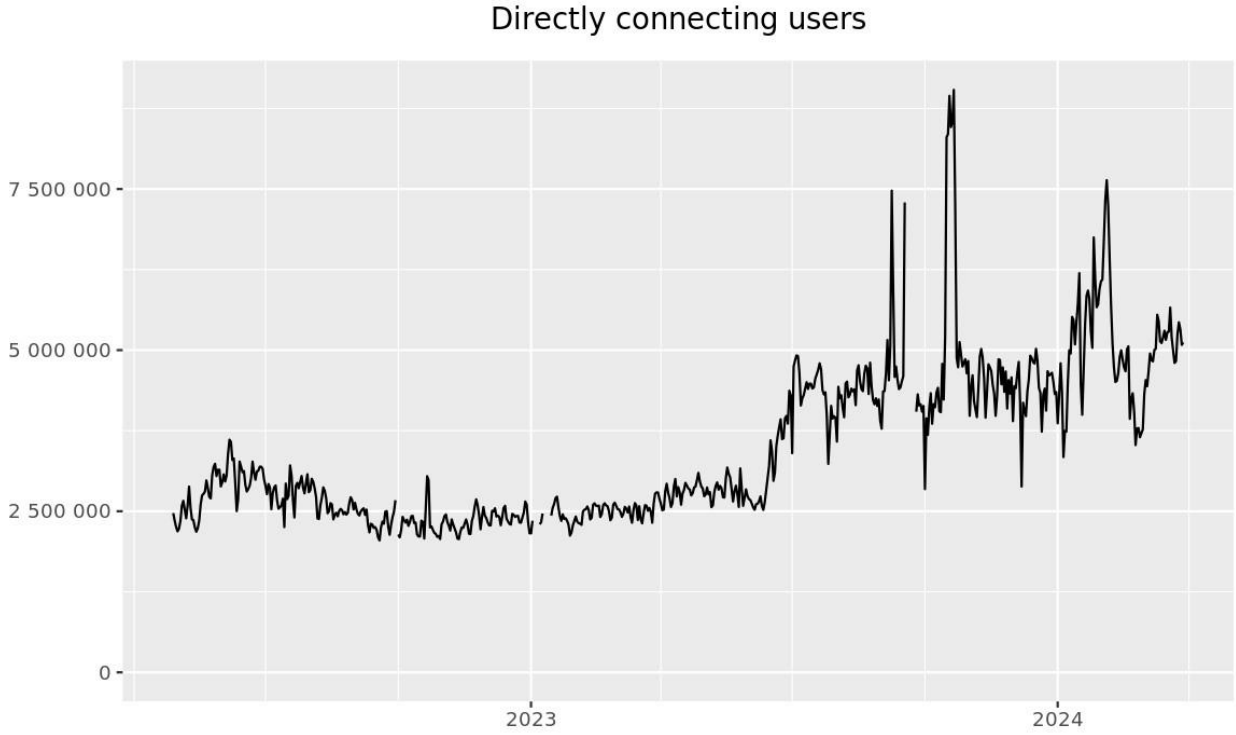


Figure 1: Tor Metric Of Daily Tor User Connections [4]

The process of user de-anonymization is very much the opposite of anonymization - its goal is to uncover (some) hidden identifying information that can ultimately reveal the user's true identity. Exactly what information is being targeted by de-anonymization process varies from attack to attack. In this thesis, and for the purposes of our research, the goal is to reveal the user's real IP address and nothing else.

To date, many Tor-user de-anonymization techniques have been proposed in the research literature. Unfortunately, many of these methods no longer hold up in real-world scenarios. On one hand, this can be attributed to the fact that Tor is open-source and through the efforts of its volunteers and researchers is constantly upgrading and evolving to withstand the latest security and privacy threats. On the other hand, certain de-anonymization methods against Tor, especially

those that utilize server-originating traffic watermarks[1], are set to fail due to their inherent susceptibility to traffic delays and traffic noise, which are generally unavoidable and tend to accumulate inside the Tor network (ultimately making traffic watermark schemes ineffective). An illustration of this phenomenon is shown in Figure 2, which is based on our real-world experimentation and shows: a) the shape of the original traffic watermark as generated by the server (blue line) and intended for one particular target-user, and b) the shape of this same watermark after passing through the Tor network and as observed at the other end of the respective Tor circuit (red line) – i.e., in the target-user's network vicinity. The obvious distortion caused by Tor-network induced traffic delays and noise evidently change the shape of the original watermark beyond recognition, ultimately leading to suboptimal user de-anonymization performance. Our work has been motivated by the need to develop novel more effective approaches to user de-anonymization in Tor, that could overcome the limitations of the existing techniques – including the ones that utilize traffic watermarks, as illustrated in Figure 2.

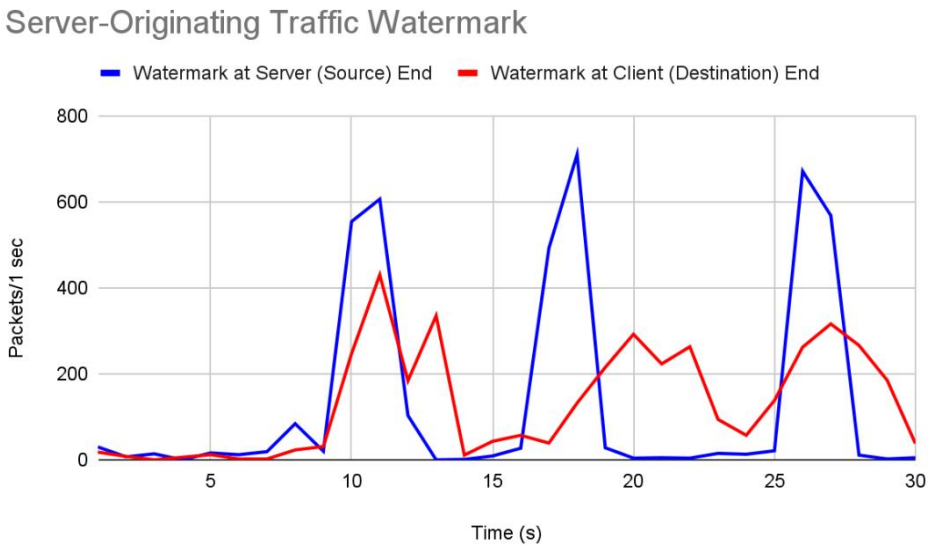


Figure 2: Server-Originating Traffic Watermark as Observed at Two Ends of a Tor Circuit

[1] A traffic watermark is a unique, embedded identifier in network traffic that enables tracking, attribution, or authentication without significantly altering the content of the traffic.

1.3 Application Scenario Considered in This Thesis

As mentioned previously, user anonymization tools are frequently abused by those who seek to conceal their identity while conducting various malicious activities online (e.g., cyber criminals, amateur hackers and hacktivists, nation state groups). Effective and accurate de-anonymization and tracking of these actors is of critical importance for cyber-investigators and threat-intelligence teams affiliated with various organizations - from law enforcement and government agencies to security product and service-provider companies. The work presented in this thesis aims to equip these teams with a novel deanonymization strategy that can aid in accurate identification a particular malicious actor from a group of malicious suspects.

The specific user de-anonymization scenario considered in the research of this thesis is illustrated in Figure 3. As the figure shows, in this use-case scenario, we assume an Undercover Law Enforcement Agent (ULEA) is interested in de-anonymizing a particular Malicious Tor User (MTU) that is an active forum member of a Criminal Chat Group (CCG) hosted on the Dark Web. While ULEA knows (only) the chat group alias of this malicious user, MTU's actual (real-world) identity is not known; though, there are several Potential Suspects (PSs). With an appropriate network surveillance warrant, ULAE and its agency are able to monitor encrypted traffic going to/from each of PS's respective device and/or home network – as shown in Figure 3. (E.g., the surveillance tools could be run by the suspect's ISPs, or a surveillance software could be installed on the suspect's LAN routers.) Unfortunately, since PSs deploy online anonymization tools, traffic surveillance alone cannot help with either of the following: a) determine which specific Internet sites they visit, b) whether they are members of the malicious Dark Web forum in question, and c) which one of them is potentially the actual MTU. In order to reveal MTU's real identity, ULEA

sends a direct/private ‘bait message’ to MTU through CCG (i.e., a message visible only to MTU), with the aim to entice/lure MTU to visit a specially crafted ’decoy’ webpage hosted on ULEA’s private server (see Figure 4). This decoy webpage is carefully designed so that during its rendering by the MTU’s device it triggers the creation of a unique traffic watermark (i.e., a unique pattern of network packets flowing to/from MTU’s device), which stands out from all other traffic and is easily detectable. In other words, if (i.e., as soon as) MTU takes the bait and clicks on the ’decoy’ webpage link (causing MTU’s browser to start the rendering of the page by retrieving its content from ULEA’s private server), the given action will results in the appearance of the unique watermark only in the traffic flow of that particular PS – ultimately allowing ULEA to reveal that this PS is the actual criminal behind the MTU alias. One clear and very important advantage of this technique, from the perspective of general user privacy, is that the anonymity of other suspect users is entirely maintained, while only the identity of the targeted malicious user is revealed/compromised.

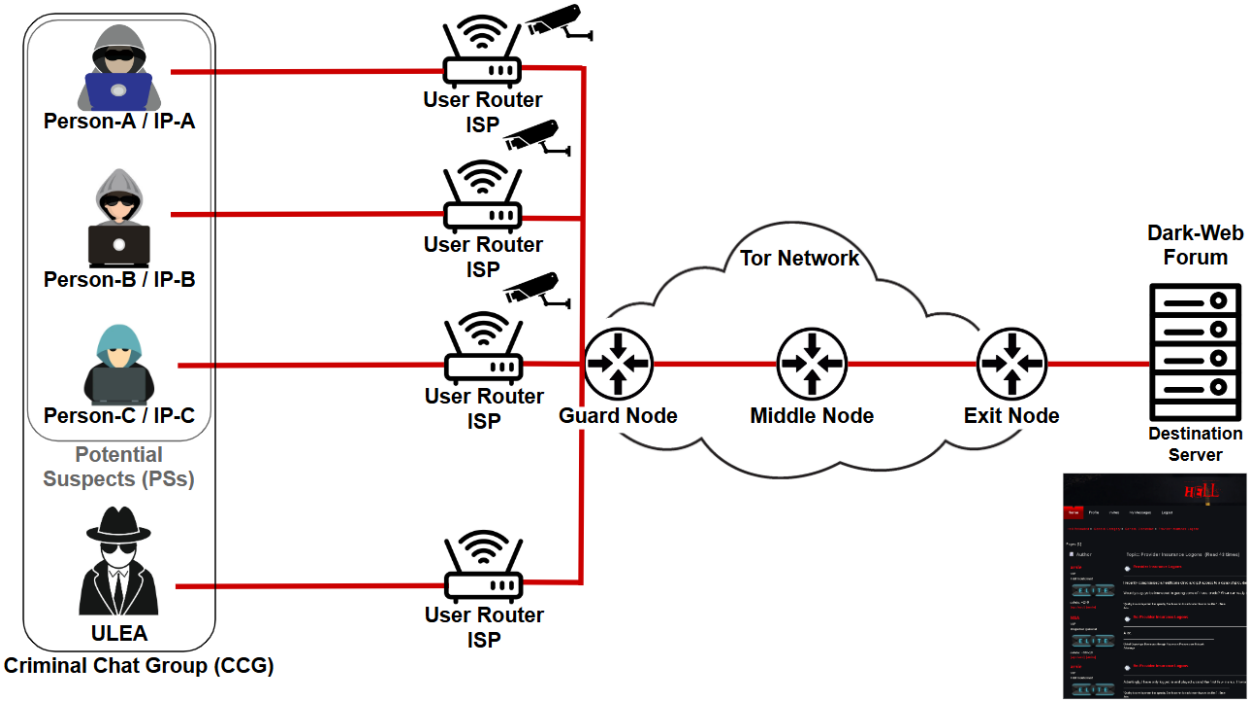


Figure 3: Real-World Use Case Scenario Considered in This Research

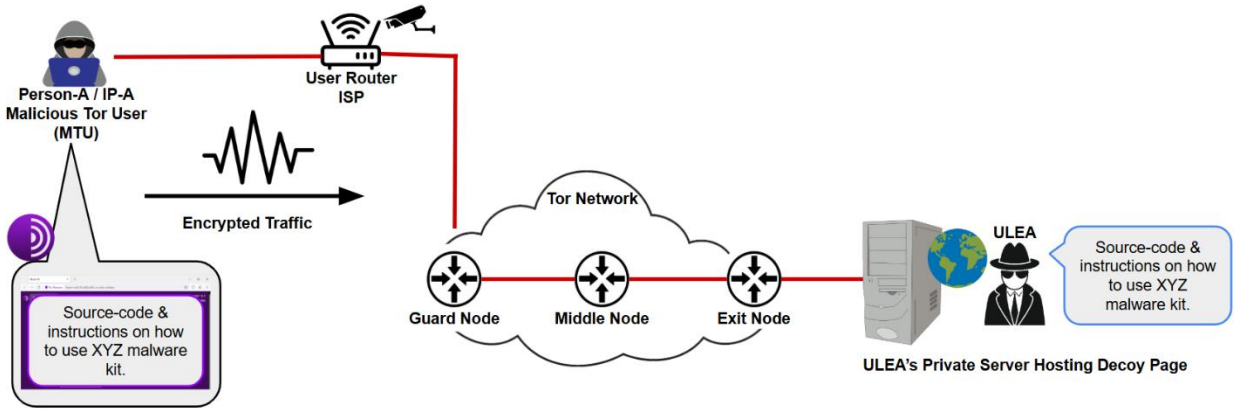


Figure 4: MTU De-Anonymization Using Traffic Watermark

1.4 Thesis Content

The remainder of this thesis is organized as follows: In Chapter 2 we will delve into the general Tor topology to establish a strong understanding of the inner workings of Tor, and we will

provide a survey of previous works on Tor User De-Anonymization and some of their shortcomings. Chapter 3, we will take a closer look into the workings of an existing Server-Side Originating Watermark (SSOW) approach with focus on analyzing its performance and limitations. In Chapter 4, we will provide full coverage of our novel Client-Side Originating Watermark (CSOW) user de-anonymization approach, including: our assumptions, the details of CSOW implementation and our experimental framework, as well as the obtained experimental results. In Chapter 5, we will give an overview of state of the art in deep learning traffic classification algorithms, the purpose of which is to select an algorithm that could/will be utilized to improve the detection capability of our CSOW watermark technique. In Chapter 6, we will delve deeper into the motivations behind the practical utilization of the selected deep learning algorithm (LSTM autoencoder), and we will thoroughly cover its integration into the CSOW user de-anonymization framework. In this chapter we will also present the experimental results obtained using LSTM-based CSOW to demonstrate its favorable practical performance. Finally, in Chapter 7, we will close this thesis with a comprehensive summary of the main contributions of our research and potential future directions.

Chapter 2

Tor Topology & User De-Anonymization Overview

2.1 Introduction

Given its great popularity as well as its connections with the Dark Web, Tor has been the subject of numerous studies looking for effective approaches to user de-anonymization. In this Chapter, we first provide a detailed overview of Tor topology and operation. Subsequently, we present a comprehensive survey and classification of the most impactful works on the topic of Tor user de-anonymization to date – discussing their limitations and the ultimate motivation for our work.

2.2 Tor Topology

Tor is an overlay network that is built on top of the Transmission Control Protocol (TCP) and that utilizes multiple (typically three) encapsulated layers of cryptographic protection, as illustrated in Figure 5. Each layer of encryption is removed by the corresponding relay node, which only knows the previous and next step in the Tor Circuit but not the entire route or the origin. Hence, this layered form of encryption is called "onion". This prevents any single relay node from seeing both the source and destination of the data, ultimately ensuring strong user anonymity.

From the networking perspective, and as illustrated in Figure 5, to actually support communication between a Tor client and a destination server, Tor creates a special circuit which in most cases consists of two relay nodes and one guard node [6]. An exception to this is a Tor circuit involving Hidden Services (also known as 'Onion Services'). Hidden Services are websites or services that operate exclusively within the Tor network, allowing both users and providers to interact anonymously. (By default, Tor does not provide anonymity to service providers, while Hidden Services ensure anonymity for both parties.) While the work of this thesis does not deal with de-anonymization of Hidden Services, it is still important to mention their components and operation, as some of the subsequently surveyed literature discusses attacks against them.

In the reminder of this section, we describe the role of different nodes in the Tor topology and the key mechanisms of Tor circuit establishment.

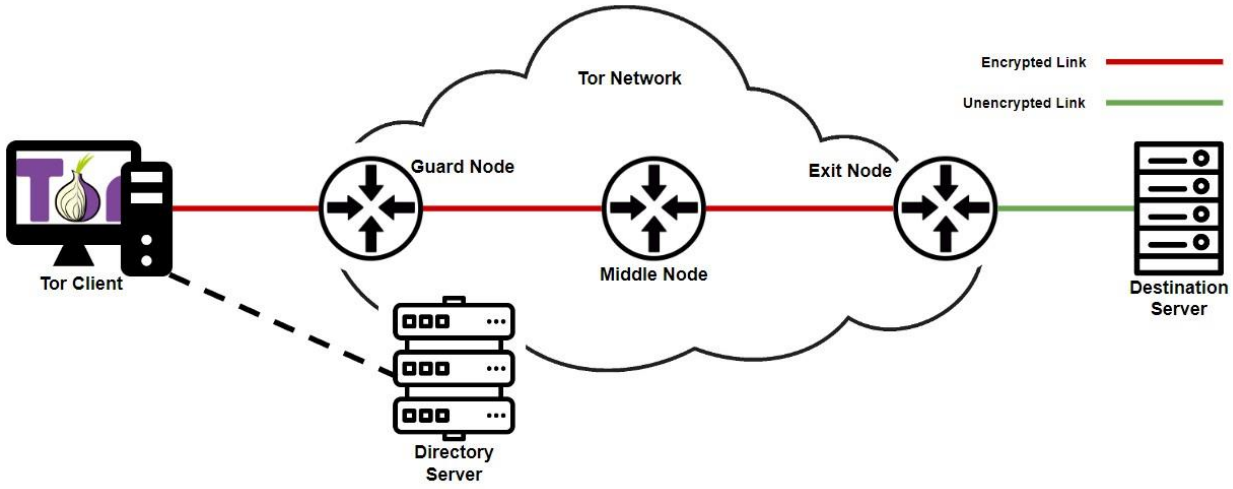


Figure 5: Tor Standard Network Circuit

2.2.1 Standard Client-Server Circuit in Tor

A Tor Relay Node (RN) is a volunteer-run server in the Tor network that helps to anonymize and route users' internet traffic. Specifically, these relay nodes are responsible for execution of onion routing, allowing data to be encrypted and passed over the Internet, while keeping the anonymity of the user's IP and their destination protected. There are three types of relay nodes within a standard client-server circuit, which include: Guard Node, Middle Node, and Exit Node. (In the later parts of this section, we describe each relay type in more detail.)

One of the most critical elements of the Tor network are Tor Directory Servers (DSs). Each Tor DS stores and distributes the list of currently available Tor relay nodes, enabling users to select and then establish connections with one of these nodes [8]. This list is also known as 'consensus document' and it includes information such as: relay IP addresses, bandwidth capacity, exit policies, and other relevant metadata.

The Onion Proxy (OP), also known as the Tor Client, is a software application installed on a user's device that serves as an intermediary between the user's Tor browser (i.e., the user's Tor GUI component) and the Tor network [7]. In particular, OP facilitates communication between the respective Tor client and a Tor's directory server, which in turn allows the client to obtain the list of available relay nodes and establish a circuit within the Tor network.

The Guard Node (GN) is the first (entry) node in an established Tor circuit through which the user's traffic is routed [7]. The OP selects a guard node randomly from a list of trusted RNs (a RN can become a trusted RN after it has been within the Tor network for at least 8 weeks) provided by the DS, and this GN is used as the entry node for all circuits originating from this OP until a different pool of trusted (i.e., potential guard) nodes is selected. The selection of guard node follows a procedure aimed at minimizing the risk of a compromised node potentially being owned/controlled by an adversary becoming the GN [7]. Since the GN is the first (entry) node in the user's Tor circuit, with a direct connection to the user's OP, this node will inevitably know the user's actual Internet Protocol (IP) address. This, in fact, is the reason why many traditional user de-anonymization attacks are aimed at compromising GNs, through which it is then relatively easy to perform user/traffic deanonymization.

The Exit Node (EN) is the final relay in a Tor circuit, after which the traffic exits the Tor network and reaches the destination server. This node acts as the interface between the Tor network (i.e., respective circuit) and the rest of the Internet, and performs the final (i.e., the inner Tor/onion layer) decryption of passing traffic. As the last node in the circuit, EN not only knows the IP address of the destination server but is also potentially able to observe unencrypted traffic exiting the Tor network [7]. This is the reason why, in addition to GNs, Tor's ENs are also frequent targets of user de-anonymization attacks.

2.2.2 Hidden Service Circuit in Tor

Tor in its standard configuration (only) provides anonymity to its users/clients against de-anonymization attacks conducted by: a) the destination server, or b) any potential adversary being able to observe the client's Tor circuit between the Guard Node and the destination server. On the other hand, in this configuration Tor provides far less anonymity protection to the destination server, as the IP address of the destination server is clearly visible not only to the connecting user but also the respective Exit Node. To address this problem and be able to conceal the IP address of the accessed website/server, the Tor community has introduced the concept of Hidden Services (HS), also known as Onion Services, (see Figure 6). A HS is a web server (i.e., domain) hosted on either a node inside the Tor network or an external node, and is assigned a URL ending in ".onion" [9]. The existence of this HS is generally advertised to the public over social media or a more targeted audience on a dark web forum, as standard search engines do not index onion services. Unfortunately, the provided anonymity of HSs is very attractive for cybercriminals seeking to circumvent detection of their illegal websites.

Introduction Points are nodes selected by each HS to register its services with the Tor network. Each HS typically deploys several of these nodes to prevent/mitigate potential impact of Denial of Service (DOS) attacks against a single Introduction Point [7]. Hidden Service Directories are used by the HSs to advertise their Introduction Points and its public key. Given the configuration of the Tor Circuit, the Introduction Points do not know the IP address of the HS as they are connected to the HS via multiple intermediate nodes [10] (as shown in Figure 6).

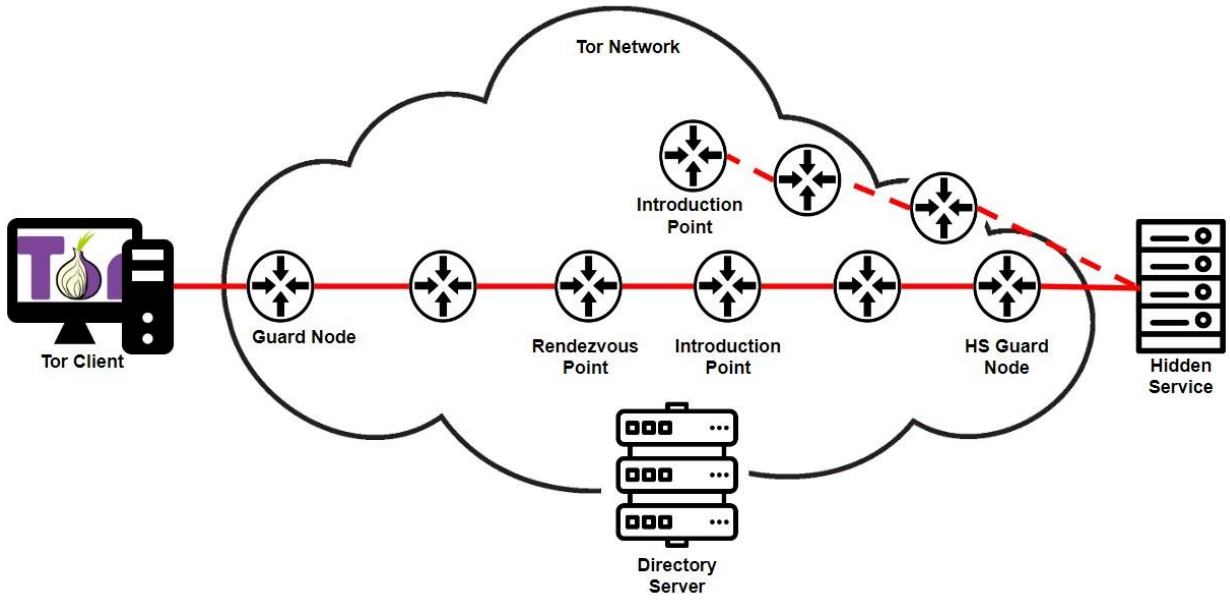


Figure 6: Tor Hidden Service Network Circuit

In the Hidden Service configuration, a Rendezvous Point (RP) is a Tor node that serves the same function as the EN in a standard three-hop Tor circuit. Before the client OP initializes a connection with any of the Introduction Points advertised by the Hidden Service Directories, it selects a Rendezvous Point (RP) [7]. Similarly to a standard Tor circuit, the OP also selects two other nodes (GN and middle) and establishes a Tor circuit to the RP via these nodes. As a result, the RP does not know the IP address of the client.

As mentioned previously, DSs maintain a publicly accessible list of existing relay nodes in the Tor network, and this information is often used by an ISP to censor and block the operation of Tor network. To circumvent this issue, Bridges were introduced as another component of the Tor infrastructure. Bridges are standard Tor relay nodes that are not listed publicly in the main Tor directory. Their position in the Tor circuit takes the place of a GN, which are used by both OP and HS. As a result, ULEAs and ISPs are unable to obtain a complete list of bridge nodes [7]. An OP

or HS that is interested in using a bridge can request bridges directly through the Tor browser or email the Tor project team [11].

2.2.3 Tor Circuit Establishment Procedure

To use Tor services, the user first must install OP software (together with a Tor browser) on its device. To establish a circuit through the Tor network, the OP first contacts a DS and requests a list of active relay nodes on the network. The OP selects three nodes from the list to act as the guard, middle, and exit nodes, and incrementally creates a circuit by exchanging encryption keys with each [7]. The key exchange is done via the Diffie–Hellman handshake. After the Tor circuit that consists of three hops has been established (see Figure 5), the user can start to communicate with the intended destination server over the Tor Network [12].

To increase the difficulty of traffic analysis attacks against its users, Tor uses fixed-length packets – also referred to as 'cells' consisting of 512 bytes [7]. There are two main types of cells, which are *control cells* and *relay cells*. *Control cells* are interpreted at the receiving nodes and can carry Tor commands (e.g., create, padding, destroy, etc.). *Relay cells* carry the actual end-to-end (client-server) data and consist of an additional packet (i.e., relay) header. This relay header includes a stream ID (i.e., the ID of the respective communication stream), end-to-end checksum for integrity checking, a relay command and the payload length. The relay header and the payload are encrypted with the 128-bit counter mode Advanced Encryption Standard (AES) and, using the symmetric encryption keys negotiated during the Diffie Hellman Key Exchange procedure [13].

2.3 User De-Anonymization Attacks in Tor: General Taxonomy

Based on our in-depth survey of the relevant research literature ([7], [14] to [46]), we conclude that most user de-anonymization attacks in Tor are based on the concept of Traffic Correlation, and can be further split into two main subcategories: Timing and Watermarking attacks.

Traffic Correlation (also known as Traffic Confirmation) attacks take place when an adversary has the ability to monitor both ends of a network connection (Tor circuit) – e.g., from the target/victim user end as well as from the respective destination end [15]. In the Tor network the adversary can achieve this objective by either compromising both the Guard and Exit nodes, or by monitoring their respective links out of the Tor network (i.e., between the client and the GN, and between the EN and the destination, as shown in Figure 5). By confirming that the traffic patterns seen at both ends are identical or highly similar, the attacker can directly link a user to their server-side activity.

One special case of correlation attacks are Timing Attacks, where one particular aspect of observed network traffic – timing characteristics – are used to determine correlations between network traffic at the user vs. destination end. In particular, the adversary compares the timing of packets entering the network (e.g., at the GN) with packets exiting the network (e.g., at the EN). If similar timing patterns are observed, the attacker infers that the entry and exit streams are likely correlated. These attacks most predominantly rely on the observed/measured inter-packet arrival time as well as packet rate and latency.

Watermarking Attacks are another special case of correlation attacks where the adversary is able to proactively manipulate Tor's traffic flow (between the target/victim user and the respective server), by either injecting, modifying, delaying or deleting packets. The main goal of a watermark attack is to produce a uniquely distinguishable traffic pattern that can be injected into the network traffic stream at one end and observed and detected at another. Our own novel approach to user de-anonymization (as described in Chapter 4) utilizes this particular attack methodology.

2.4 User De-Anonymization Attacks in Tor: Taxonomy Based on Node Involvement

Another way to categorize various de-anonymization attacks in Tor is by considering the level of control the adversary is required to have over the Tor network and its different components. Based on our survey of the related literature, we distinguish between works in which the adversary needs to control one (of more) of the following aspects of Tor network: a) Guard and Exit Nodes, b) Client or Single Relay Nodes, c) Side Channel (traffic flow), and d) a combination of a) to c). In general, Guard and Exit Nodes attacks require the attacker to control both the entry and exit nodes of the circuit at the same time. Client or Single Relay Node attacks are launched by an attacker that controls a single Tor node. Side Channel attacks are typically carried out by monitoring and/or manipulating the links between different circuit components. It should be noted here that most of the existing Side Channel attacks have been based on the so-called Server-Side Originating Watermarks (SSOW) – i.e., watermarks that are generated by/from the server end of a Tor circuit. The work of this thesis introduces a completely novel Side Channel (i.e., watermark) paradigm, in which watermarks get generated by/from the client end of a Tor circuit, and which we name Client-Side Originating Watermark (CSOW). Lastly, Hybrid Attacks

is a category Tor user deanonymization techniques that are a combination of the former approaches.

Given an extensive volume of related research literature, in the reminder of this chapter we will be surveying 3 most representative and impactful papers per each attack category to keep information condensed. Nevertheless, in Table 7, we systematically group all the surveyed works based on the type of control over Tor components the adversary is required to have in order to launch a successful user de-anonymization attack.

Tor De-Anonymization Techniques				
Paper	**Guard & Exit Node Compromise**	**Single Node Compromise**	**Side Channel (Traffic Watermarks)**	
			Server Origin	**Client Origin**
2004 Wright [14]	X			
2005 Murdoch [40]		X	X	
2007 Abbot [15]	X		X	
2007 Bauer [16]	X			
2009 Fu [17]	X			
2009 Bauer [18]	X			
2009 Evans [41]		X	X	
2009 Herrmann [24]			X	
2010 Bauer [42]	X		X	
2010 Hopper [43]		X	X	
2010 Chakravarty [44]		X	X	
2011 Wang [19]	X			
2011 Le Blond [22]		X		
2011 Panchenko [25]			X	
2012 Ling [20]	X			
2012 Cai [26]			X	
2013 Wang [27]			X	
2013 Sulaiman [45]		X	X	
2013 Chan-Tin [46]		X	X	
2013 Gedds [47]		X	X	
2014 He [28]			X	
2014 Wang [29]			X	
2015 Arp [30]			X	
2016 Panchenko [31]			X	
2016 Hayes [32]			X	
2017 Yang [23]		X	X	
2017 Nasr [33]			X	
2017 Greschbach [34]			X	
2017 Mavroudis [48]			X	
2018 Rochet [21]	X		X	
2018 Sirinam [35]			X	
2018 Rimmer [36]			X	
2018 Zhou [37]			X	
2020 Pulls [38]			X	
2022 Sun [73]			X	
2023 Gegenhuber [74]	X		X	
2024 Brown [39]				**X**

Figure 7: Table Overview of Preliminary De-Anonymization Attacks

2.4.1 Guard and Exit Nodes De-Anonymization

This category of attacks requires an adversary to gain control over both GN and EN of the target/victim Tor circuit, which can be achieved by either compromising some of the existing Tor nodes or by introducing new attacker-controlled relay nodes into the Tor network. (Figure 8 shows the general attack scenario for this category where a malicious entry guard and a malicious exit

node are nodes that are either compromised or run by the attacker.) Conceptually, both compromised nodes also need to be connected to another attacker-controlled device called 'central authority', which is able to view and analyze (i.e., correlate) the data from both ends of the target/victim circuit, ultimately leading to de-anonymization of the respective client/user.

When pursuing this particular type of attack by means of newly introduced malicious relay nodes, certain steps should be taken by the adversary to increase the chances that these specific Tor nodes get selected as an entry or an exit node of the target/victim Tor circuit. For example, malicious Tor nodes can specify that they only should be used as exit nodes and that they support some specific (sought-after) protocols. Furthermore, a malicious node can falsely advertise high available bandwidths and high uptime to be considered a preferred entry guard.

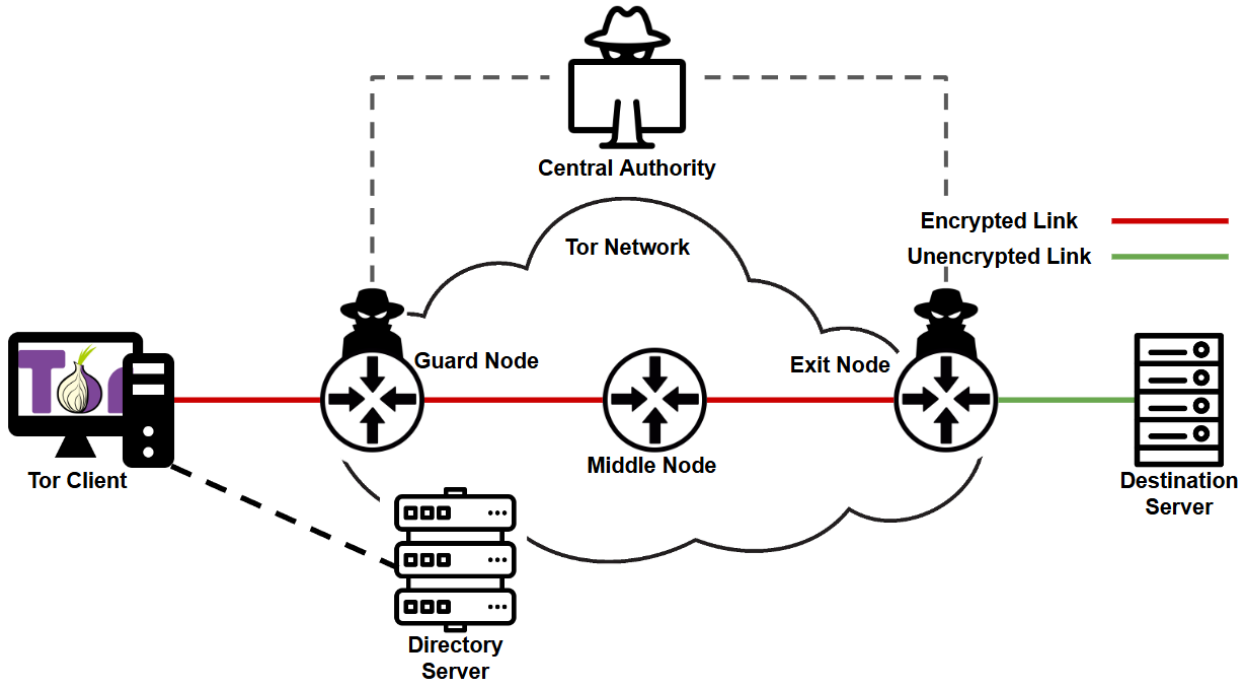


Figure 8: Guard and Exit Node Attack Topology

Wright et al. [14] introduced the so-called Predecessor Attack, a de-anonymization technique that targets users and is applicable to various platforms; however, in this study, we will only discuss its Tor application. In this method, an attacker is assumed to control multiple relay nodes within the network, and it first attempts to identify the communication circuits that consist solely of these particular nodes, using techniques like timing analysis. If the attacker can ensure that an entire circuit comprises only their controlled nodes, they can trace the user's IP address. However, this attack relies on several unrealistic assumptions, specifically: both guard nodes and exit nodes are required to be under the control of the adversary; and, only one user is connected through the circuit at a time. While potentially feasible in the early days of Tor, such assumptions are simply unrealistic in today's real-world scenario, given the Tor's growth in both the number of relay nodes and its popularity (i.e., number of users).

In 2007, Bauer et al. [16] published a paper that introduces low-resource attacks against the Tor network, which exploit Tor's mechanisms for routing optimizations to gain control over GN and EN. Namely, by design, Tor favors high-bandwidth, high-uptime nodes for circuit establishment and routing in order to reduce latency. Inadvertently, this makes it easier for attackers to compromise Tor circuits by controlling/owning such high-bandwidth/high-uptime nodes. Though, attackers do not need to necessarily own such nodes, but instead can conduct the attack by simply misrepresenting their resources to increase their likelihood of selection. Similar to other attacks which aim at controlling both GN and EN these methods have assumptions that are simply unrealistic in a real-world application. Primarily, the likelihood that an adversary would end up controlling both GN and EN nodes belonging to the same circuit for a particular target user is unrealistic, despite using false resource representation.

In 2009, Bauer et al. [18] have described another user de-anonymization attack that exploits the vulnerability of Tor paths while targeting specific type of application traffic, such as SMTP, and peer-to-peer (P2P) protocols. Namely, Bauer points out that applications using specific protocols and ports (e.g., SMTP and P2P) face a higher risk of path compromise. This is due to a limited number of Tor relay nodes that support these ports, which makes it easier for attackers' nodes which support these protocols to be selected as exit nodes. However, such attacks are very niche and not very likely to successfully de-anonymize a targeted user.

2.4.2 Tor Client/Relay Node De-Anonymization

In this category of attacks, the adversary uses a single compromised Tor network component - such as the Tor client of the targeted user or a Relay Node (RN) - to conduct de-anonymization. The functionality of the compromised component is usually altered to match the requirements of the attack (e.g., send periodic traffic bursts/patterns). Some of the attacks in this category are aimed at de-anonymizing HSs, unlike the methods described in Section 2.3.1 which are almost exclusively geared towards de-anonymization of Tor users.

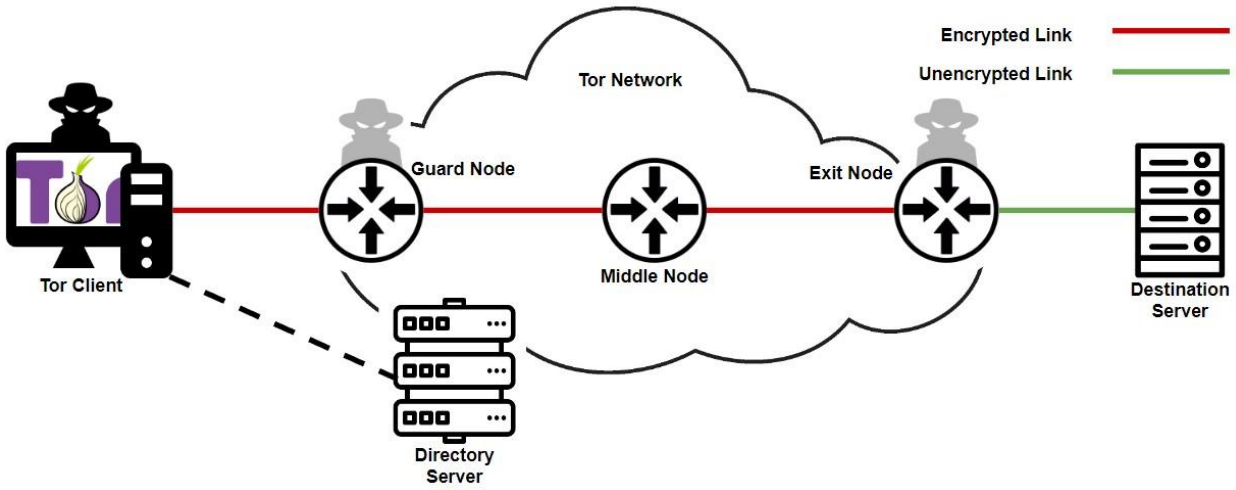


Figure 9: Tor Client/Relay Node Attack Topology

Le Blond et al. [22] introduced a de-anonymization attack that focuses on the vulnerability of Tor users that utilize peer-to-peer (P2P) applications, particularly BitTorrent, to compromise their anonymity. The researchers demonstrate that when a Tor user utilizes an insecure application, like BitTorrent, it can inadvertently expose their IP address. This insecure application or misconfiguration of the application allows the adversary to link the Tor user directly to an adversary controlled EN, thereby bypassing the remainder of the Tor network (i.e., Tor Client connects directly to malicious EN, exposing their IP address). Unfortunately, the assumptions required for this attack are not feasible in a real-world scenario. Particularly, the target user is required to be using a mis-configured BitTorrent application over the Tor network while also having the adversary's malicious EN included in its Tor circuit, all of which is exceptionally hard to accomplish in an actual real-world scenario.

2.4.3 Side-Channel De-Anonymization

Side-channel attacks use means other than compromising the main Tor components to execute a de-anonymization attack. The most common form of side-channel deployed against a Tor user is the intercepted (encrypted) traffic exchanged between this user's Tor client and its respective GN. This channel/traffic is readily accessible by network administrators or Internet Service Providers (ISPs) that manage or control the underlying communication link. It is important to understand that the attacks in this category do not require any element of Tor network to be owned or compromised by the attacker. A special subcategory of side-channel attacks are Passive Side-Channel attacks (i.e., attacks that do not alter the traffic in any way), which very often utilize the concept of 'website fingerprinting'. Website fingerprinting is a technique used to identify

websites that a user visits by analyzing the patterns of encrypted network traffic generated during the website's retrieval/rendering - even without knowing the actual content or purpose of the communication. More specifically, during a passive side-channel attack, the attacker observes the size, timing, and other characteristics of packets sent and received by the victim user, and then matches the observed traffic against the traffic patterns of known websites, potentially revealing sensitive browsing activities despite encryption and anonymization efforts.

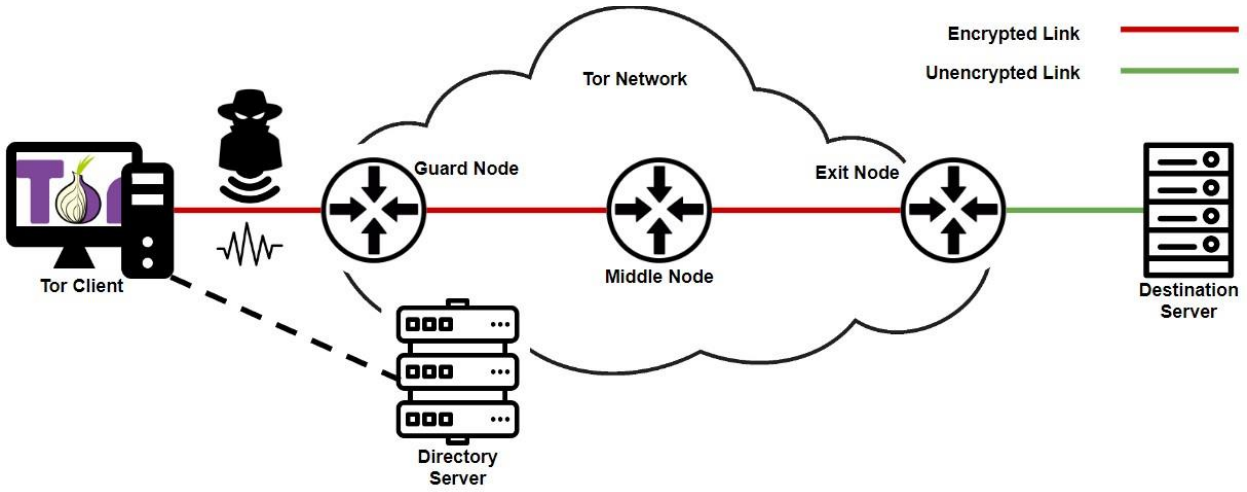


Figure 10: Side-Channel Attack Topology

Panchenko et al. [25] discusses a passive side-channel attack which utilizes website fingerprinting to identify specific sites visited by Tor users. By examining characteristics such as traffic volume and timing, the authors are able to build a machine learning model (specifically, a support vector machine (SVM) model) that is capable of recognizing associated patterns with targeted websites. However, this approach has several drawbacks. Namely, Tor network utilizes a 3-hop system which induces a lot of natural noise that can greatly reduce the accuracy of such attacks. Furthermore, utilizing SVMs with large datasets makes this approach computationally

intensive, paired with the need to constantly re-train the SVM due to changing watermarks (i.e., website fingerprints) – which makes this methodology unrealistic in a real-world setting.

Wang et al. [29] propose improvements for website fingerprinting in user de-anonymization attacks against Tor (and other anonymity networks). They introduce a k-Nearest Neighbors (k-NN) classifier that leverages a feature set to achieve higher accuracy in identifying websites based on traffic patterns. The classifier adjusts feature weights in an effort to improve robustness of fingerprints against various obfuscation techniques. However, Tor induced traffic noise still greatly impacts the performance of such an attack. Other drawbacks of this technique include practical and computational complexity, as the utilized k-NN classifier requires a large training dataset.

Arp et al. [30] introduce a new side-channel de-anonymization attack on Tor called Torben – which, effectively, is a form of 'active' side-channel attack. (In an active side-channel attack, the adversary is able to directly manipulate the side-channel's traffic.) This attack takes advantage of two key scenarios/concepts: 1) the attacker's ability to manipulate a web page so it loads content from an untrusted origin, and 2) the attacker's ability to observe (raw) traffic patterns that correspond to request-response pairs, even though the traffic itself is encrypted. The core idea of the attack is to embed a web page marker in the server's response, creating a recognizable traffic pattern that a passive observer can detect at the user's end. Arp et al. describe two variants of the proposed attack based on the type of marker used. The first variant, deploying 'remote markers', exploits non-malicious (i.e., trusted) web pages that can load content from external sources, such as advertisements. The second variant, deploying 'local markers', involves the use of a web page under direct control of the attacker. Similar to other side-channel attacks, this attack assumes that

the adversary is able to observe the traffic between the Tor client and the GN, and its performance can also be greatly impacted by the traffic noise that inherently accumulates along Tor circuit(s).

2.4.4 Hybrid De-Anonymization

If a mix of Tor network components from the previous categories is used for an attack, we place that attack/work in the so-called Hybrid category.

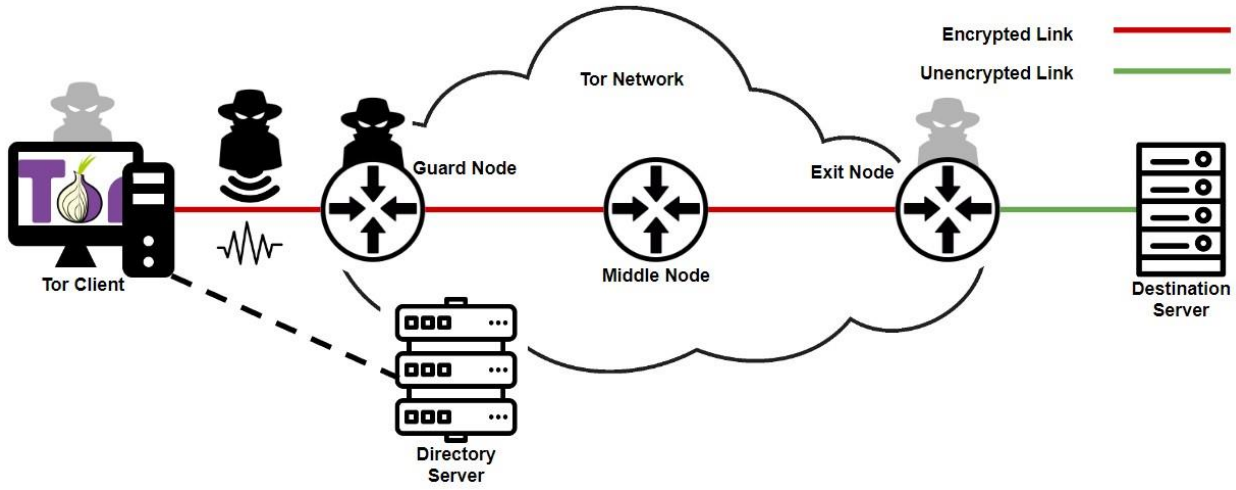


Figure 11: Hybrid Attack Topology

Abbot et al. [15] introduce a browser-based timing attack targeting Tor users' anonymity, leveraging JavaScript and HTML meta refresh tags to perform traffic analysis. The attack involves a malicious Tor exit node inserting JavaScript into web pages accessed by the user. This JavaScript periodically sends a unique signal to an external server, generating identifiable traffic patterns. If, the user's circuit also utilizes a malicious GN, the attacker can correlate this traffic and acquire the user's IP address. Unfortunately, the assumption of a Tor circuit containing both a compromised GN and EN is simply unrealistic in a real-world scenario.

Rochet et al. [21] present a so-called ‘dropmark’ traffic confirmation attack aimed at user de-anonymization. The dropmark attack is an end-to-end correlation method for associating entry and exit traffic flows in Tor. It works by marking traffic with dropped packets at specific timing intervals, which a guard node can detect. The method capitalizes on Tor's handling of unrecognized cells and achieves high correlation rates without visibly impacting user experience. Similar to standard GN and EN de-anonymization attacks, it is unrealistic to assume that both GN and EN are compromised.

Yang et al. [23] propose an active website fingerprinting user de-anonymization attack against Tor. Unlike some of the passive fingerprinting techniques discussed in our side-channel category, which simply observes traffic, this active method manipulates data packets at the GN (controlled by the adversary), introducing delays in HTTP requests to separate distinct web objects, thereby creating a clearer traffic pattern for classification. This is done through the following steps: by controlling an GN, the attacker actively delays selected HTTP request packets, effectively separating web objects in Tor’s encrypted traffic. After delaying the traffic, the attacker records the modified packet sequences and extracts identifiable features, such as cell positions and transmission patterns. Finally, using machine learning classifiers, the attacker compares these features to a known dataset to identify the likely website being accessed. Similar to other techniques, this approach is also based on a few unrealistic assumptions. Firstly, in today’s Tor, it may be hard for the adversary to actually control the GN of the target user. Secondly, this attack only works on HTTP applications, meaning that in the case of HTTPS client-server communication the attack would not work. Finally, the deliberately induced delay in the client-server traffic can negatively affect the user browsing experiences and thus raise the risk of detection.

2.5 Conclusion

In this chapter we have presented the main categories of user de-anonymization techniques in Tor, and we have surveyed the most relevant and impactful research works in each of these categories. Overall, we found that many of the existing user de-anonymization techniques simply have unrealistic assumptions, or deploy methodologies that are not easy to execute in today's real-world environment. Nevertheless, we believe that side-channel methods are the most feasible ones as they do not require actual compromise of any Tor network components. This is what motivated us to propose our own side-channel user de-anonymization approach that can effectively address the shortcomings of the earlier techniques. Our novel approach (which is presented in Chapter 4) is partially based on the ideas of Torben [30]; hence, in the next chapter, we provide a more detailed overview of Torben as well as our initial experimentation using this de-anonymization technique.

Chapter 3

Study of Torben's Server Originating Watermark Effectiveness

3.1 Introduction

It was explained in Chapter 2 that considering the current size and popularity of Tor, the side-channel user de-anonymization techniques are not only most effective but also significantly less involved (i.e., less complex to execute) for the adversary. One common feature of all existing side-channel user de-anonymization techniques is the fact that they rely on server-side originating watermarks (SSOW). In our survey, we have identified Torben [30] as one of the most promising SSOW side-channel user de-anonymization solutions, as it requires the adversary to control only the destination server while being able to passively observe the first hop of the target user's Tor circuit. The first stage and objective of our research has been to evaluate Torben's real-world performance and identify its potential drawbacks. In this chapter, we showcase the main aspects of our experimentation with a Torben based SSOW user de-anonymization framework. We also

present samples of the real-world results obtained using this framework, which highlight the key drawbacks of Torben and have motivated the development of our novel Tor-user de-anonymization technique using client-side originating watermarks (CSOW) – which will be presented in Chapter 4.

3.2 Torben Based Server-Side Tor User De-Anonymization: Our Implementation

The user de-anonymization approach named 'Torben', proposed by Arp et al. [30], is a technique that deploys server-side originating watermarks, as depicted in Figure 12. The technique allows two possible implementations (i.e., actual origins of the watermark) - one via an online ad injected into a third-party webpage, and the other via a webpage that is in full control of the attacker. In our evaluation of Torben we have utilized the second approach. From the page-level implementation both approaches are very similar. However, in terms of the actual mounting of a Torben attack the second approach is practically more feasible, given the fact that the first approach would require a hacking of a legitimate website. It is important to note that as a side-channel method, this approach makes no specific assumptions about the Tor network, and the attacker does not need to own/control any of its elements. Instead, it is assumed that the attacker (only) has full control of a webpage and some means of listening to the network traffic arriving at the machine running the victim user's Tor client/browser. E.g., if the victim user is on a public WiFi, the arriving traffic can simply be sniffed off the wireless medium. On the other hand, if the victim user is on their home network, the attacker (i.e., ULEA from Chapter 1) needs to either compromise the victim's router or obtain a warrant for traffic monitoring by the user's ISP in order to observe (i.e., acquire) the passing traffic.

In terms of its actual implementation, the main premise of Torben de-anonymization attack is to embed one or more 'markers' (e.g., special-purpose HTML tags or JavaScript snippets) into the source code of the attacker-controlled webpage, which the victim user is lured into visiting using the approach previously described in Chapter 1. As also previously explained, the actual purpose of the HTML markers is to cause the target user's browser to generate a unique traffic/packet pattern (i.e., watermark) during the downloading/rendering of the adversary's webpage. By observing this specific watermark in the network flow that streams from the server hosting the attacker's page to the machine of a particular victim user, this user can be fully identified/de-anonymized [30].

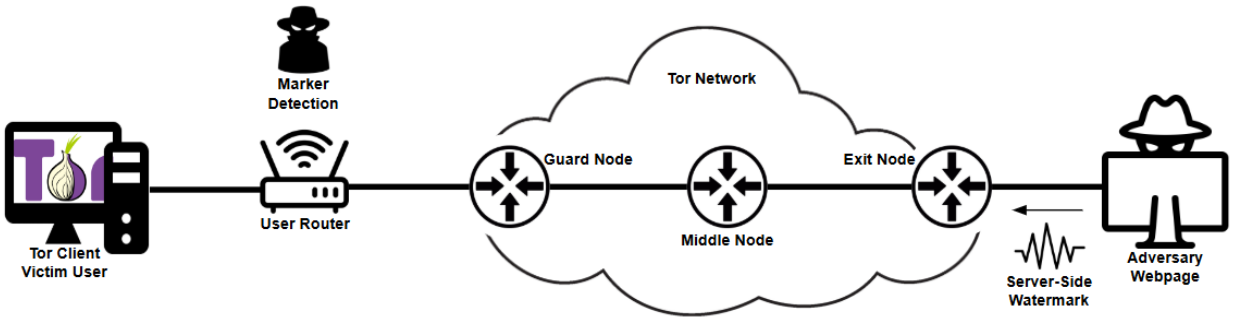


Figure 12: Server-Side Originating Watermark Topology

In our evaluation of the Torben attack, and assuming the role of the adversary, we have created an HTML watermark-generating webpage which the target-user would be lured into visiting. (This page can be accessed from: [80].) On the surface, our Torben webpage appears to be rather ordinary; however, the HTML body of the page contains several pieces of 'malicious' JavaScript code – i.e., code that results in the creation of a desired traffic watermark. (Further details of this code and the webpage appearance can be found in Appendix 1.) Namely, when

executed, this code loads an image into each of the three already defined HTML division (“ ”) elements which are invisible to the user until certain (predefined) delays are met. The delays are simple in the sense that they only check for a certain amount of time to pass in order to t induce additional traffic (i.e., trigger the retrieval of the aforementioned image), thereby resulting in a unique traffic watermark. The deployed image is approximately 835kB in size, and its purpose is to generate large and observable traffic peaks during the rendering of the adversary webpage. With three division elements in the adversary webpage, the image downloading process is repeated three times, with an inter-request delay of 7500ms between each division element.

As for the stealthiness of our Torben implementation, the image that appears in our attack page and is responsible for the creation of the traffic watermark is made visible to the user. Nevertheless, the image could easily be made hidden (e.g., using HTML tags to reduce it display size to <0 by 0> pixels), without affecting the attack performance. Namely, the size in which an image is displayed on a webpage does not impact the number of bytes transmitted during its transfer/downloading from the server to the client. Of course, in the case of a more advanced user (i.e., user that inspects the source code of each downloaded page for potential signs of malicious intent or activity) some form of code obscuration may be necessary to make the attack stealthier. Note, however, this aspect of Torben attack was never discussed by its original authors [30], and thus was out of scope of our research as well.

3.3 Torben Based Server-Side Tor User De-Anonymization: Our Experimental Results

The real-world experimentations with our own variant of Torben attack have revealed major performance issues that ultimately render suboptimal de-anonymization results. (We believe these issues are inherent to all other side-channel de-anonymization techniques that deploy server-originating traffic watermarks, such as those presented in [28,29,31,32].) Namely, since the watermark generated by the server has to pass through the Tor network before reaching the target user - and the respective watermark detector set up by the adversary in the target's network vicinity (see Figure 12) - the watermark inevitably ends up being corrupted by different types of traffic noise which accumulates along the Tor's pathways. (This could be either Tor specific as well as general Internet traffic noise). In some cases, the noise can cause the watermark to get altered beyond the point of recognizability. Figures 13 and 14 are samples of our experimental results with Torben that illustrate this point. In particular, Figure 13 shows the original 3-spike watermark as captured in the network vicinity of the server. Figure 14 shows the appearance of the same 3-spike watermark when finally captured/observed by the watermark detector in the network vicinity of the victim's machine. The detrimental impact of the network noise on the look of this watermark is very evident since the watermark captured in the target-user vicinity appears to contain numerous (i.e., certainly more than three) ill-shaped spikes.

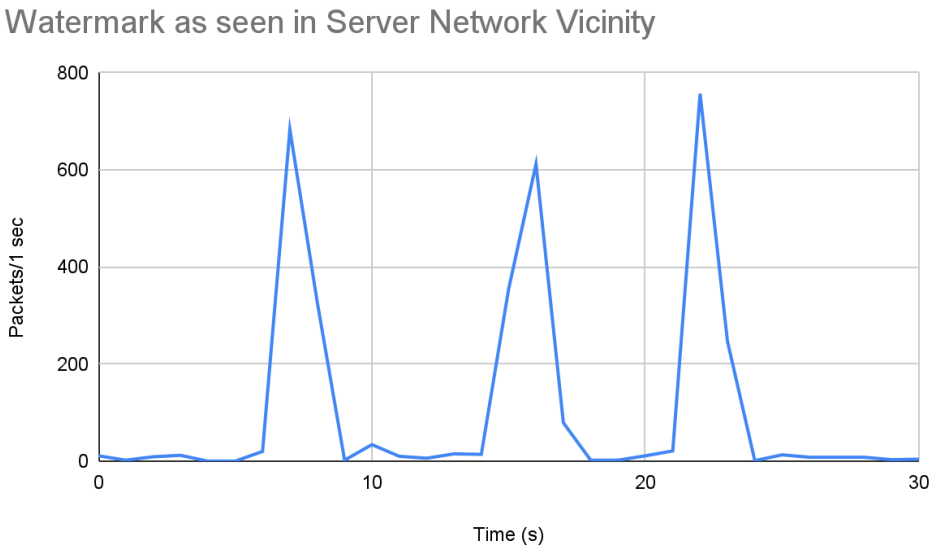


Figure 13: Server-Side Originating Watermark (SSOW) as Captured in Server Network Vicinity

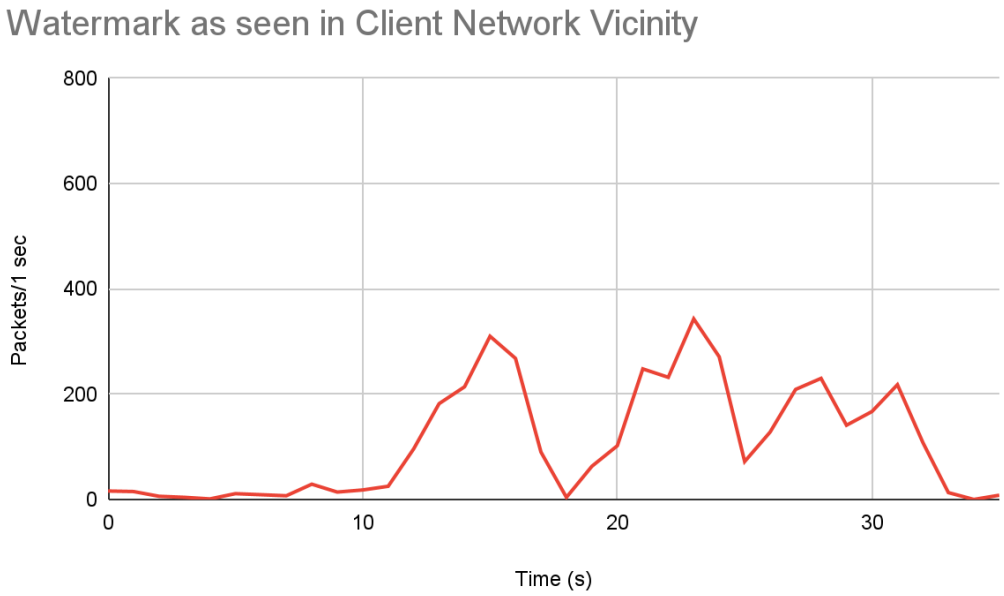


Figure 14: Sever-Side Originating Watermark (SSOW) as Captured in Client Network Vicinity

3.4 Conclusion

To summarize, in this chapter we have outlined our user de-anonymization framework based on a Torben inspired implementation of server-side originating watermark (SSOW). The experimental results obtained using this framework showcase some key drawbacks of Torben in a real-world environment. Specifically, SSOW techniques are shown to be highly susceptible to traffic distortion while passing through the Tor network, which can cause alteration of the actual watermark beyond recognition. In the following chapter we present our novel user de-anonymization approach based on client-side originating watermark (CSOW) methodology, and we compare the performance of CSOW against that of SSOW based techniques.

Chapter 4

Novel Tor User De-anonymization Technique Deploying Client-Side Originating Watermark (CSOW)

4.1 Introduction

In order to overcome the challenges of user de-anonymization techniques in Tor that rely on server-side originating watermarks (such as Torben [30]), we propose a novel Client-Side Originating Watermark (CSOW) paradigm. In particular, as illustrated in Figure 15, we propose an approach in which the 'markers' (i.e., special malicious traffic-generating JavaScript elements) placed inside the adversary's page are designed in a way so as to trigger transmission of modulated traffic bursts (i.e., traffic watermark) from the victim's Tor browser towards the server hosting the given page – i.e., in the direction opposite to the one assumed by Torben [30]. With this inverted watermark traffic-flow scheme, which now ensures very close network

proximity of the watermark's origin and the watermark detector (see Figure 15), the effects of traffic noise on the appearance of the captured watermark are greatly reduced, ultimately resulting in much higher accuracy of the user de-anonymization process. In this chapter, we give an overview of the implementation details of our CSOW scheme, and we present some of our experimental results which prove the performance superiority of this scheme over those that rely on server-side originating watermarks.

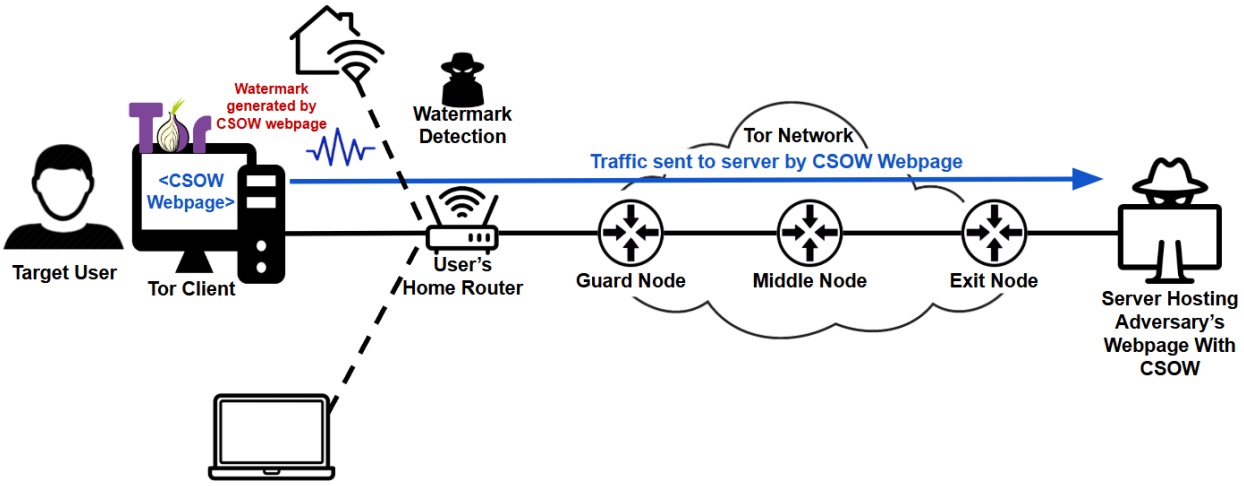


Figure 15: Client-Side Originating Watermark (CSOW) Topology

4.2 CSOW User De-Anonymization Technique: Assumptions

Before we further elaborate on the technical aspects of our Client-Side Originating Watermark (CSOW) user de-anonymization scheme, let us outline our key assumptions concerning the attacker and the target user.

Assumption 1: The adversary conducting the user de-anonymization attack owns (or has managed to successfully compromise) a publicly accessible webpage. For the reminder of this

document we will refer to this webpage as ‘CSOW-page’. The CSOW-page’s content is structured/coded in such a way that it generates a unique ‘in upload direction’ watermark while being loaded (i.e., rendered) by a browser of any user requesting this page.

Assumption 2: By deploying some of the social-engineering schemes discussed in Chapter 1, the adversary manages to successfully lure one (or more) users specifically targeted by CSOW attack into requesting the CSOW-page.

Assumption 3: By holding a strategic position that allows monitoring of the communication link between the target-user’s machine and its respective ISP router (i.e., the first hop of the respective Tor circuit), the adversary is able to observe if/when the watermark appears in the traffic flowing over the monitored link – see Figure 15.

Assumption 4: The target-user is (may be) visiting webpages other than the CSOW-page. However, we assume that all webpages are visited ‘in sequence’; or, in other words, no two pages are being rendered at the same time by the user’s browser.

Assumption 5: The target-user’s dwell-time on each requested page is long enough to ensure that all relevant objects of that webpage are fully retrieved before the next page is requested or the tab/browser is closed. Put another way, the entire ‘traffic fingerprint’ of each webpage is always acquired, and never overlaps with the ‘traffic fingerprint’ of another page. (Note, in the reminder of this thesis we will use the terms ‘traffic fingerprint of a webpage’ and ‘webpage fingerprint’ interchangeably.)

Assumption 6: Based on the shape of the observed traffic - specifically the end-tail of a web-page fingerprint which is typically characterized by a significant drop in traffic intensity -

attacker is able to detect/observe when the retrieval of one webpage is over and the retrieval of new web-page has begun.

It should be noted that the above assumptions have been deployed by most previous works on user de-anonymization using traffic-watermarking, and our research (i.e., CSOW technique) is not expanding the list.

4.3 CSOW User De-Anonymization Technique: Implementation Details

In the course of our research, we have tested a variety of methods for creating the inversed (i.e., upload direction) client-side originating watermarks (CSOWs), most of which have relied on the use of HTTP Post Requests. As it stands, the best approaches we have identified for creating Post Requests that produce significant, reliable, and well-shaped bursts of traffic are the ones that utilize images (i.e., image bytes) already present in the attacker controlled CSOW-page requested by the target-user. More specifically, this approach operates as follows: In order to send the bytes of an image present in the (CSOW) webpage back to the server, it is necessary to utilize HTML 'Canvas' element [76], since simply using a POST request on an 'image' element would only send out its source (URL). Once the Canvas is created, the desired image is drawn onto the canvas and then used to crate a Blob, where JavaScript 'Blob' element [77] represents a file-like object/chunk of raw data (in our case the raw data of the manipulated image), which can be used as a payload for the Post Request(s). Once this is completed, an ordinary fetch function is used to send the actual Post Request. The full code is shown in Figure 16.

```
function sendPostRequest(imageElement, url) {
    // Create a canvas element to draw the image
    var canvas = document.createElement("canvas");
    var context = canvas.getContext("2d");
    // Set the canvas dimensions to match the image element
    canvas.width = imageElement.width;
    canvas.height = imageElement.height;
    // Draw the image onto the canvas
    context.drawImage(imageElement, 0, 0);
    // Convert the canvas content to a Blob object
    canvas.toBlob(function (blob) {
      // Create a FormData object to prepare the image for upload
      var formData = new FormData();
      // Add the image blob with a file name
      formData.append("image", blob, "image.jpg");
      // Send the FormData using the Fetch API
      fetch(url, {
        method: "POST", // Specify HTTP method
        body: formData  // Include the form data in the request body
      })
      .then(function (response) {
        // Parse the server response as JSON
        return response.json();
      })
      .then(function (responseData) {
        // Log the parsed JSON response to the console
        console.log(responseData);
      })
      .catch(function (error) {
        // Handle and log any errors that occur during the request
        console.log(error);
      });
    }, "image/jpg"); // Specify the output format for the Blob
  }
```

Figure 16: Post Request Example Code

Two specific implementation that we have devised, based on the above procedure and ensuring the best possible watermark-generating results, include:

- CSOW Implementation 1, which utilizes a for-loop to send a specified number of consecutive Post Requests and thus generate traffic spikes of desired amplitude (i.e., a desired watermark) – as shown in Figure 17.

```
function loopRequests() {
  for (var i = 0; i < 10; i++) {
      sendPostRequest(imageElement, url);
    }
}
  // Send the POST requests every 7.5 seconds
  setInterval(loopRequests, 7500);
```

Figure 17: CSOW Implementation 1 Example Code

- CSOW Implementation 2, which utilizes a JavaScript function “Promise.all()” [78] to send a pre-set amount of Post Requests concurrently – as shown in Figure 18. ”Promise.all()” is a utility function which takes an iterable input and returns an output that is a combination of all elements in the iterable input. The output is resolved only when all elements have been resolved. As a result, an array of resolved elements from the original iterable input is returned. In pseudo terms, Implementation 2 consists of the following steps. First, we get the reference to the image we wish to use for Post requests (out of potentially multiple images appearing on CSOW-page). With the image chosen, and by calling the sendPostRequest.js function, we create an array of POST requests that use the bytes

of the given image as their payload. Next, we send all requests concurrently using the ”Promise.all()” utility function. The implementation/operation of this function can be separated into three components, ”.then”, ”.catch”, and ”.finally”. The first two components are both response indicators - where ”.then” is an indicator of successful completion, while ”.catch” returns an error code should something unexpected occur. The third component ”.finally”, is where additional steps pertaining to the function can be executed. In our implementation, the first step under “.finally” is to increment a global counter to keep track of the total number of iterations performed. The next step under “.finally” uses an if statement to check if a preset maximum number of iterations have been reached, at which point the function terminates. If the number of iterations has not reached the maximum, a recursive call for sendReqeust.js function is made.

```
var interval = 7500; // Interval between a set of requests
var totalIterations = 3; // Total number of iterations (Number of Spikes)
var totalRequests = 1; // Total number of requests per iteration
var currentIteration = 0; // Counter variable to track the current iteration
var imageElement; // Global variable for storing the image

function sendRequests(imageElement) {
  // Create an array of concurrent POST requests
  var requests = Array.from({ length: totalRequests }, function () {
    return sendPostRequest(imageElement);});

  // Send all requests concurrently
  Promise.all(requests)
    .then(function (responses) {
      console.log("All requests completed:", responses);
    })
    .catch(function (error) {
      console.log("Error occurred:", error);
    })
    .finally(function () {
      currentIteration++;

      // Check if all iterations have been completed
      if (currentIteration < totalIterations) {
        // Send the next set of requests after the interval
        setTimeout(function() {sendRequests(imageElement);}, interval);
      }
    });
}

// Event Listener to ensure the image has loaded so that global varaible is not null
document.addEventListener("DOMContentLoaded", function() {
  imageElement = document.getElementById("1marker"); // Chosen image to be used as payload
  setTimeout(function() {sendRequests(imageElement);}, interval);
});
```

Figure 18: CSOW Implementation 2 Example Code

The main advantage of Implementation 2 (using “Promise.all()” to simultaneously send a number of Post Requst) over Implementation 1 (a simple for-loop that sends Post Requests consecutively) is the fact that Implementation 2 is able to produce larger-amplitude and narrower-in-time traffic spikes, which together result in better shaped and thus more accurately identifiable traffic watermarks. This point is illustrated in Figure 19, where the green watermark (CSOW

Implementation 1) is contrasted against the blue watermark (CSOW Implementation 2), and as observed by the watermark detector located in the client's network vicinity (see Figure 15).

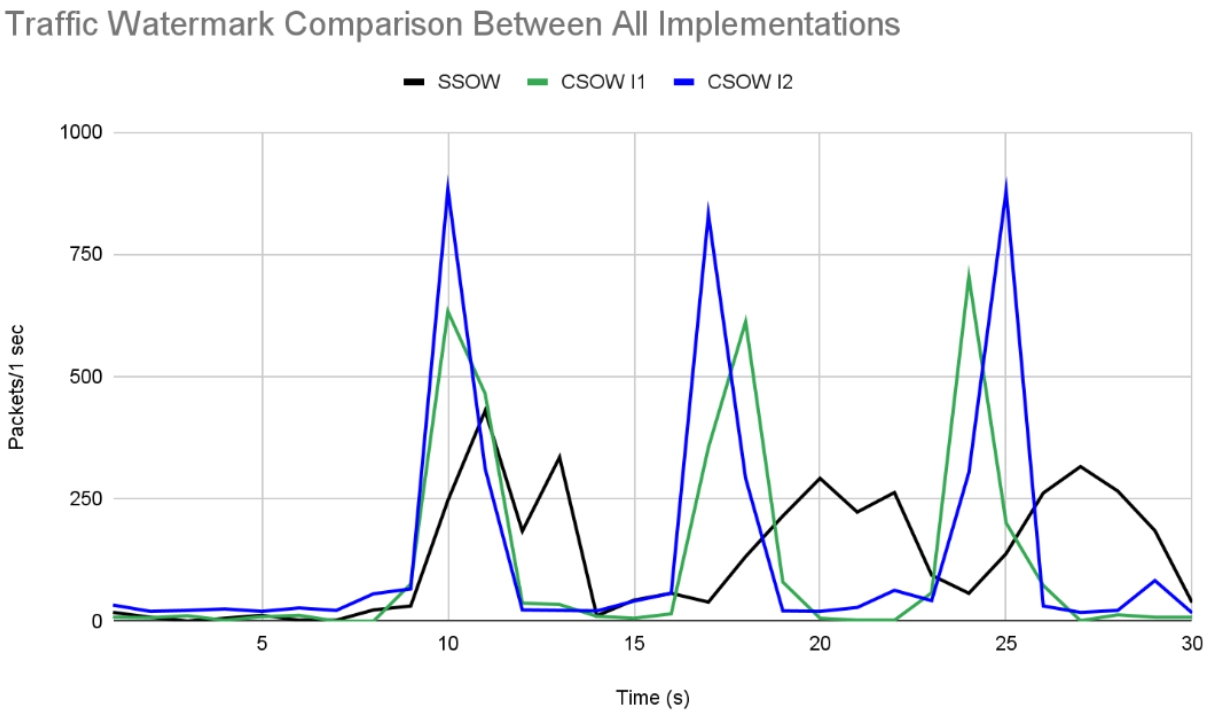


Figure 19: Traffic Watermark Comparison Between All Implementations

Additionally, Figure 19 confirms not only the superior performance of CSOW Implementation 2 (CSOWI2) over CSOW Implementation 1 (CSOWI1) watermark, but also demonstrates that both of these watermarks are considerably less susceptible to traffic noise and are able to produce traffic spikes of considerably larger amplitude relative to those produced by SSOW approach.

4.4 CSOW User De-Anonymization Technique: Experimental Setup

For the purpose of collecting traffic data generated during an actual real-world CSOW attack, we have set up an experimental environment as depicted in Figure 20. In this environment, the adversary's webpage is hosted on a York University's server located at the Keele campus in Toronto. (The webpage is hosted on a personal domain provided by York University to Lassonde students.)

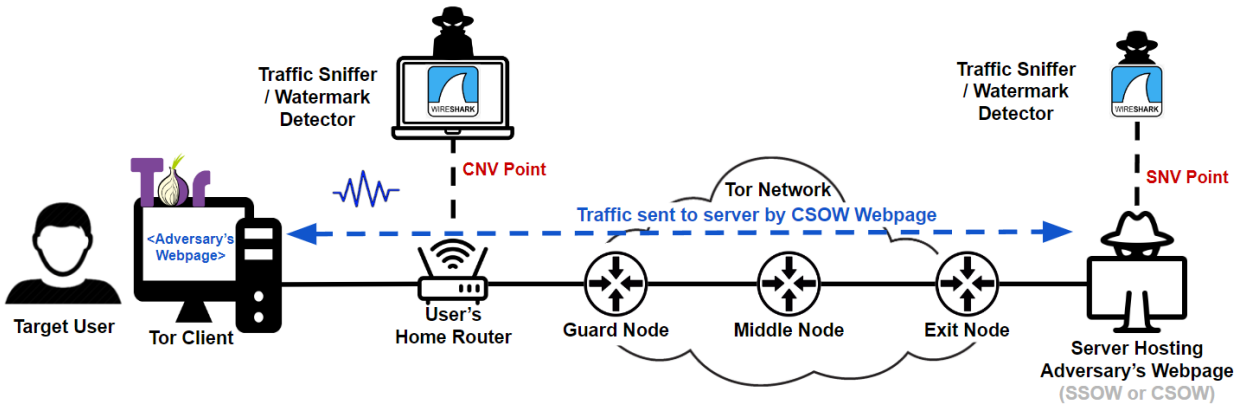


Figure 20: Traffic Capture Points (CNV = Client Network Vicinity, SNV = Server Network Vicinity)

The target user is located in a home/residential network within the GTA area, and is serviced by one of the leading Canadian ISPs. A WireShark packet sniffer is deployed on a Windows device that is located in the same home network (and is connected to the same home router), but is different from the device used by the presumed target user. The WireShark sniffer is tasked with continuous monitoring of the communication link (i.e., the traffic exchanged) between the target-user's machine and the respective home router. We refer to (i.e., mark) this specific WireShark location as Client Network Vicinity [CNV] Point.

The process of collecting traffic data generated during an SSOW attack is conducted within the same general experimental environment as shown in Figure 20. However, given that WireShark

packet sniffing is generally not permitted within the York University's network domain, for the purposes of SSOW attack simulation we had to 'flip' the physical locations of the server and the client. In particular, during the SSOW attack simulation, the adversary's webpage was hosted on a Linux machine located in the previously described home network, while the target user was now located at York University. The traffic exchanged between the SSOW server and the home router was collected in a similar way as described in the case of CSOW attack – though, now the location of the traffic sniffer was referred to (i.e., marked) as Server Network Vicinity (SNV) Point. In this attack scenario, the same victim user machine located at York University network was used to run the Tor client as well as the WireShark sniffer responsible for the CNV-Point traffic collection.

As for the procedure for measuring of the fidelity and/or similarity of the captured traffic watermark(s), we had considered several options suited for time-series data, but ultimately chose Dynamic Time Warping (DTW) method [10]. (Appendix 2 provides details on the workings of DTW when applied to traffic traces, as was the case in our research.) The main reasons we opted for DTW are:

1) DTW is able to handle small and irregular variations in time-series data, commonly referred to as jitter. This is very important as Tor deliberately induces many of these smaller-scale distortions over their network in order to combat some possible correlation attacks by causing misalignment between corresponding traffic points. From the perspective of traffic watermarks, this implies that two points of an identical watermark that gets transmitted at two different time instances may be allocated rather different time-stamps. Put another way, DTW will align two sequences even if they are not fully synchronized, which is very useful if one is

looking to measure the similarity of two watermarks transmitted in/through noisy network environment(s).

2) DTW takes amplitudes of a time-dependent data series into account, unlike other common metrics such as Time-Series Correlation. This is extremely important as one of our key watermark characteristics is the amplitude (magnitude) of traffic peaks. Without the amplitude information taken into account, we lose an important aspect of watermark measurement, which could lead to incorrect matches or mismatches, and consequently result in a significant number of false-positives or false-negatives.

In our analysis, DTW algorithm is applied to watermark captures as the first step in assessing their similarity – with the intention to best align their respective time-points. (In the reminder of this document, watermarks transformed with DTW for purposes of their comparison will be referred to as DTW watermarks.) Subsequently, to determine the actual similarity score between two DTW watermarks, we calculate Euclidean Distance between their respective points. A Lower Euclidean Distance implies a greater similarity between the watermarks, and vice versa. Or, in the context of SSOW and CSOW attack, a greater similarity between the original and captured watermark implies a higher certainty that the user in question had visited the attack webpage – which is the key for successful user de-anonymization (as explained in Chapter 1).

4.5 CSOW User De-Anonymization Technique: Experimental Results

The goal of this experimentation was to perform a comparative analysis of the user de-anonymization effectiveness of the traditional server-side originating watermark (SSOW)

approach (as described in Chapter 3) against our novel client-side originating watermark (CSOW) approach (as described in preceding sections of this chapter). Specifically, the experiments were conducted to compare the quality of SSOW and CSOW watermarks (each produced/generated in a separate experiment), and as appearing at the Client Network Vicinity (CNV) Point – i.e., the point where the adversary is assumed to conduct surveillance of the target-user's in- and out- going traffic, as shown in Figure 20.

Overall, two main groups of experiments were conducted:

1) SSOW Experiments, in which the target user is lured into accessing a SSOW webpage controlled by the adversary.
2) CSOW Experiments, in which the target user is lured into accessing a CSOW webpage controlled by the adversary.

In each set of experiment (SSOW Experiments and CSOW Experiments), 10 repeated and independent retrievals of the attacker's decoy page by the target-user (i.e., by the target-user's Tor browser) were performed – resulting in a total of 20 independent experiments. In both sets of experiments, SSOW and CSOW pages were designed to ensure that the original watermark traffic consisted of 3 main traffic spikes and were generated by transmitting the same overall volume of traffic – approximately 900 kB of traffic per traffic spike. By 'original watermark traffic' we mean the SSOW and CSOW traffic as first appearing at their respective sources – i.e., at the moment the watermark traffic is first placed 'on the wire'. In each individual experiment, during each decoy page retrieval, the passing traffic was observed (i.e., collected by WireShark sniffer) at the CNV Point.

From each group of 10 retrievals (involving SSOW and CSOW watermark, respectively), we selected one representative capture (the capture with the best performance i.e., highest amplitude, fewest distortions, etc.) and showcased them in Figure 21 – where black graph is a capture corresponding to the server-side generated traffic-watermark, while blue graph is a capture corresponding to our novel client-side traffic-watermark. (The opaque red traffic-watermark will be discussed later).

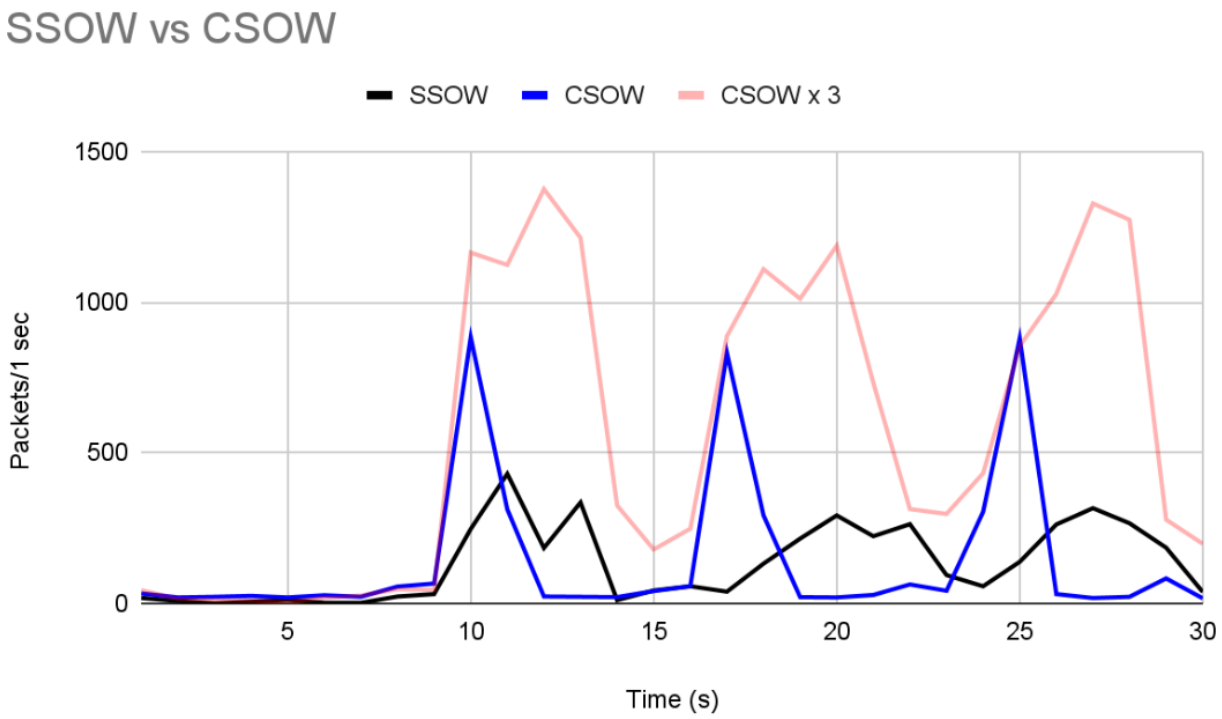


Figure 21: Comparing SSOW against CSOW

A simple visual inspection of the watermarks in Figure 21 shows that time-wise both types of watermarks (SSOW and CSOW) have a duration of approximately 30 seconds. However, the traffic amplitudes of our CSOW method are much better preserved - reaching peaks of over 800 packets/second versus approximately 400 packets/second in the server-side approach - making the

CSOW watermark stand out much better and thus be much easier to detect. It should be noted here that the distortion in the SSOW watermark is evident not only in the lower and partially flattened amplitudes of individual traffic peaks, but also in the presence of 'additional peaks'. Namely, even though the original watermark (as generated by the server and observed at the SNV capture point) consisted of 3 main and well-formed traffic bursts, the delays and noise imposed on traffic as it moved over the Tor pathways appear to have caused these bursts to get deformed, 'stretched', and ultimately broken into multiple smaller bursts.

Next, to quantify the performance of each of method individually (SSOW and CSOW), and their actual susceptibility to noise, we compare two different DTW watermarks from the same approach as observed at the CNV capture point. For this purpose, the two DTW watermarks of the same attack approach selected for relative comparison are: a) the watermark with the best performance (i.e., highest amplitude, lowest disturbance) vs. b) the watermark with the worst performance (i.e., lowest amplitude, highest disturbance). Figure 22 illustrates two DTW watermarks from SSOW attack. It is pretty obvious that the time alignment of these DTW watermarks is fairly good; however, where SSOW approach clearly fails is in the consistency of the watermarks' traffic peaks (amplitude, y-axis). Namely, even with the help of DTW alignment, the amplitudes of the two instances are noticeable different from one another, and ultimately result in Euclidean Distance of 1210. Likewise, Figure 23 illustrates two DTW watermarks from CSOW attack. Clearly, with CSOW approach, we are seeing a good alignment in the time of arrival as well as the amplitude (i.e., traffic peaks), ultimately resulting in a much lower Euclidean Distance of 120. (Recall, a lower Euclidean Distance implies a greater similarity). Based on these observations, we can state that the CSOW attack approach is far less sensitive to noise, and as such is able to ensure more robust detection and, ultimately, more reliable user de-anonymization.

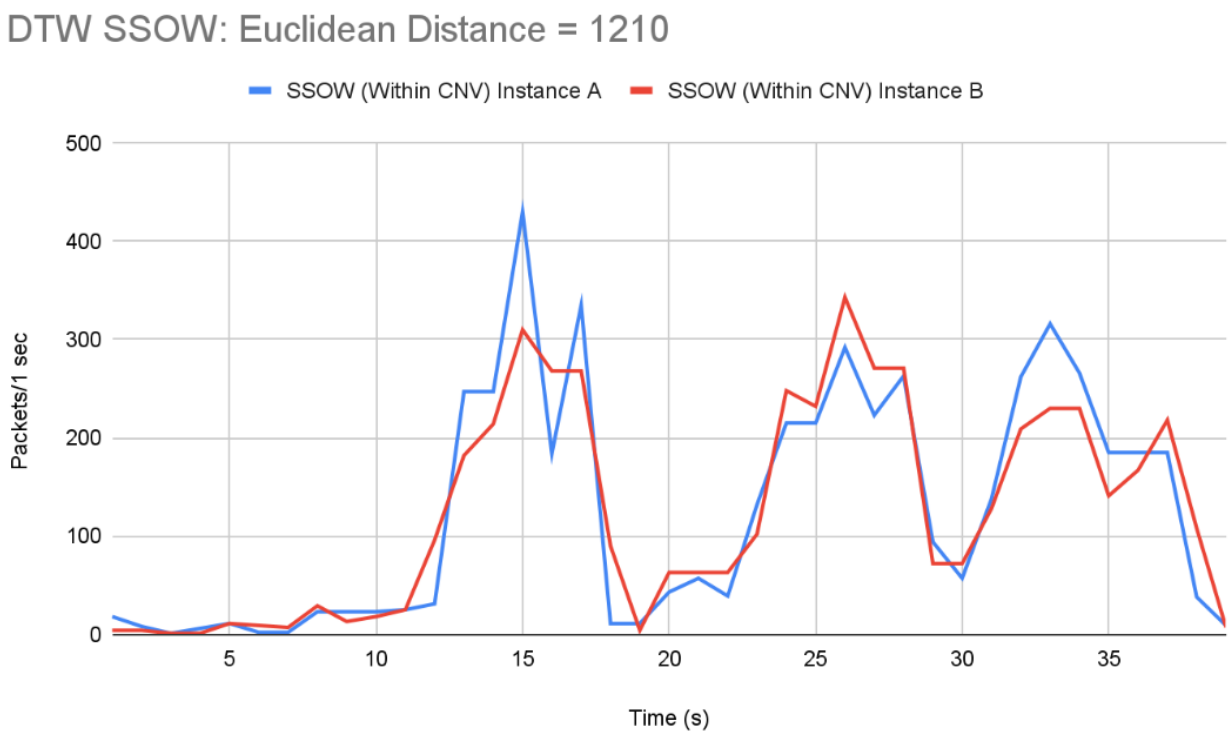


Figure 22: Two Representative Instances of Server-Side Originating DTW Watermarks as Observed at CNV

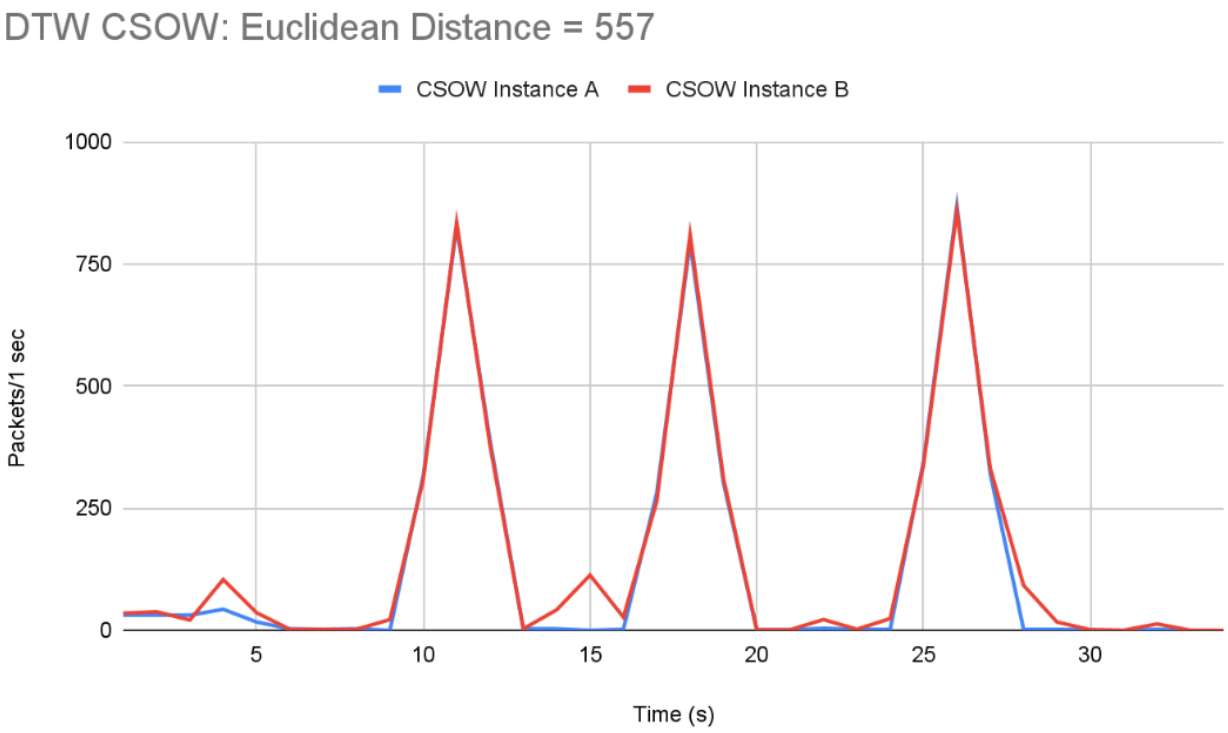


Figure 23: Two Representative Instances of Client-Side Originating Watermark (CSOW) as Observed at CNV

One final remark in terms of the obtained results is related to the potential (in)effectiveness of deploying CSOW watermarks with excessively large traffic peaks. To understand this point, note that back in Figure 21 the opaque-red traffic-watermark is the result of our attempt to triple the number of POST requests in the CSOW attack, and thus potentially further improve the size and detectability of the produced watermark. Unfortunately, our experimentation shows that sending this many post requests did not achieve the desired objective, and it only resulted in a performance drop (i.e., an overly distorted watermark). In more general terms, this implies that attempting to increase the number of POST requests beyond a certain point will only result in deformed and spread-out traffic bursts/peaks. Therefore, during a real-world utilization of the CSOW de-anonymization technique, it is critically important to balance the size of watermark spikes and not attempt to send too much data over a short period of time.

4.6 Summary and Important Open Issues

To summarize, in this chapter we have outlined the six key assumptions of our CSOW user de-anonymization framework (which are also deployed by SSOW techniques of previous research works). We have also described the implementation details of our CSOW technique, and presented the experimental setup used to validate the performance of this technique in the real world. The obtained results showcase the obviously superior performance of the CSOW based user de-anonymization against that of the SSOW approach.

It should be pointed out that most of the analysis of this chapter has been based on simple (manual) analytical and visual inspection of collected watermark(s) and traffic captures. By 'inspection' we refer to the process in which the original CSOW watermark is compared against another (random) packet capture in order to determine whether the given capture contains the original CSOW watermark - as illustrated in the below figure.

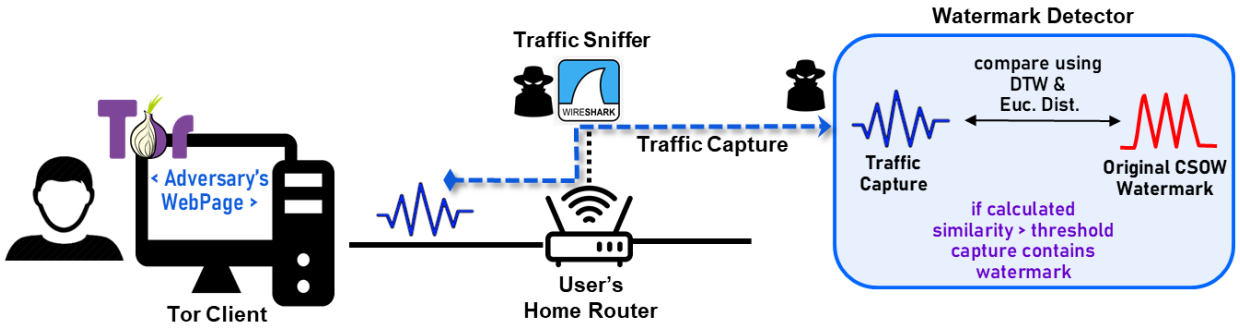


Figure 24: Initially assumed operation of CSOW based user de-anonymization system

To better understand Figure 24, note that our procedure for obtaining/establishing of the '**Original CSOW Watermark**' has consisted of the following two steps:

Step 1: The retrieval of the adversary's CSOW page is initiated from the victim user's Tor browser (i.e., machine), during which the network traffic passing to and from this machine is collected. We will refer to the entire traffic capture collected during this step as '*full watermark traffic capture*'.

Step 2: From the *full watermark traffic capture* **only the 'outgoing'** Tor packets - i.e., packets sent from the victim user's browser/machine to the respective Guard Node and effectively carrying the upload traffic triggered by the rendering of the CSOW page - are isolated, ultimately constituting the CSOW watermark.

(It should also be noted that in theory the Watermark Detector in Figure 24 could deploy a CSOW watermark collected on another communication link during the retrieval of the CSOW page by another machine/browser. But, given that each machine/browser behaves in a unique manner when retrieving and rendering a web-page, such a watermark could potentially yield suboptimal results. We discuss this particular point in more detail in Chapter 6.)

In Figure 24, the '**Traffic Capture'** fed into the Watermark Detector is acquired in a similar manner as the **Original CSOW Watermark**, and it consists only of outgoing Tor packets collected on the respective communication link during the retrieval of an 'unknown' web-page by the victim user. Clearly, a high degree of similarity between the Traffic Capture and the Original CSOW Watermark would imply a high likelihood that the 'unknown' web-page being retrieved by the victim user is actually the CSOW page.

Now, in this chapter we have discussed potentially detrimental impact of traffic noise on some aspects of watermark detection – primarily on the 'shape' of (random) Traffic Capture (Figure 24). For simplicity purposes, we have assumed that the 'Original CSOW Watermark' remains largely unaffected by noise; or, more specifically, that extracting of the CSOW watermark

from one single traffic capture will provide accurate watermark detection and user de-anonymization results. However, we do recognize that in some real-world environments and/or situations, the Original CSOW Watermark itself could potentially be impacted by traffic noise (e.g., if during the watermark capture there are other networking applications running on the victim's machine and causing variable delay to different segments of *full watermark traffic capture*).

Therefore, if we are to build a truly effective and robust CSOW user de-anonymization framework, it would be necessary to incorporate into this framework an automated module capable of building a well-established noise-resistant watermark 'profile' from (not only one but) a number of observed watermarks - as illustrated in the below figure. This very idea is what has motivated our research presented in the subsequent chapters, where we pursue the development of an automated CSOW watermark detector using a Deep Learning algorithm.

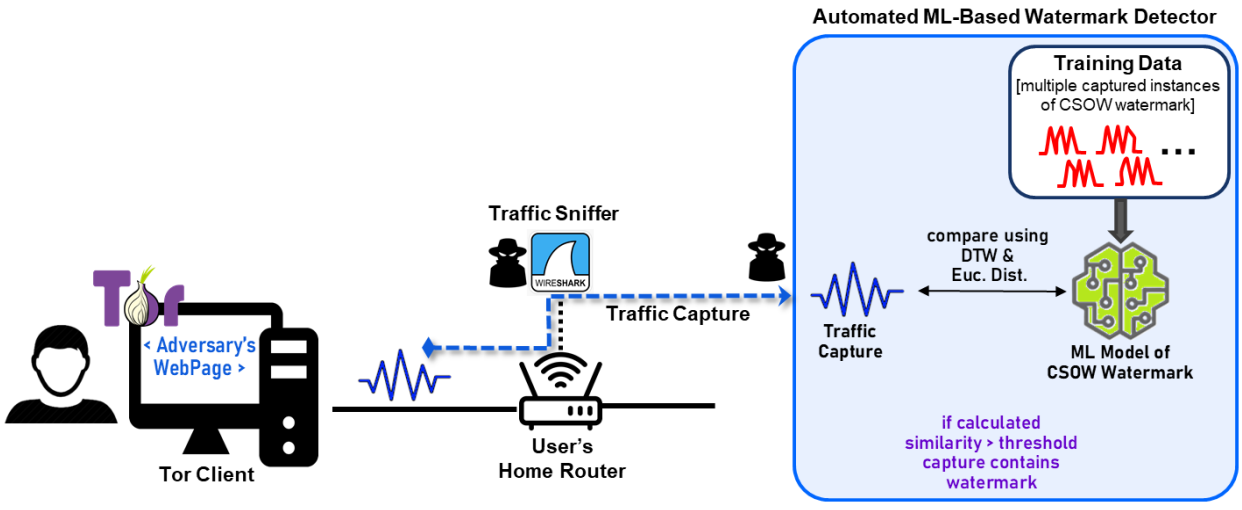


Figure 25: De-anonymization system with an automated ML-based CSOW watermark detector

Chapter 5

Deep Learning for Traffic Classification: Overview

5.1 Introduction

As outlined in the conclusion of Chapter 4, one shortcoming of our initial implementation of CSOW-based user de-anonymization technique is its lack of built-in robustness against possible presence of traffic noise in the benchmark CSOW watermark. Namely, the CSOW solution outlined in Chapter 4 creates/extracts the 'Original CSOW Watermark' profile out of one individual traffic capture. Obviously, if this single traffic capture happens to be affected by noise, the fidelity of the extracted CSOW watermark as well as the accuracy of the entire watermark detection process could be significantly impaired. This shortcoming has inspired us to look for possible ML-based solutions for building a robust and generic model/profile of the CSOW watermark – a model/profile that would be trained on a number of CSOW watermark instances collected at different times (i.e., under different conditions and noise levels). One

challenge, though, in identifying such a solution is the fact that each CSOW watermark is effectively an instance of time-series data. Hence, the selected ML algorithm would have to be able to operate on time-series data.

In this Chapter we provide a survey of previous research work on the use of machine and deep learning solutions for the purpose of network traffic and/or time series data classification. The main focus of the survey are papers specifically discussing Tor traffic classification and anomaly detection problems. In total we survey nine high quality papers, and group them into 3 categories based on the type of traffic being classified and the chosen model – as shown in Figure 26. The main takeaways of the survey is that LSTM model [79] seems to be the most suitable ML algorithm for automated building of a robust CSOW watermark.

Deep Learning Traffic Classification Survey	
Tor Traffic Classification Non-LSTM	
Paper	**Finding**
2019 Radha [53]	Showcases reduced false-positives from DL models and outlines the increase in efficiency over larger data.
2020 Sarkar [54]	Showcases significant increase in performance of DL models over traditional ML models.
2023 He [58]	Showcases increased performance and efficiency of FlowMFD over other DL models.
Tor Traffic Classification LSTM	
Paper	**Finding**
2021 VishnuPriya [56]	Showcases increased performance and robustness of RNN-LSTM versus traditional ML models.
2021 Sarwar [57]	Showcases increased accuracy of CNN-LSTM over other DL models, furthermore outlines performance in identifying type of traffic (browser, video-stream, audio stream, etc.).
Non-Tor Traffic Classification LSTM	
Paper	**Finding**
2020 Akbari [55]	Showcase inadequacies of DL models relying on features from TLS handshake header fields, propose novel feature that utilizes LSTM to increase performance and robustness.
2020 Elsayed [61]	Showcases a novel LSTM-Autoencoder model with improved performance and accuracy, furthermore lower computational overhead allowing for practical real-time deployment.
2023 He [59]	Showcases the superior performance of LSTM models versus traditional ML models for identifying encrypted versus non-encrypted traffic.
2023 Malekghaini [60]	Showcase the effects of data drift of encrypted traffic on standard ML models and illustrates the superior performance LSTM based models in combating the data drift issues.

Figure 26: Deep Learning Traffic Classification Survey Table

5.2 Tor Traffic Classification Using Non-LSTM Models

5.2.1 Detection and Classification of Tor Traffic using Deep Neural Network Learning

In their 2020 paper “Detection of Tor Traffic using Deep Learning”, Sarkar et al. outline their approach to identification and classification of Tor traffic using deep learning techniques [54]. The authors utilize the UNB-CIC Tor and non-Tor real-world dataset, which includes 8,044 samples labeled as Tor and 59,784 samples labeled as non-Tor traffic. The UNB-CIC dataset was originally acquired by setting up three users to perform various activities on the Tor network and two users to conduct non-Tor traffic activities, while capturing their aggregate traffic using

WireShark. The dataset includes 28 different features, including: source and destination IP addresses, ports, flow duration, packet counts, and inter-arrival times.

The methodology employed in this study involves a preprocessing phase where two prominent feature selection methods were applied: Symmetric Uncertainty (SU) and Correlation-Based Filter Selection (CFS). SU is based on information theory and measures the uncertainty of a random variable, helping to identify relevant features by reducing bias. CFS evaluates the predictive power of an attribute and its redundancy relative to other features, ensuring the selection of the most significant attributes for the final model. Using these methods, the original 28 features were reduced to a more manageable subset of 11 features, optimizing the model's overall performance.

The machine learning models used in this research are deep neural networks (DNNs) of various configurations, specifically designed to handle the complex task of traffic classification. The models consist of multiple layers of non-linear processing units, with neurons in each layer responsible for mapping inputs to outputs. During the training phase, the weights of the neurons are adjusted using backpropagation and gradient descent algorithms to minimize the classification errors.

Evaluation metrics such as precision, recall, F-measure, accuracy, and the Area Under the Receiver Operating Characteristic Curve (AUC-ROC) are used to validate the models. (70% of the original dataset are used for training, 10% for validation, and 20% for testing.) The authors' best performing configuration was able to achieve an accuracy of 99.89% in distinguishing Tor traffic from non-Tor traffic. For the multi-class classification task, where the goal was to identify specific types of Tor traffic, the model achieved an accuracy of 95.6%.

The authors acknowledge that their methodology has a few weak points and limitations. Primarily, the DNN models developed for detecting Tor traffic are vulnerable to Generative Adversarial Network (GAN) attacks. GAN synthetic traffic is not detected by the authors' DNN classifiers. Furthermore, the authors discuss potential issues with overfitting of the DNN models when they are trained on a specific dataset and then evaluated on a separate synthetic dataset. The authors raise concern that the model might not generalize as effectively to entirely new datasets or in real-world applications where the traffic characteristics can differ significantly from the training data. Furthermore, such model requires a substantial amount of training data (the others had nearly 70,000 samples) and training time could take upwards of 10 hours.

5.2.2 Tor Traffic Classification using Feed Forward Deep Neural Network Learning

In 2019, Radha and Upadhyay published a paper entitled "Tor Traffic Classification using Deep Learning" [53]. The study employs a deep learning approach, specifically a Feed Forward Neural Network (FFN), to analyze and classify Tor traffic. The researchers have utilized the UNB-CIC Tor/non-Tor dataset for training and evaluation of their FFN model. The methodology involves extracting various time-based features from network traffic data, such as Forward Inter Arrival Time (FIAT), Backward Inter Arrival Time (BIAT), Flow Inter Arrival Time (FLOWIAT), among other time-based features. These features help in distinguishing the characteristics of Tor traffic from regular internet traffic.

The deep learning model is trained using Keras and TensorFlow frameworks, with the architecture comprising multiple hidden layers to capture complex patterns in the data. The performance of the model is improved by using backpropagation which adjusted the weights of the neurons based on the error between the predicted and actual outputs. The authors have

experimented with different numbers of hidden layers and found that eight hidden layers provides optimal results.

The authors compare the performance of their FFN deep learning model with various traditional machine learning algorithms, including Naive Bayes, Logistic Regression, Support Vector Machine (SVM), and Random Forest. Their results indicate that the deep learning approach achieves a higher accuracy in classifying Tor traffic compared to the other traditional methods. Specifically, the deep learning model attains an accuracy of 98.27%, slightly outperforming the Random Forest classifier, which achieves 98.26%. The performance of other classifiers, such as SVM and Logistic Regression, exhibits significantly lower performance, with accuracies of 93.40% and 93.86%, respectively.

The authors emphasize the importance of reducing false positives in detecting non-Tor traffic, and highlight that the deep learning model performs better in minimizing these errors compared to traditional machine learning methods. They also note that while the dataset used in the experiments was relatively small, the deep learning model's performance is expected to improve with larger datasets, where traditional machine learning algorithms may require more computational resources to retrain.

The methodology proposed for Tor traffic classification using a deep learning-based feed-forward neural network unfortunately contains limitations in several areas. The authors acknowledge that the dataset used is relatively small for deep learning systems, which may hinder the generalizability of the model to diverse network scenarios. Additionally, while the model performs well with larger datasets, the increased computational demand for training on such datasets is not thoroughly addressed. Furthermore, the reliance on features like inter-packet arrival times and flow durations assumes consistent traffic patterns, which may not hold in real-

world applications with variable network conditions, particular network conditions such as ones found in Tor.

5.2.3 Tor Traffic Classification using Convolutional Neural Network Learning

In their 2023 paper "FlowMFD: Characterisation", He et al. have illustrated a new approach for classification of Tor traffic using a **M**ount **F**requency **D**irection (MFD) chromatographic feature and spatial–temporal modelling [58]. The methodology employed in this study begins with the extraction of time series features from packets of a Tor traffic capture. The authors observe that Tor traffic exhibits distinct interaction patterns between applications and servers, which can be captured through the amount-frequency-direction of packet transmissions. These interaction patterns are then represented using a **M**ount-**F**requency-**D**irection **C**hromatographic **F**eatures (MFDCF). These features integrate multiple low-dimensional **T**iming **S**ynchronization **F**unctions (TSFs) into a single plane, preserving essential pattern information while enabling the characterization of traffic from different Tor applications.

To capture the spatial-temporal dependencies between MFDCF, the study employs a cascaded model combining a two-dimensional convolutional neural network (2D-CNN) and a bidirectional gated recurrent unit (Bi-GRU), as shown in Figure 27 [58]. The 2D-CNN extracts spatial dependencies by processing MFDCF images, while the Bi-GRU captures temporal dependencies within the traffic data. This hybrid model allows for feature extraction and classification of complex Tor traffic patterns.

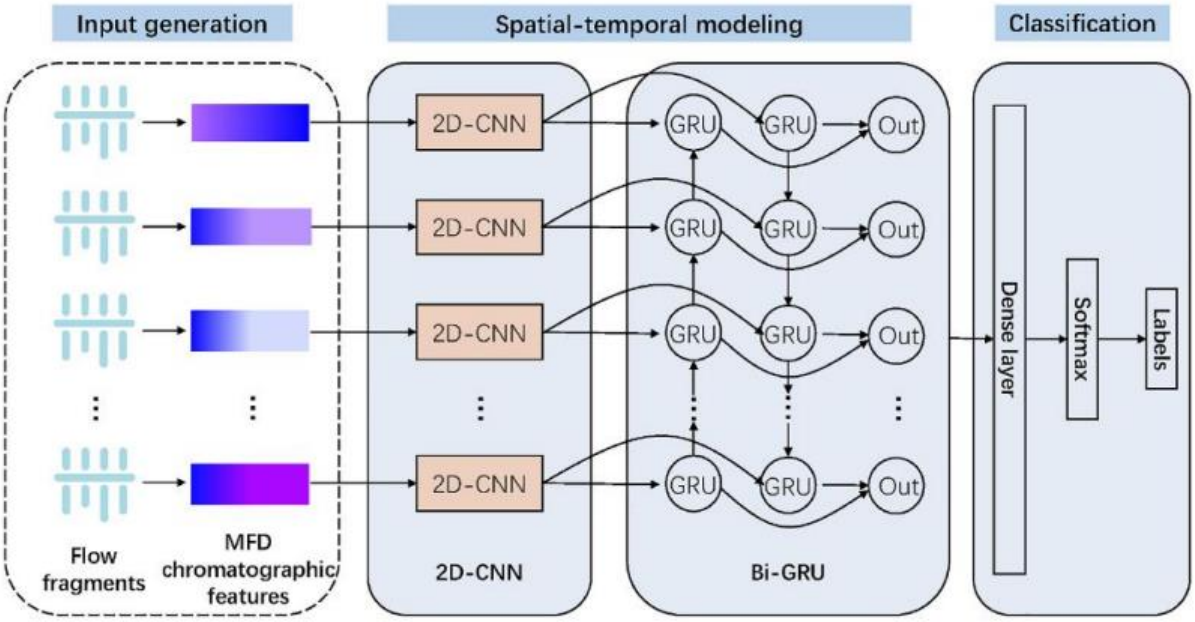


Figure 27: FlowmMFD Framework [58]

The researchers validate FlowMFD using the public ISCXTor2016 dataset and a self-collected dataset comprising dynamic Tor traffic. The results demonstrate that FlowMFD achieves an accuracy of 92.1% on the ISCXTor2016 dataset and 88.3% on the self-collected dataset. The authors compare their performance to traditional machine learning algorithms to highlight the effectiveness of MFDCF and spatial-temporal modelling in enhancing the classification performance of Tor traffic.

The FlowMFD methodology for Tor traffic classification exhibits certain limitations as outlined by the authors. The reliance on specific packet sizes for feature extraction may not generalize well to dynamic or highly variable traffic patterns, potentially limiting applicability in more divergent network environments such as Tor. Additionally, although the method leverages both spatial and temporal dependencies using 2D-CNN and Bi-GRU networks, it assumes relatively uniform connections in traffic, which may not hold in real-world scenarios involving

noisy data. The evaluation is conducted primarily on controlled datasets, such as ISCXTor2016 and self-collected AAT, which may not adequately capture the diversity and complexity of real-world Tor traffic. Furthermore, the model tends to overfit to typical packet sizes, as evidenced by calibration bias and overconfidence in predictions.

5.3 Tor Traffic Classification Using LSTM

5.3.1 DarkDetect LSTM Classifer

I In their 2021 paper entitled "DarkDetect: Darknet Traffic Detection and Categorization Using Modified Convolution-Long Short-Term Memory" [57], Sarwar et al. have presented their findings pertaining to identification and categorization of darknet traffic using deep learning techniques. The study specifically focuses on Convolutional Neural Networks (CNN) integrated with Long Short-Term Memory (LSTM) networks. The dataset used for training and validation purposes is an amalgamation of previously published datasets, enhanced to include various categories of VPN and Tor traffic. The final aggregated dataset comprises 85 features, including labels for traffic types (Tor, Non-Tor, VPN, Non-VPN) and application categories (Audio-Stream, Browsing, Chat, E-mail, P2P, Transfer, Video-Stream, VOIP) [57].

In the preprocessing stages, the proposed system performs several data handling steps, starting with a stage/technique for addressing of missing data, as shown in Figure 28 [57]. Specifically, dataset rows with missing information are removed to avoid any adverse effects on the learning process. The dataset is then balanced using the Synthetic Minority Over-sampling Technique (SMOTE) to ensure that every class is adequately represented, and thereby improve the accuracy and precision of the classifier. The study specifically increases the instances of the Tor class, which initially had fewer samples.

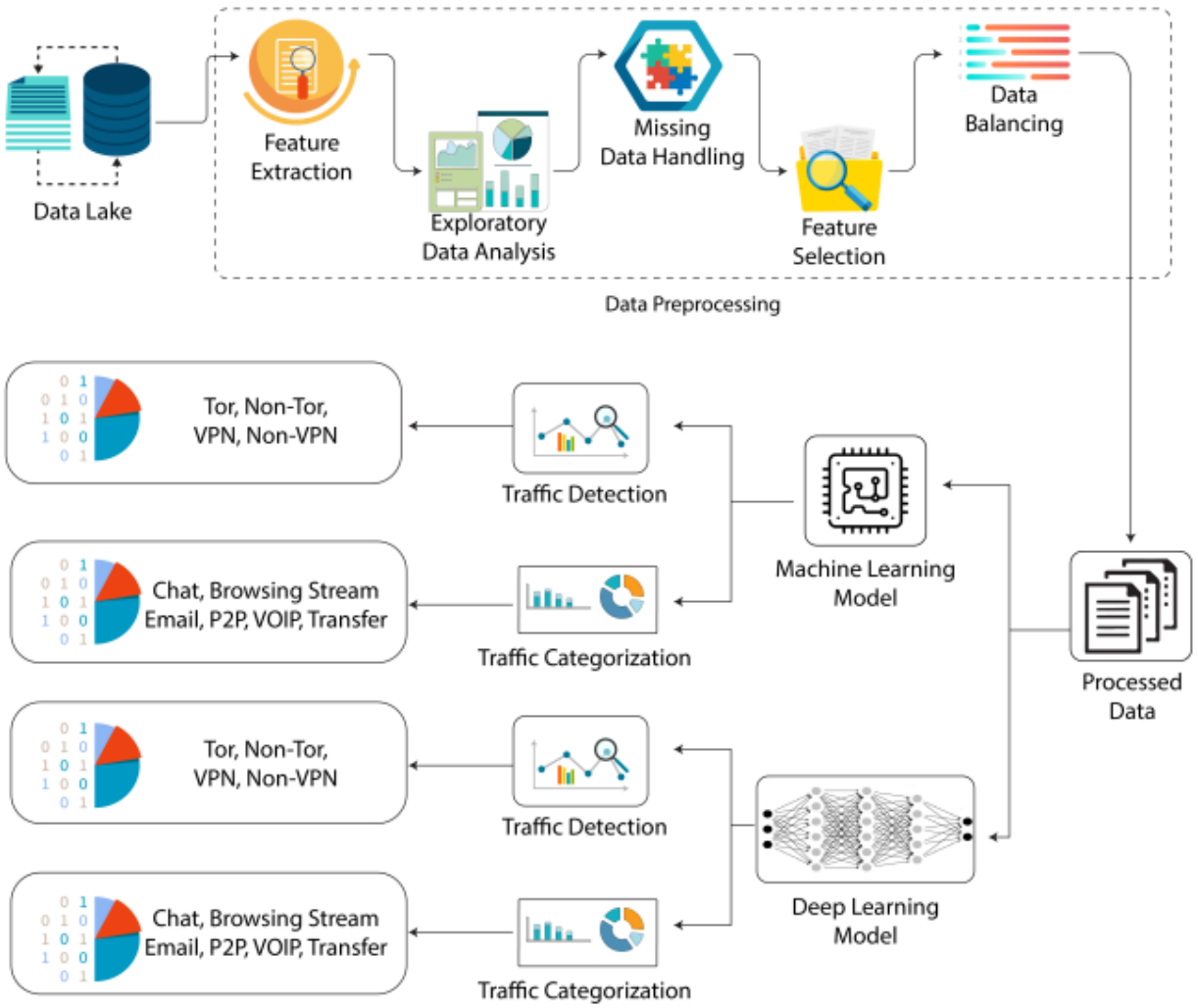


Figure 28: DarkDetect Taxonomy [57]

The authors point out to feature selection as a critical part of their methodology, involving Principal Component Analysis (PCA), Decision Trees (DT), and Extreme Gradient Boosting (XGB). These methods help identify the most relevant features out of the 85 available, so as to ultimately reduce the dataset dimensionality and improving the overall performance of the classification models. PCA is particularly noted for producing orthogonal, uncorrelated features

ranked by their explained variance, while DT and XGB use different criteria to select the optimal features based on their importance.

The core classification task is approached using modified deep learning models: CNN-LSTM and CNN-GRU (Gated Recurrent Unit). These models are designed to handle sequential data, capturing the temporal dependencies in network traffic. The CNN layer extracts features from the input data, while the LSTM or GRU layer handles the sequence prediction. This dual approach allows the models to effectively learn and classify complex patterns in darknet traffic.

The authors compare their performance using various metrics such as precision, recall, F-score, and accuracy. The study finds that the CNN-LSTM model outperforms other models, achieving higher precision, recall, and F-score. The CNN-LSTM model achieves 1%, 4%, and 3% better precision, recall, and F-score respectively compared to the CNN-GRU model on different traffic types.

The authors also perform darknet traffic categorization at a more intricate level, analyzing specific traffic categories like Audio-Stream, Chat, Email, P2P, VOIP, Video-Stream, Transfer, and Browsing. The results show that traditional machine learning models like Random Forest Regressor (RFR), Gradient Boosting (GB), and Decision Tree (DT) achieve significant performance, but the deep learning approach, particularly CNN-LSTM, provides superior accuracy and stability.

The methodology proposed in "DarkDetect" for traffic detection and categorization shows several advancements but also several limitations as outlined by the authors. While the CNN-LSTM model achieves competitive accuracy, its reliance on oversampling techniques like SMOTE for data balancing may introduce overfitting or bias, especially in real-world scenarios with more complex imbalances. The approach also heavily depends on the performance of XGB

for feature selection, yet the interplay between feature selection and classifier performance in different traffic types is not extensively explored. Additionally, the method's evaluation on only one dataset may limit its generalizability to broader darknet traffic contexts.

5.3.2 RNN-LSTM Based Deep Learning Model for Tor Traffic Classification

The work titled "RNN-LSTM Based Deep Learning Model for Tor Traffic Classification" by VishnuPriya et al. provides insight of how deep learning models, particularly those using Recurrent Neural Networks (RNN) and LSTM, can be applied to classify network traffic as either Tor or non-Tor [56]. The methodology employed in this study is centered on using the UNB-CIC Tor and non-Tor dataset (8,044 Tor samples, 59,784 non-Tor samples).

In the preprocessing phase, two feature selection methods are applied: Correlation Based Filter Selection (CFS) and Symmetric Uncertainty (SU), similar to the process outlined in [54]. CFS evaluates the predictive power of an attribute and its redundancy relative to other features, ensuring that only the most relevant attributes are ultimately deployed. SU, on the other hand, measures the uncertainty of a random variable, helping to identify the most informative features. These methods reduce the original feature set to a more manageable subset, which is then used to train the deep learning models.

The core of the study is the implementation of deep learning models, specifically RNN and LSTM architectures. These models are well-suited for handling time-series data, which is crucial for analyzing network traffic that changes over time. The RNN-LSTM model developed in this study consists of multiple layers that process the input data sequentially, capturing temporal dependencies and patterns that are indicative of Tor or non-Tor traffic. The model was trained using the Adam optimizer, which adjusts the learning rate to improve the model's accuracy over time.

The performance of the RNN-LSTM model is compared to traditional machine learning models like DNN. The results show that the RNN-LSTM model achieves e higher accuracy and precision in classifying Tor traffic. Specifically, the RNN-LSTM model achieves an accuracy of 88%, while the DNN model only achieves 86%. The precision and recall metrics also indicate better performance for the RNN-LSTM model, highlighting its effectiveness in distinguishing between Tor and non-Tor traffic.

To further validate the model's performance, the researchers have generated additional traffic samples using the CIC Flowmeter tool in a controlled virtual environment, with the ultimate purpose to balance the original dataset (see Figure 29) [56]. Even on the extended dataset, the RNN-LSTM model continues to outperform the DNN model, achieving an accuracy of 97%.

Additionally, the study explores the robustness of the RNN-LSTM model by comparing it to other traditional machine learning classifiers such as ZeroR, J48, Naive Bayes, Random Forest, and K-Nearest Neighbors (KNN). The results show that while some traditional classifiers perform well, the RNN-LSTM model consistently provides better accuracy and precision, making it a more reliable choice for Tor traffic classification.

The RNN-LSTM-based deep learning model for Tor traffic classification addresses challenges in time-based feature extraction but exhibits limitations. The dataset imbalance, particularly the underrepresentation of Tor samples, poses a challenge despite the use of data augmentation. While the authors achieve improvements in precision and recall, the study uses only two datasets (ISCXTor2016 and a generated dataset), which constrain the generalizability of the findings to broader or more diverse network environments. Additionally, the model’s

heavy computational requirements, exacerbated by the parameter-intensive nature of LSTMs, make it less feasible for deployment in resource-constrained settings.

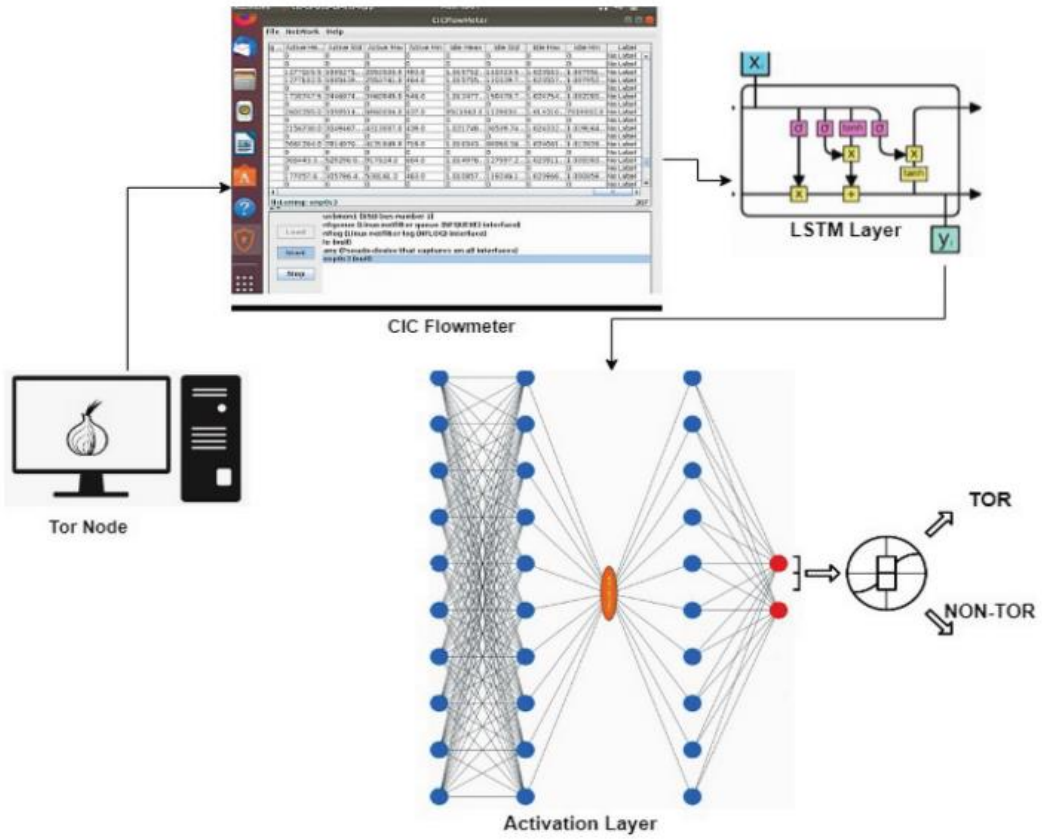


Figure 29: RNN-LSTM Block Diagram [56]

5.4 Non-Tor Traffic Classification Using LSTM

5.4.1 Network Traffic Classification Using a Combined CNN and LSTM Model

In the 2023 paper titled “Network Traffic Classification Model Based on Attention Mechanism and Spatiotemporal Features” [59] Hu et al. introduce a model capable of handling large and complex volumes of network traffic, including encrypted traffic. The model integrates an attention mechanism with spatiotemporal features to enhance the accuracy and reliability of

traffic classification. To validate the model performance, the authors use multiple datasets, including the ISCX VPN-nonVPN, USTC-TFC2016, and a YouTube dataset. These datasets are chosen for their diversity, encompassing both encrypted and unencrypted traffic, as well as normal and abnormal traffic patterns. Dataset preprocessing proposed in [59] consists of three main steps: traffic filtering, image generation, and IDX (Incremental Design Exchange) format conversion. The first (traffic filtering) step involves splitting raw packets into session-level packets and cleaning the data to remove any empty or duplicate files that could affect model training. In the second (image generation) step, the network session flows are converted into 784-byte images, which are then used as inputs for the CNN network. The visual representation of these images allows for clear distinctions between different types of traffic, demonstrating the feasibility of using session flow-generated images for traffic classification. It the final third step, the images are converted into the IDX format, which is commonly used for storing large multidimensional arrays or tensors efficiently. This format ensures compatibility of input data for both the LSTM and CNN networks used in the study, facilitating further analysis and classification tasks.

The core of the model is the integration of LSTM and CNN with an attention mechanism. LSTM networks are well-suited for analysis of network traffic as they can learn long-term dependencies and are effective for handling of temporal correlations. The LSTM network used in this study consists of several long- and short-term memory units, allowing it to save and retrieve values and gradients over several subsequent time steps. The CNN layers extract spatial features from the input data, which are then processed by the LSTM layers to capture the temporal dependencies.

The attention mechanism is incorporated into the classification model to enhance its ability to focus on the most relevant parts of input data. This mechanism dynamically assigns weights to different parts of the input, allowing the model to prioritize more significant features. The combination of CNN, LSTM, and attention mechanism enable the model to effectively capture both spatial and temporal features, resulting in a more accurate and reliable traffic classification.

The evaluation of the proposed model is based on its comparison against traditional machine learning models such as Random Forest, Gradient Boosting, and Decision Tree, as well as other deep learning models. The results demonstrate that the proposed model outperforms existing methods in terms of accuracy, precision, recall, and F1-score, including classification accuracy of different types of encrypted vs. unencrypted traffic.

The proposed spatiotemporal network traffic classification model leveraging LSTM, CNN, and Squeeze-and-Excitation (SE) mechanisms demonstrate increased accuracy but has several disadvantages. The methodology relies heavily on predefined datasets, such as ISCX VPN-nonVPN and USTC-TFC2016, which may not fully reflect the diversity of real-world encrypted network traffic. Additionally, while the model employs the SE mechanism to refine features, the computational overhead introduced by such attention mechanisms is not adequately addressed, especially for deployment in resource-constrained environments. The preprocessing steps, which involve converting traffic data into 28x28 images, can result in the loss of nuanced temporal information critical for certain traffic patterns. The experiments focus on balanced datasets, however the model's performance under imbalanced noisy traffic conditions is not thoroughly evaluated.

5.4.2 A Look Behind the Curtain: Traffic Classification

In the 2020 paper titled "A Look Behind the Curtain: Traffic Classification in an Increasingly Encrypted Web" Akbari et al. present a novel approach to traffic classification of encrypted web traffic corresponding to protocols like HTTP/2 and QUIC [55]. The authors' methodology begins by highlighting the inadequacies of current deep learning models, which often rely on simplistic logic derived from certain header fields of TLS handshake packets. The authors claim this reliance not only limits the robustness of the models to future versions of encrypted protocols but also overlooks crucial domain-specific considerations. To address these issues, the authors design a novel feature engineering approach that generalizes well across encrypted web protocols and developed a neural network architecture that leverages Stacked LSTM layers and CNN (see Figure 30).

The feature engineering process involves extracting three types of features: 1) raw bytes from the TLS handshake packets, 2) flow time-series data (including packet sizes, directions, and inter-arrival times), and 3) standard flow statistics (such as mean, standard deviation, and median of packet sizes, TCP flag counts, and flow durations). This comprehensive feature set is crafted to capture the intrinsic characteristics of encrypted traffic without relying on easily identifiable but potentially misleading header fields.

For the neural network architecture, the authors implemented a three-part model that processes these three types of features separately before concatenating their outputs and passing them through additional fully-connected layers for the final classification. The raw handshake bytes are fed to a one-dimensional CNN, the flow time-series is processed by a stacked LSTM network, and the flow statistics are handled by the fully-connected network, see Figure 30.

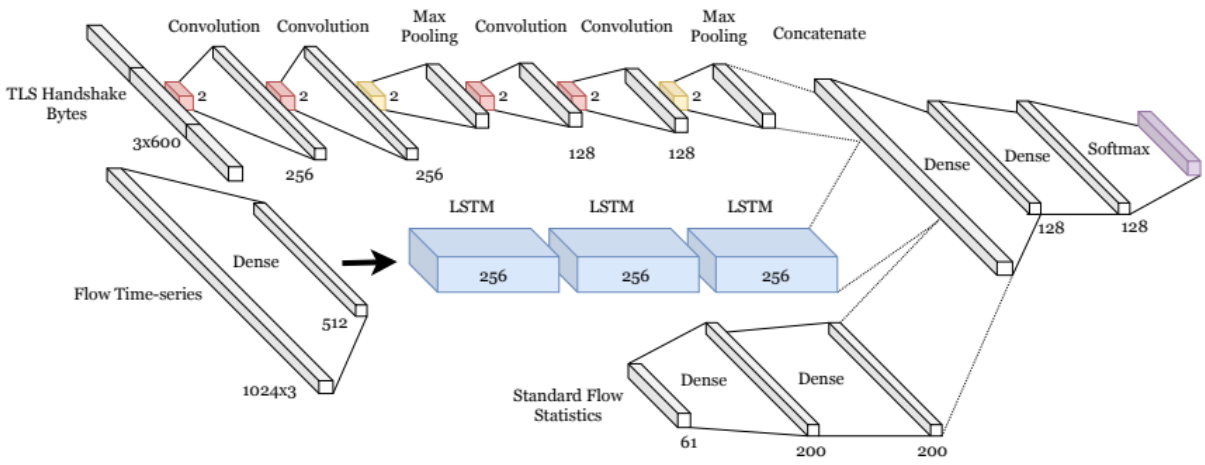


Figure 30: Three-Part Neural Network Architecture [55]

The training strategy involves using an Adam optimizer with manual reduction of the learning rate as training progresses, which helps balance the trade-off between parameter search granularity and training speed. The dataset used for evaluation comes from a major ISP and Mobile Network Operator, and includes both service-level and application-level classifications. The results show that their model achieves a 95% accuracy in service classification with fewer false classifications compared to other methods surveyed by the authors. CNN and Stacked LSTM layers that is used to extracted features for distinguishing between different traffic classes. Real-world mobile traffic dataset from an ISP is used to showcase their methodology performance in service classification of encrypted web traffic. The authors achieve an average accuracy of over 95% for classification exclusively over HTTPS.

The proposed methodology for traffic classification in an encrypted web environment leverages feature engineering and hybrid CNN-LSTM architecture but presents several disadvantages. The reliance on pre-defined datasets, such as the Orange'20 dataset, raises concerns about generalizability to other real-world traffic patterns, especially in diverse or highly

dynamic network conditions. Additionally, the computational expense associated with the proposed model, limits its scalability in large-scale or resource-constrained environments. Furthermore, the use of manual preprocessing techniques, such as flow time-series extraction, could introduce bias or limit the applicability of the model in fully automated or real-time traffic analysis systems.

5.4.3 Deep Learning for Encrypted Traffic Classification in the Face of Data Drift

In 2023 Malekghaini et al. publish a paper "Deep Learning for Encrypted Traffic Classification in the Face of Data Drift: An Empirical Study" [60], in which they outline the challenges and solutions associated with classifying encrypted network traffic, particularly under the conditions of data drift. Data drift refers to a situation when the distribution of input data changes over time, which can degrade the performance of pre-trained models as they encounter new and evolving traffic patterns. The authors use several real-world datasets, collected from a major ISP's mobile network over two years, to observe the extent of data drift and its impact on traffic classification models in practical settings. The datasets include traffic encrypted using TLS and Quick UDP Internet Connections (QUIC) protocols.

The authors used two encrypted traffic classifiers as the basis for experiments: the University of Waterloo's tripartite model and the UCDavis CNN model. The tripartite model consists of three parts: convolutional neural networks (CNNs) operating on TLS header bytes, long short-term memory (LSTM) layers processing traffic-flow time-series data, and dense layers analyzing statistical flow data. The UCDavis CNN model, on the other hand, focuses on early classification using the first few packets of a flow.

To evaluate the effect of data drift, the models are trained on one and then tested on another dataset collected at a different time. This approach is intended to show how changes in

traffic patterns over time can impact the model performance. The study reveals that both models experience significant performance degradation when applied to newer datasets, with the degree of degradation correlating with the gap between the collection time of training and testing datasets.

One interesting finding of this work is that the tripartite model's performance drop is more pronounced in its TLS header component compared to the flow time-series component. This suggests that time-dependent traffic features are more robust to data drift than TLS header byte features, which can change more significantly over time. The researchers also observe that some service classes, such as streaming and download, are more affected by data drift than others, leading to higher misclassification rates.

To mitigate the impact of data drift, the paper proposes several architectural adaptations and best practices. For instance, reducing the dropout rate in the LSTM layers and simplifying the model architecture by using bidirectional LSTMs (BLSTMs) or one-dimensional convolutional layers (CONV1D) for smaller datasets to help improve model robustness. The study also emphasizes the importance of updating model architectures to accommodate changes in dataset size and distribution.

The study further examines the role of application-layer protocol negotiation (ALPN) fields in TLS traffic. It is noted that the performance of the tripartite model is significantly affected by changes in the ALPN field values over time. For example, the decline in the use of the SPDY protocol and the rise in HTTP/2 adoption appear to have contributed to the observed data drift. To determine its actual impact on classification accuracy, the researchers have conducted additional experiments by obfuscating the ALPN field, revealing that clear ALPN

values slightly enhance model performance but are not the primary cause of performance degradation.

The methodology proposed for encrypted traffic classification in the face of data drift demonstrates several limitations. While the authors address the decay in model performance over time, its reliance on specific datasets from a major ISP's mobile network may limit generalizability across diverse network environments. The methodology also assumes a relatively consistent distribution of labeled flows, which is not reflective of the real-world imbalance and variability in traffic data, Tor in particular. Furthermore, the computational complexity of adapting the model architecture to smaller datasets, while effective, is not evaluated for deployment feasibility in real-time or resource-constrained environments.

5.4.4 Network Anomaly Detection Using LSTM Based Autoencoder

Lastly, Elsayed et al. have published a paper entitled "Network Anomaly Detection Using LSTM Based Autoencoder" [61], which examines the use of a combined LSTM autoencoder and One-Class Support Vector Machine (OC-SVM) for purposes of anomaly detection in network traffic. The central objective of the study is to address the challenge of identifying anomalies in network traffic, which is of particular importance for security of Software-Defined Networks (SDN). The methodology employed in this paper involves several key steps. First, the study uses the InSDN dataset, which contains diverse network traffic data, including both normal and malicious traffic. The dataset is preprocessed to convert it into a suitable format for machine learning models.

The core of the proposed approach is an LSTM-autoencoder model, designed to learn and reconstruct the normal patterns of network traffic. The autoencoder compresses the input data into a lower-dimensional latent space and then reconstructs it. The reconstruction error is used as

an indicator of anomalies, with higher errors suggesting anomalous traffic. This approach leverages the temporal correlations in network traffic data, making LSTM a suitable choice due to its capability to model sequences effectively.

To enhance the anomaly detection capabilities, the authors integrate the LSTM-autoencoder with an OC-SVM, see Figure 31 [61]. The OC-SVM is trained on the compressed representations obtained from the autoencoder, learning the boundary of normal data and identifying points that fall outside this boundary as anomalies. This hybrid approach addresses the limitations of using OC-SVM alone, particularly its inefficiency in handling high-dimensional data directly.

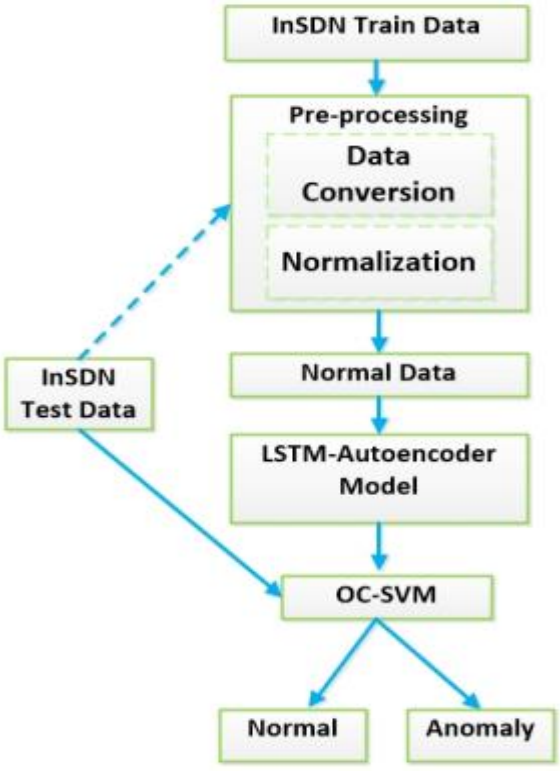


Figure 31: InSDN LSTM OC-SVM Flow Chart [61]

The evaluation of the proposed model involves training it on normal traffic data and then testing its performance on a dataset containing both normal and anomalous traffic. The study measures performance using precision, recall, F1-score, and accuracy metrics. The results indicate that the hybrid LSTM-autoencoder-OC-SVM model achieves high detection rates with significantly reduced false positives compared to traditional methods.

Additionally, the authors discuss the computational efficiency of the proposed model, emphasizing its suitability for real-time intrusion detection in SDN environments. The hybrid model not only improves detection accuracy but also offers a more practical solution for real-world operational deployment due to its lower computational overhead.

5.5 Conclusion & Chosen Technique

Out of the techniques surveyed in this chapter, we believe the one outlined in [61] is best suited for the purposes of our Automated CSOW Watermark Detector (see Figure in Chapter 4). Although several other papers utilize similar LSTM-based solutions to achieve traffic detection and/or classification, and some even deal specifically with Tor network traffic [56, 57], the approach outlined in [61] has most overlap with our intended application and goals. Namely, the main objective of the LSTM solution from [61] is to 'learn' a single type/class of traffic (which is considered to be a "normal" traffic category), and then label any other/different type of traffic as an "anomaly". This is very similar to what we are aiming our ML model to accomplish, which is: 1) learn a very general profile of CSOW watermark (using a larger training dataset of collected and possibly noisy watermarks), and 2) utilize the trained model to categorize any new traffic instance as "contains watermark" if it is deemed sufficiently close/similar to the established model/profile, or "does not contain watermark" otherwise. It is also important to point out that

the authors of [61] claim that their hybrid model is ideal for real-world operational deployment due to its lower computational complexity compared to other models.

Chapter 6

CSOW User De-Anonymization Deploying LSTM Autoencoder With OC-SVM Classifier

6.1 Introduction

In Chapter 4 we have discussed a potential real-world deployment and some inherent challenges of our originally proposed CSOW approach to User De-Anonymization, shown in Figure 24. We refer to this entire application frameworks/system as **CSOW-UDA** system. In this section, we build off the initial CSOW-UDA system by introducing an LSTM component, and ultimately arrive at the **LSTM-Based CSOW-UDA** system shown in Figure 25. We also present and discuss some illustrative experimental results obtained using our novel LSTM-Based CSOW-UDA, which confirms the system's highly effective real-world performance.

6.2 Motivation for Using LSTM Autoencoder

For the reasons explained in Chapter 5, LSTM was selected as the main ML component for our CSOW-UDA system in Figure 25, with the purpose to achieve the following: a) build/learn a robust watermark profile, and 2) use the built profile to reliably detect the presence (or absence) of a watermark in a selected traffic capture. Some of the reasons justifying the selection of LSTM as our ML of choice included:

1) LSTM networks are known for their ability to effectively handle long-term dependencies [62]. Namely, in time series data, past information is critical for making accurate predictions, and LSTM's architecture is designed to remember and utilize information from previous time steps over long sequences. This capability is essential for our watermark detection, as the sequence of packet volumes needs to be analyzed over extended periods to reliably identify patterns indicative of a watermark

2) LSTM networks address and mitigate the vanishing gradient problem, which is a common issue in traditional RNNs [62]. In standard RNNs, gradients can become exceedingly small during backpropagation through time, leading to poor learning of long-range dependencies, hence the name ‘vanishing gradient’. LSTM's unique cell structure with gating mechanisms (input, output, and forget gates) helps maintain more constant error gradients, thereby supporting more effective learning over long sequences. This is crucial for watermarks that spawn over longer sequences in order to induce more robust traffic patterns.

3) LSTM networks are adept at learning complex patterns and trends in sequential data. There may exist watermarking scenarios which involve detecting subtle and/or potentially intricate variations in packet volumes in order to accurately signify the

presence of a watermark. For example, when a user is browsing the webpage intended to generate the watermark while also having a secondary application or webpage generating traffic. LSTM networks are particularly capable of modelling non-linear and sophisticated relationships within the training data, therefore being better suited for capturing these cases versus standard RNNs.

4) LSTMs offer flexibility with irregular time steps. Real-world data often does not follow perfectly regular intervals, and the ability of LSTMs to handle such irregularities without significant loss of performance makes them ideal for our application. As discussed in previous chapters, irregularity of data flow through the Tor network (to obfuscate traffic correlation) is the key contributor to the suboptimal performance in many traditional watermarking approaches. This flexibility ensures that even if packet arrival times vary, the model can still accurately detect the watermark.

6.2 LSTM-Based Client-Side User De-Anonymization Overview

In the previous chapters, we have discussed a potential real-world application scenario and some of the key assumptions for our initially proposed **CSOW-UDA** system. In the case of **LSTM-Based CS-UDA** much of the initial assumptions remain the same. Particularly, all elements involved in the generation of traffic watermark remain unchanged. This includes the webpage and the methodology used to generate the traffic watermark from the victim user's machine. Likewise, all previous assumptions pertaining to the **ULEA** with the ability to lure the Malicious Tor User (**MTU**) into visiting the webpage as well as the ability to monitor their traffic remain the same. The type of traffic being observed also has not changed, with the traffic

remaining encrypted and the attacker only being able to observe the time and number of packets arriving, in order to prevent unintended privacy breaches.

A depiction of our updated LSTM-Based CSOW-UDA approach can be seen in Figure 32. The key difference between Figures 4 and 30 is (only) the addition of the LSTM Autoencoder model which replaces the previous 'naïve' watermark detection procedure. Also note that in the real-world deployment of the system in Figure 32, the LSTM element is trained prior to the luring of the MTU to visit the CSOW webpage, at which point the (trained) LSTM element is used as a watermark detector.

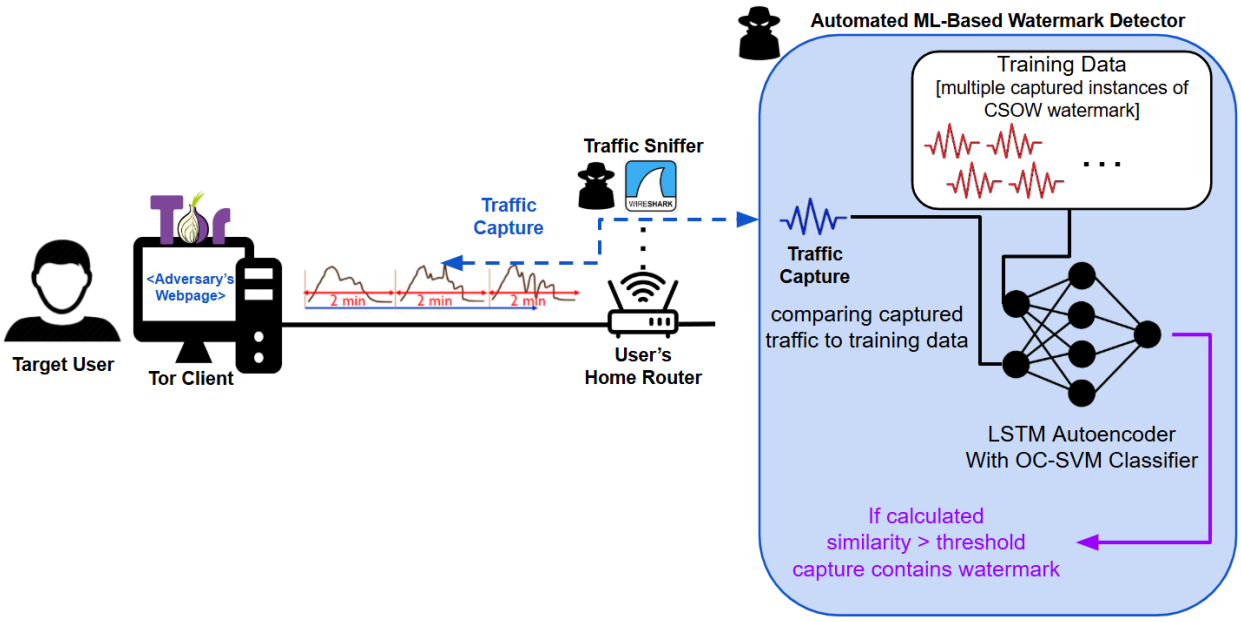


Figure 32: Depiction of LSTM-Based CS-UDA in a Real-World Environment

6.3 LSTM-Based CSOW-UDA: Training/Testing Data Acquisition and Conditions

Given the encrypted nature of Tor traffic - and thus our acquired traffic captures - our model is effectively trained on traffic-capture instances that consist only of arrival times of

observed packets and the respective volume of data/bytes in these packets. There are two important categories of data to consider for our LSTM-Based CSOW-UDA: training and testing. In our model, training data consists of only "normal" traffic instances collected during the retrieval of the watermark-generating CSOW webpage (i.e., traffic instances that contain the CSOW watermark in them – though, possibly corrupted with noise). Test data, on the other hand, contains both normal and anomaly traffic instances to evaluate the ability of our model to accurately detect the presence of the watermark.

To generate our actual training and testing data, we deployed two different methodologies:

1) **Data Collection – Methodology 1.** The first methodology is visualized in Figure 15, and it assumed us generating the traffic data on/from our own personal machine. To accomplish this, we created a python script that would launch Tor browser and then request (i.e., issue the rendering of) the CSOW webpage while simultaneously running a Wireshark (i.e., collecting a traffic capture) for 35 seconds, after which the browser would completely shut down and restart for the next capture. (The python script is able to take parameters such as the number of iterations, destination server, and the duration of the capture.) For our training data we set the number of iterations to 1000 and the destination server to be the one hosting our CSOW webpage. Overall, the collection process took approximately 38 hours which spanned over the course of about a week. Majority of the time spent collecting the data was not from the capture duration but rather from the restart and bootup time of the Tor browser. As for the collecting of 'normal' test data, the process remained largely unchanged, except for the iterations parameter (i.e., number of page retrievals) which was reduced to 200. This collection process took approximately 8 hours

and was done over the course of two days. To generate 'anomaly' test data, the destination server parameter was changed 10 times (i.e., to retrieve 10 'anomaly' pages[2] different from the CSOW page), and included 10 iterations of each of these "anomaly" webpages, for a total of 100 iterations - which took approximately 4 hours.

Now, we do recognize that in the context of CSOW deanonymization attack, this data collection methodology is somewhat unrealistic, as it would practically imply that the attacker (i.e., adversary owning/controlling the LSTM-Based CSOW-UDA system) has access to the victim-user's machine and is able initiate the collection of LSTM's training and testing data from that machine. To address this issue, we have also deployed another more realistic data collection methodology, discussed next.

2) **Data Collection – Methodology 2.** In this approach, visualized in Figure 33, the training data was the same as the one collected in Methodology 1. However, to collect testing data, we utilized a completely different machine (connected to the same router as the machine used for collection of training data). In total, from this second machine, we gathered 200 instances of 'normal' test data which took approximately 11 hours and was done over two days. For 'anomaly' test data we gathered 100 instances which took approximately 5.5 hours and were done in one day. The main purpose of this second data collection method is to evaluate the performance of our LSTM-Based CSOW-UDA in a more realistic scenario – when the adversary does NOT have direct access to the victim-user's machine, but rather needs to train the LSTM model on data (i.e., watermark traffic captures) generated from another machine.

[2] The 10 anomaly webpages consist of varying forms of media, links to which can be found in the references [63 – 72].

The total time spent to generated and gather data for all methodologies is illustrated in figure 34.

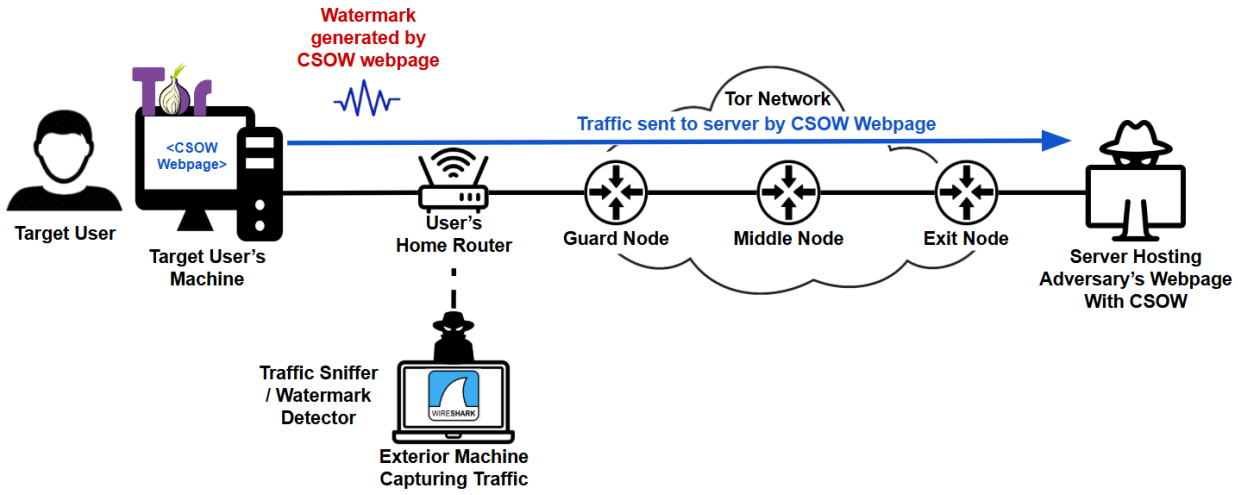


Figure 33: External Data Collection Topology

Total Training/Testing Data Acquisition		
Training Data		
Instances		**Time-Taken**
Normal	1000	38 Hours
Test Data – Method 1		
Instances		**Time-Taken**
Normal	200	8 Hours
Anomaly	100	4 Hours
Test Data – Method 2		
Instances		**Time-Taken**
Normal	200	11 Hours
Anomaly	100	5.5 Hours

Figure 34: Total Training/Testing Data Acquisition Table

6.4 LSTM-Based CSOW-UDA: Components, Features and Setup

Our implementation of the LSTM Autoencoder closely follows the research of [61], using mostly the same parameters and procedures – including various stages of data handling and processing. To begin, we start off by feeding training data to the LSTM Autoencoder in order to learn/build the "normal" traffic (watermark) pattern. Subsequently, the trained LSTM model is presented both normal and anomaly test data. This allows us to evaluate how well the model is able to both detect/recognize the CSOW watermark in a randomly presented traffic instance, as well as to differentiate the learned CSOW watermark profile from other webpages' traffic.

Different components of our LSTM model can be seen in Figure 35. Note that this figure depicts the model in the training stage. The only difference compared to the testing stage is that the OC-SVM will also begin classifying inputs as anomalies.

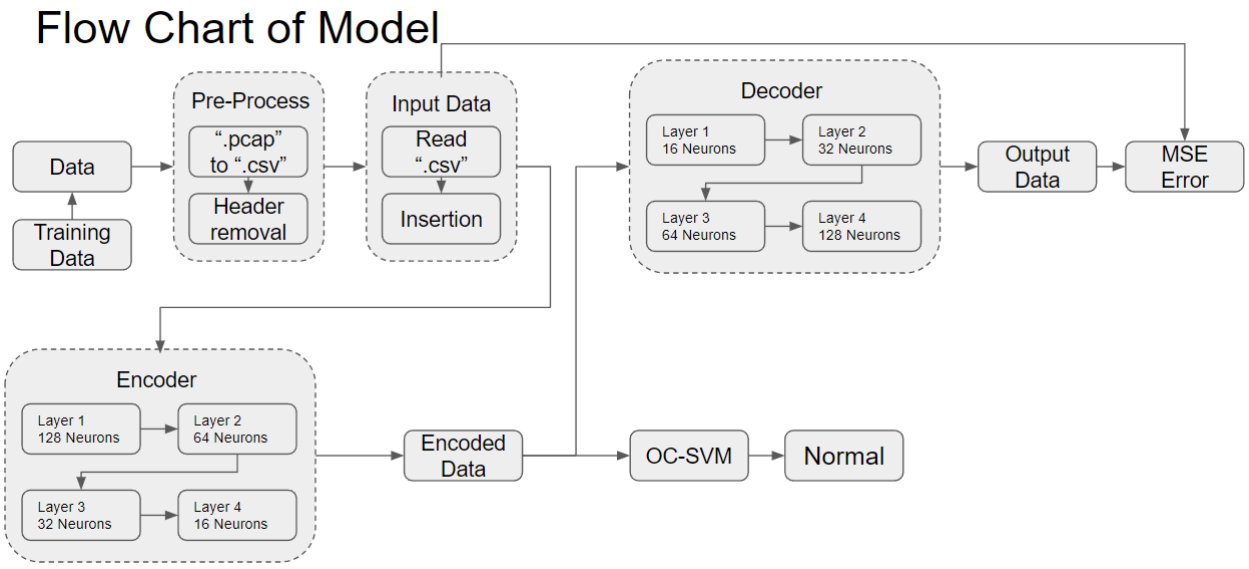


Figure 35: LSTM Autoencoder OC-SVM Flow Chart

Our model from Figure 35 is implemented in Python, and we have utilized several libraries that provide robust tools and frameworks to build and train our model efficiently. These libraries include numpy and pandas for file conversion, management, and reading. Sklearn is used for OC-SVM implementation and preprocessing scalers as well as metrics reports. Keras' libraries are used to build the LSTM autoencoder itself. Lastly, TensorFlow is incorporated to generate representations of the data.

The first operational component in our model is the pre-processing stage that involves converting pcap files (which are packet capture files that contain network traffic data from Wireshark) into csv format. Csv files are easier to handle and manipulate using various data processing tools. Proper formatting arrangements are made to ensure that the data is structured in a way that is suitable for our intended purpose within the model. This step includes cleaning the data, removing headers, handling missing values, and normalizing the features to ensure consistency and accuracy.

After the data has been pre-processed (now forming properly formatted 'input data'), it is passed to the encoder which consists of four layers with decreasing numbers of neurons (128, 64, 32, and 16 neurons) from layer 1 to layer 4, respectively. The encoder compresses the input data into a lower-dimensional representation, capturing the most critical features while discarding redundant information. This process of dimensionality reduction helps in focusing on the essential aspects of the data that are most relevant for anomaly detection while also reducing the total training time needed for the OC-SVM given the reduction in dimensions.

The encoder will then output the encoded data which will be passed to both the decoder and the OC-SVM. The purpose of the OC-SVM in our model is to serve as the classifier that

distinguishes between normal data (i.e. watermark) and anomaly data (i.e. other webpages). After training, the OC-SVM performance is tested on both normal and anomaly data.

The other segment to which the encoded data is sent is the LSTM decoder which reconstructs the encoded data. The decoder is structured in reverse order of the encoder, with four layers containing 16, 32, 64, and 128 neurons, respectively. Its role is to reconstruct the input data from the encoded representation. The output data produced by the decoder is then compared to the original input data to calculate the Mean Squared Error (MSE). This comparison helps in validating the accuracy of the detection process by ensuring that the reconstructed data closely matches the original input data, indicating that the encoding and decoding processes have effectively captured the essential features of the training data.

Additional parameters/features of our LSTM model are as follows: The model uses Adam optimizer [83], as it outperforms other similar algorithms. The selected loss function is MSE, which allows us to minimize reconstruction error. The activation function employed is Tanh [84], a well-suited function for time series data. The number of learning epochs (iterations) is set to 100, and the batch size to 32, ensuring a thorough training process without excessively long training times. However, if there is no gain in performance after several epochs the model is designed to cease further training.

For the OC-SVM component, the following parameters have been selected to optimize its performance. The gamma value is set to 0.01 [85], controlling the influence of individual training examples. The parameter nu is set to 0.4 [85], defining an upper bound on the fraction of training errors and a lower bound of the fraction of support vectors. Finally, the Radial Basis Function (RBF) kernel is chosen for its effectiveness in handling anomaly detection.

6.5 LSTM-Based CSOW-UDA: Experimental Results

In our experimental results, we present some representative samples of our obtained experimental results, which include: training and validation loss per epoch as well as confusion matrix for various categories of data. First, our training and validation loss is shown in Figure 36. Observing this graph, we can see there is no underfitting as both training and validation loss descent smoothly, until they stabilize at a specific point - which is the exact characteristic of a good fit. Likewise, there is no overfitting as the validation loss remains very close to the training loss.

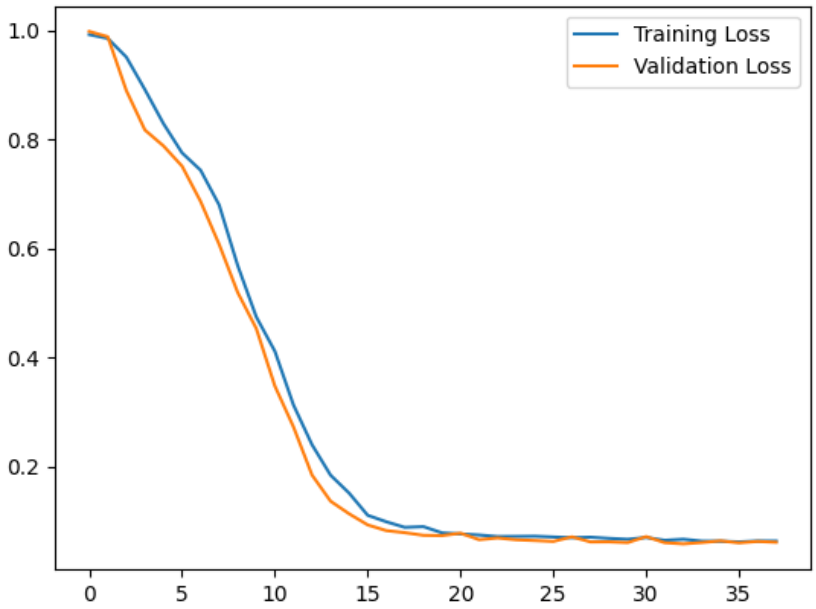
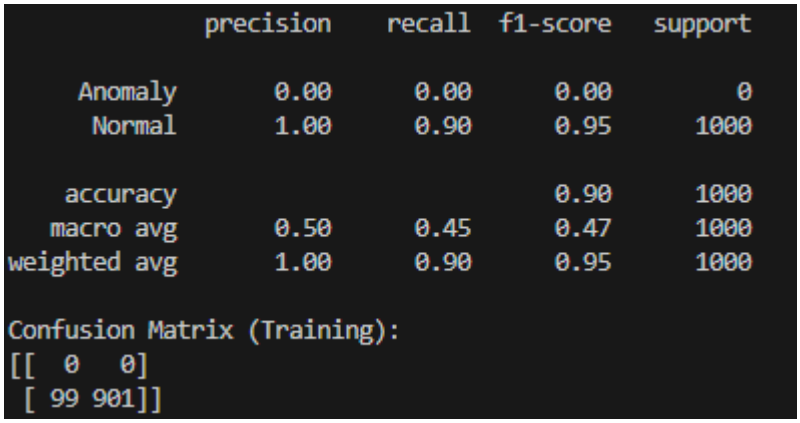


Figure 36: Training and Validation Loss Graph

There are five important confusion matrices we wish to discuss, each pertaining to a different aspect of the results we collected using two previously explained data collection methodologies. Beginning with the training data collected using Methodology 1, its respective

confusion matrix is shown in Figure 37. From this matrix, we can see that the fully trained model was able to accurately identify 90% of instances. As for 10% of misclassified data, it is likely that these were watermarks experiencing above average traffic noise. This percentage could be potentially remedied by increasing our MSE threshold. Though this also would have potential to increase the chances of false positives and would require further testing.



	precision	recall	f1-score	support
Anomaly	0.00	0.00	0.00	0
Normal	1.00	0.90	0.95	1000
accuracy			0.90	1000
macro avg	0.50	0.45	0.47	1000
weighted avg	1.00	0.90	0.95	1000

Confusion Matrix (Training):
[[0 0]
[99 901]]

Figure 37: Confusion Matrix for Training Data Collected Using Methodology 1

The confusion matrix for 'normal' test data collected with Methodology 1 (see Figure 38) shows that we were able to achieve a very high accuracy of approximately 93% for 200 test samples. From the practical point of view, this implies that our proposed LSTM CSOW-UDA system is able to very accurately depict when a user visits our watermark webpage. Or, in other words, the system is able to accurately de-anonymize the MTU in 93% of cases.

Perhaps even more important are our results for the anomaly portion of the test data collected with Methodology 1, shown in Figure 39. Namely, we can see from Figure 39 that for the 100 samples of traffic instances corresponding to non-CSOW webpages, not one of them was mistaken for our watermark - resulting in 100% accuracy. From the practical point of view, this

confirms excellent performance of our CSOW watermark as well as LSTM-Based CSOW-UDA system that guarantees 0% false positives, which is especially important in the cases of real-world user de-anonymization by an ULEA. Namely, a false positive greater than 0% would suggest the possibility of an innocent user getting falsely accused of being the MTU.

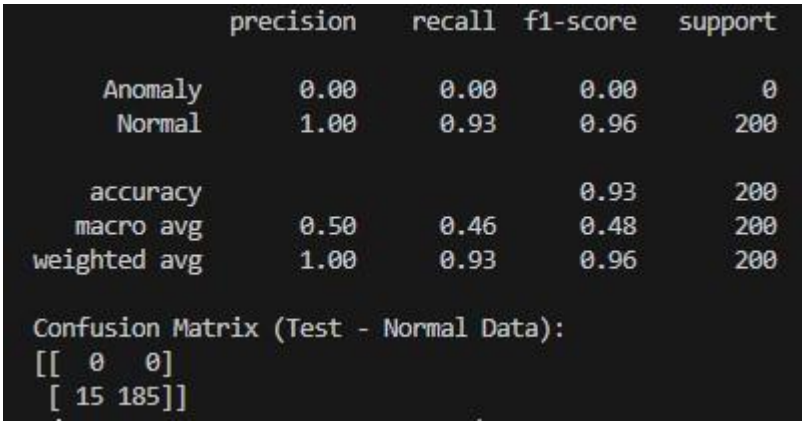


Figure 38: Confusion Matrix for 'Normal' Testing Data Collected Using Methodology 1

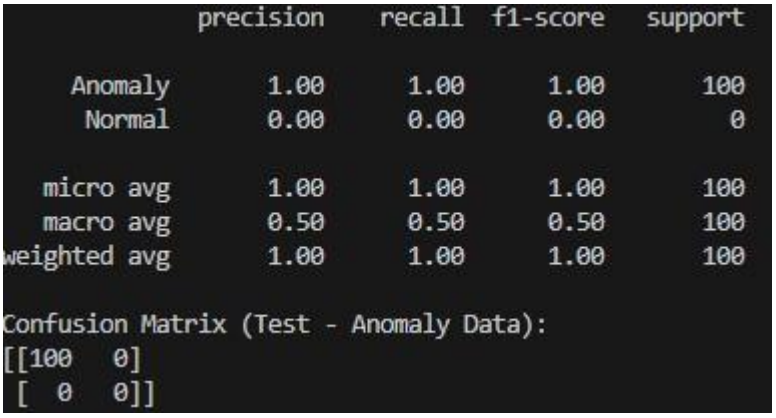


Figure 39: Confusion Matrix for 'Anomaly' Testing Data Collected Using Methodology 1

In the confusion matrix for 'normal' test data collected with Methodology 2 (Figure 40), we see a drop in accuracy down to 85% from 93%. This drop is not entirely surprising, given that Methodology 2 means that the model is being validated on data generated (i.e., the CSOW page being retrieved) by a completely different machine than the data used for the model's training. Regardless, 85% accuracy seems still reasonably satisfactory, from the practical point of view.

Figure 41 shows the confusion matrix for ‘anomaly’ test data collected with Methodology 2. This matrix once again shows no false positives, which further proves: a) that our CSOW approach is highly effective in generating a watermark unique enough to not be mistaken for any other traffic pattern/instance, and b) our LSTM-based detector is robust enough even when trained on data generated by a machine other than the machine of the target user.

```
              precision    recall  f1-score   support

     Anomaly       0.00      0.00      0.00         0
      Normal       1.00      0.85      0.92       200

    accuracy                           0.85       200
   macro avg       0.50      0.42      0.46       200
weighted avg       1.00      0.85      0.92       200

Confusion Matrix (Test - Normal Data):
[[  0   0]
 [ 30 170]]
```

Figure 40: Confusion Matrix for ‘Normal’ Testing Data Collected Using Methodology 2

```
              precision    recall  f1-score   support

     Anomaly       1.00      1.00      1.00       100
      Normal       0.00      0.00      0.00         0

   micro avg       1.00      1.00      1.00       100
   macro avg       0.50      0.50      0.50       100
weighted avg       1.00      1.00      1.00       100

Confusion Matrix (Test - Anomaly Data):
[[100   0]
 [  0   0]]
```

Figure 41: Confusion Matrix for ‘Anomaly’ Testing Data Collected Using Methodology 2

6.6 LSTM-Based CSOW-UDA: Conclusion

In this chapter, we have detailed the integration of an LSTM-based autoencoder with an OC-SVM classifier with our original CSOW UDA framework. In particular, we first outlined the

motivation for employing LSTM models, given their ability to handle sequential data with temporal dependencies and noise resilience. Subsequently, we have described the design and implementation of the LSTM-based CSOW UDA system, while highlighting its key components, feature selection, and experimental setup. The experimental results obtained demonstrate the system's effectiveness in accurately detecting client-side watermarks with no false positives, ultimately proving its robustness and usefulness in real-world applications involving de-anonymization of Tor users.

6.7 Final Remarks: Impact of Noise

As mentioned previously, our samples are under the assumption the user is only accessing one webpage at a time (no exterior noise outside of standard network traffic), however, it is important to clarify what kind of noise is in fact problematic to our approach. For network traffic to impact our watermark it has to stem from the same machine accessing the CSOW webpage (meaning if there are multiple devices in use under the same router their noise can be filter out from the p-cap by observing the IP addresses). Furthermore, the noise being generated on the machine as to be generating specifically from the same Tor browser (i.e., traffic is going to the same Guard Node) as traffic heading to any other IP address can also be filtered out and does not constitute as problematic noise and will not impact the performance of our technique.

Chapter 7

Conclusion

In this thesis we showcase the limitations of the existing Tor user deanonymization attacks that utilize SSOW watermarks. As an alternative, we have proposed reverse-flow CSOW watermarks, and demonstrate their superior performance and much better resistance to traffic noise. Furthermore, we have proposed the use of LSTM deep learning algorithm for automated and more robust watermark detection, and we have demonstrated its excellent performance through real world experimentation.

As for our future work, we plan to focus on further improvement of LSTM-Based CSOW-UDA system to better deal with: a) exterior (Tor network induced) noise, and b) noise in the testing data caused by simultaneous network activity of the machine generating the test data (e.g., the target user is streaming traffic to their machine at the same time when requesting the CSOW page). Regarding objective b),

in our research we have found that the noise in data caused by simultaneous network activity of the machine generating the test traffic may have a significant impact on the system

performance. However, we have also observed that most streaming applications (producing video and/or audio streams, such as YouTube or TikTok) or social media sites (like Instagram and Facebook) ultimately generate recognizable traffic patterns. This claim is further supported by some of the LSTM research papers we have surveyed, in which their proposed LSTM models are able to accurately identify different categories of network traffic (browser, video, audio, etc.).

Lastly, we would like to discuss some of our ethical considerations in the context of the privacy of ordinary/innocent users. Our work points to the weaknesses existing in Tor and although our goal is to De-Anonymize Malicious Tor Users (MTU) our work also brings awareness to the regular users about the possibility of De-Anonymization. Furthermore, there are possible ways for users to go about defending themselves against our approach. For instance, users can ultimately negate this methodology by disabling JavaScript or generating additional traffic specifically through Tor to alter the watermark to avoid detection.

References

[1] Fazlioglu, Müge. “Most Consumers Want Data Privacy and Will Act to Defend It.” *International Association of Privacy Professionals*, 22 Mar. 2023, iapp.org/news/a/most-consumers-want-data-privacy-and-will-act-to-defend-it.

[2] Mercado, Andrea. “VPN Usage Explodes: Must-Know VPN Statistics for 2024.” *Skillademia*, 20 Mar. 2024, www.skillademia.com/statistics/vpn-usage-explodes-must-know-vpn-statistics-for-2024/.

[3] “The Tor Project: Privacy & Freedom Online.” *Tor Project | Anonymity Online*, www.torproject.org/. [Accessed Jul. 15, 2023]

[4] “Welcome to Tor Metrics!” *Tor Metrics*, metrics.torproject.org/. [Accessed Jul. 15, 2023]

[5] S. Nazah et al., ”Evolution of Dark Web Threat Analysis and Detection: A Systematic Approach”, *IEEE Access*, Volume 8, 2020.

[6] L. Basyoni et al., ”Traffic Analysis Attacks on Tor: A Survey”, *IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, Doha, Qatar, February 2020.

[7] I. Karunanayake et al. “De-anonymisation attacks on Tor: A Survey.” *IEEE Communications Surveys*, 2021, pp. 2324–2350, doi:10.1109/COMST.2021.3093615

[8] P. Choorod et al., ”Classifying Tor Traffic Encrypted Payload Using Machine Learning”, *IEEE Access*, Volume 12, 2024.

[9] “Onion Services.” *ONION SERVICES | Tor Project | Tor Browser Manual*, tb-manual.torproject.org/onion-services/. [Accessed Jul. 18, 2023]

[10] “Tor Specifications.” *Hidden Services: Overview and Preliminaries. - Tor Specifications*, spec.torproject.org/rend-spec/overview.html#:~:text=Introduction%20Point%20%2D%2D%20A%20Tor,which%20relays%20traffic%20between%20them. [Accessed Jul. 18, 2023]

[11] “Bridges.” *BRIDGES | Tor Project | Tor Browser Manual*, tb-manual.torproject.org/bridges/. [Accessed Jul. 18, 2023]

[12] “Tor Specifications.” *Creating Circuits - Tor Specifications*, spec.torproject.org/tor-spec/creating-circuits.html. [Accessed Jul. 18, 2023]

[13] Dingledine, Roger, et al. *Tor: The Second-Generation Onion Router*, 1 Jan. 2004, https://doi.org/10.21236/ada465464.

[14] Wright, Matthew K., et al. “The predecessor attack.” *ACM Transactions on Information and System Security*, vol. 7, no. 4, 11 Nov. 2004, pp. 489–522, https://doi.org/10.1145/1042031.1042032.

[15] Abbott, Timothy G., et al. “Browser-based attacks on tor.” *Privacy Enhancing Technologies*, June 2007, pp. 184–199, https://doi.org/10.1007/978-3-540-75551-7_12.

[16] Bauer, Kevin, et al. “Low-resource routing attacks against tor.” *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society*, 29 Oct. 2007, https://doi.org/10.1145/1314333.1314336.

[17] X. Fu, et al. “One cell is enough to break Tor’s anonymity,” in Proc. Black Hat Tech. Security Conf., July 2009, pp. 578–589.

[18] Bauer, Kevin, Dirk Grunwald, et al. “Predicting tor path compromise by Exit Port.” *2009 IEEE 28th International Performance Computing and Communications Conference*, Dec. 2009, https://doi.org/10.1109/pccc.2009.5403852.

[19] Wang, Xiaogang, et al. “A potential HTTP-based application-level attack against tor.” *Future Generation Computer Systems*, vol. 27, no. 1, Jan. 2011, pp. 67–77, https://doi.org/10.1016/j.future.2010.04.007.

[20] Ling, Zhen, et al. “A new cell-counting-based attack against tor.” *IEEE/ACM Transactions on Networking*, vol. 20, no. 4, Aug. 2012, pp. 1245–1261, https://doi.org/10.1109/tnet.2011.2178036.

[21] Rochet, Florentin, and Olivier Pereira. “Dropping on the edge: Flexibility and traffic confirmation in Onion Routing protocols.” *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 2, 20 Feb. 2018, pp. 27–46, https://doi.org/10.1515/popets-2018-0011.

[22] S. Le Blond et al., “One bad apple spoils the bunch: Exploiting P2P applications to trace and profile Tor users,” in Proc. 4th USENIX Workshop Large-Scale Exploits Emergent Threats, March 2011, pp. 1–8.

[23] Yang, Ming, et al. “An active de-anonymizing attack against Tor Web Traffic.” *Tsinghua Science and Technology*, vol. 22, no. 6, Dec. 2017, pp. 702–713, https://doi.org/10.23919/tst.2017.8195352.

[24] Herrmann, Dominik, et al. “Website fingerprinting.” *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, 13 Nov. 2009, https://doi.org/10.1145/1655008.1655013.

[25] Panchenko, Andriy, et al. “Website fingerprinting in Onion Routing based anonymization networks.” *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*, 17 Oct. 2011, https://doi.org/10.1145/2046556.2046570.

[26] Cai, Xiang, et al. “Touching from a distance.” *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, 16 Oct. 2012, https://doi.org/10.1145/2382196.2382260.

[27] Wang, Tao, and Ian Goldberg. “Improved website fingerprinting on tor.” *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, 4 Nov. 2013, https://doi.org/10.1145/2517840.2517851.

[28] He, Gaofeng, et al. “A novel active website fingerprinting attack against Tor Anonymous System.” *Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, May 2014, https://doi.org/10.1109/cscwd.2014.6846826.

[29] T. Wang, et al. “Effective attacks and provable defenses for Website fingerprinting,” in Proc. 23rd USENIX Security Symp, August 2014, pp. 143–157.

[30] Arp, Daniel, et al. “Torben.” *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, 14 Apr. 2015, https://doi.org/10.1145/2714576.2714627.

[31] Panchenko, Andriy, Fabian Lanze, et al. “Website fingerprinting at internet scale.” *Proceedings 2016 Network and Distributed System Security Symposium*, Feb. 2016, https://doi.org/10.14722/ndss.2016.23477.

[32] J. Hayes, et al. “K-fingerprinting: A robust scalable Website fingerprinting technique,” *Proceedings 25th USENIX Security Symposium*, August 2016, pp. 1187–1203.

[33] Nasr, Milad, et al. “Compressive Traffic Analysis.” *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 30 Oct. 2017, https://doi.org/10.1145/3133956.3134074.

[34] Greschbach, Benjamin, et al. “The effect of DNS on Tor’s anonymity.” *Proceedings 2017 Network and Distributed System Security Symposium*, 2017, https://doi.org/10.14722/ndss.2017.23311.

[35] Sirinam, Payap, et al. “Deep fingerprinting.” *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 15 Oct. 2018, https://doi.org/10.1145/3243734.3243768.

[36] Rimmer, Vera, et al. “Automated website fingerprinting through Deep Learning.” *Proceedings 2018 Network and Distributed System Security Symposium*, Feb. 2018, https://doi.org/10.14722/ndss.2018.23105.

[37] Zhuo, Zhongliu, et al. “Website fingerprinting attack on anonymity networks based on Profile Hidden Markov model.” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, May 2018, pp. 1081–1095, https://doi.org/10.1109/tifs.2017.2762825.

[38] Pulls, Tobias, and Rasmus Dahlberg. “Website fingerprinting with website oracles.” *Proceedings on Privacy Enhancing Technologies*, vol. 2020, no. 1, 1 Jan. 2020, pp. 235–255, https://doi.org/10.2478/popets-2020-0013.

[39] Brown, Daniel, and Natalija Vlajic. “Poster: Novel client-side watermarking technique for tor user de-anonymization.” *Proceedings of the 2023 ACM on Internet Measurement Conference*, 24 Oct. 2023, https://doi.org/10.1145/3618257.3624997.

[40] Murdoch, S.J., and G. Danezis. “Low-cost traffic analysis of tor.” *2005 IEEE Symposium on Security and Privacy (S&Amp;P’05)*, May 2005, https://doi.org/10.1109/sp.2005.12.

[41] N. S. Evans, et al. “A practical congestion attack on Tor using long paths,” *Proceedings 18th USENIX Security Symposium*, August 2009, pp. 33–50.

[42] K. Bauer, et al. “On the optimal path length for Tor.” *Proceedings HotPets Conjunct. 10th Int. Symposium Privacy Enhanc. Technol. (PETS)*, 2010, pp. 1–17.

[43] Hopper, Nicholas, et al. “How much anonymity does network latency leak?” *ACM Transactions on Information and System Security*, vol. 13, no. 2, Feb. 2010, pp. 1–28, https://doi.org/10.1145/1698750.1698753.

[44] Chakravarty, Sambuddho, et al. “Traffic analysis against low-latency anonymity networks using available bandwidth estimation.” *Computer Security – ESORICS 2010*, Sept. 2010, pp. 249–267, https://doi.org/10.1007/978-3-642-15497-3_16.

[45] Sulaiman, Muhammad Aliyu, and Sami Zhioua. “Attacking tor through unpopular ports.” *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*, July 2013, https://doi.org/10.1109/icdcsw.2013.29.

[46] Chan-Tin, Eric, et al. “Revisiting circuit clogging attacks on tor.” *2013 International Conference on Availability, Reliability and Security*, Sept. 2013, https://doi.org/10.1109/ares.2013.17.

[47] Geddes, John, et al. “How low can you go: Balancing performance with anonymity in tor.” *Privacy Enhancing Technologies*, July 2013, pp. 164–184, https://doi.org/10.1007/978-3-642-39077-7_9.

[48] Mavroudis, Vasilios, et al. “On the privacy and security of the ultrasound ecosystem.” *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 2, 1 Apr. 2017, pp. 95–112, https://doi.org/10.1515/popets-2017-0018.

[49] Arma “Tor Security Advisory: ‘Relay Early’ Traffic Confirmation Attack: Tor Project.” *The Tor Project*, 30 July 2014, blog.torproject.org/tor-security-advisory-relay-early-traffic-confirmation-attack/.

[50] Arma. “Traffic Correlation Using Netflows: Tor Project.” *The Tor Project*, 14 Nov. 2014, blog.torproject.org/traffic-correlation-using-netflows/.

[51] Gianchandani, Prateek. “Timing Analysis Attacks in Anonymous Systems.” *Infosec*, 4 Feb. 2012, www.infosecinstitute.com/resources/hacking/timing-analysis-attacks/.

[52] Iacovazzi, Alfonso, and Yuval Elovici. “Network flow watermarking: A survey.” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, 2017, pp. 512–530, https://doi.org/10.1109/comst.2016.2604405.

[53] Radha, Raval, and Deepak Upadhyay. “Tor Traffic Classification using Deep Learning.” *Journal of Emerging Technologies and Innovative Research*, vol. 6, no. 5, May 2019, pp. 360–366, JETIR1905555.

[54] Sarkar, Debmalya, et al. “Detection of tor traffic using Deep Learning.” *2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA)*, Nov. 2020, https://doi.org/10.1109/aiccsa50499.2020.9316533.

[55] Akbari, Iman, et al. “A look behind the curtain: Traffic classification in an increasingly encrypted web.” *ACM SIGMETRICS Performance Evaluation Review*, vol. 49, no. 1, 22 June 2021, pp. 23–24, https://doi.org/10.1145/3543516.3453921.

[56] A, VishnuPriya., et al. “RNN-LSTM based deep learning model for Tor Traffic Classification.” *Cyber-Physical Systems*, vol. 9, no. 1, 24 May 2021, pp. 25–42, https://doi.org/10.1080/23335777.2021.1924284.

[57] Sarwar, Muhammad Bilal, et al. “DarkDetect: Darknet traffic detection and categorization using modified convolution-long short-term memory.” *IEEE Access*, vol. 9, 2021, pp. 113705–113713, https://doi.org/10.1109/access.2021.3105000.

[58] He, Liukun, et al. “FlowMFD: Characterisation and classification of tor traffic using MFD chromatographic features and spatial–temporal modelling.” *IET Information Security*, vol. 17, no. 4, 25 May 2023, pp. 598–615, https://doi.org/10.1049/ise2.12118.

[59] Hu, Feifei, et al. “Network traffic classification model based on attention mechanism and spatiotemporal features.” *EURASIP Journal on Information Security*, vol. 2023, no. 1, 12 July 2023, https://doi.org/10.1186/s13635-023-00141-4.

[60] Malekghaini, Navid, et al. “Deep learning for encrypted traffic classification in the face of Data Drift: An empirical study.” *Computer Networks*, vol. 225, Apr. 2023, p. 109648, https://doi.org/10.1016/j.comnet.2023.109648.

[61] Said Elsayed, Mahmoud, et al. “Network Anomaly Detection Using LSTM based Autoencoder.” *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, 16 Nov. 2020, https://doi.org/10.1145/3416013.3426457.

[62] Srivatsavaya, Prudhviraju. “LSTM - Implementation, Advantages and Diadvantages.” *Medium*, Medium, 5 Oct. 2023, medium.com/@prudhviraju.srivatsavaya/lstm-implementation-advantages-and-diadvantages-914a96fa0acb#:~:text=Handling%20Long%20Sequences%3A%20LSTMs%20are,NLP)%20and%20time%20series%20analysis.

[63] *York University*, York University, www.yorku.ca/. [Accessed Jun. 27, 2024].

[64] "Welcome to University of Toronto." *University of Toronto*, www.utoronto.ca/. [Accessed Jun. 27, 2024].

[65] "Watch, Listen, and Discover with Canada's Public Broadcaster." *CBCnews*, CBC/Radio Canada, www.cbc.ca/. [Accessed Jun. 27, 2024].

[66] *Home | Toronto Sun Home Page | Toronto Sun*, torontosun.com/. [Accessed Jun. 27, 2024].

[67] "Main Page." *Wikipedia*, Wikimedia Foundation, 23 May 2024, en.wikipedia.org/wiki/Main_Page.

[68] "Let's Build from Here." *GitHub*, github.com/. [Accessed Jun. 27, 2024].

[69] *YouTube*, YouTube, www.youtube.com/. [Accessed Jun. 27, 2024].

[70] "BMO Personal Banking." *BMO*, www.bmo.com/en-ca/main/personal/. [Accessed Jun. 27, 2024].

[71] "Welcome to CIBC Personal Banking." *CIBC*, www.cibc.com/en/personal-banking.html. [Accessed Jun. 27, 2024].

[72] "Get the Personal Banking Products and Services You Need." *Personal Banking - RBC Royal Bank*, www.rbcroyalbank.com/personal.html. [Accessed Jun. 27, 2024].

[73] Sun, Haili, et al. "Neural-factor: Neural representation learning for website fingerprinting attack over Tor Anonymity." 2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Dec. 2022, pp. 435–440, https://doi.org/10.1109/trustcom56396.2022.00067.

[74] Gegenhuber, Gabriel K., et al. "An extended view on measuring tor as-level adversaries." *Computers & Security*, vol. 132, Sept. 2023, p. 103302, https://doi.org/10.1016/j.cose.2023.103302.

[75] Armstrong, H.L., and P.J. Forde. "Internet anonymity practices in computer crime." *Information Management & Computer Security*, vol. 11, no. 5, 1 Dec. 2003, pp. 209–215, https://doi.org/10.1108/09685220310500117.

[76] "HTML5 Canvas." *HTML5 Canvas | Tizen Docs*, docs.tizen.org/application/web/guides/w3c/graphics/canvas/. [Accessed Nov. 21, 2024].

[77] MozDevNet. "Blob - Web Apis: MDN." *MDN Web Docs*, developer.mozilla.org/en-US/docs/Web/API/Blob. [Accessed Nov. 21, 2024].

[78] MozDevNet. "Promise.All() - Javascript: MDN." *MDN Web Docs*, developer.mozilla.org/en-

US/docs/Web/JavaScript/Reference/Global_Objects/Promise/all. [Accessed Nov. 21, 2024].

[79] Karim, Fazle, et al. “LSTM fully convolutional networks for Time Series Classification.” *IEEE Access*, vol. 6, 14 Feb. 2018, pp. 1662–1669, https://doi.org/10.1109/access.2017.2779939.

[80] “TOR-User Deanonymization.” *Tor-User Deanonymization*, www.eecs.yorku.ca/~danb13/Website_Framework/TorDeanonymizationTorbenV2.html.

[81] Kate, Rohit J. “Using dynamic time warping distances as features for improved time series classification.” *Data Mining and Knowledge Discovery*, vol. 30, no. 2, 7 May 2015, pp. 283–312, https://doi.org/10.1007/s10618-015-0418-x.

[82] Tavenard, Romain. *An Introduction to Dynamic Time Warping*, 2021.

[83] Wei, Dagang. “Demystifying the Adam Optimizer in Machine Learning.” *Medium*, Medium, 30 Jan. 2024, medium.com/@weidagang/demystifying-the-adam-optimizer-in-machine-learning-4401d162cb9e.

[84] Ali, Moez. “Introduction to Activation Functions in Neural Networks.” *DataCamp*, DataCamp, 12 Sept. 2024, www.datacamp.com/tutorial/introduction-to-activation-functions-in-neural-networks.

[85] V., Mounish. “A Comprehensive Guide for SVM One-Class Classifier for Anomaly Detection.” *Analytics Vidhya*, 2 Apr. 2024, www.analyticsvidhya.com/blog/2024/03/one-class-svm-for-anomaly-detection/.

Appendices

Appendix 1

The JavaScript code behind the implementation for the Torben webpage has two crucial functions:

The first function is responsible for setting the delays on when the payload should arrive. This is done using the “setTimeout()” function present in JavaScript. It is also important to understand that the delays are not sequential (i.e., the second delay of 15 seconds will not wait until the first delay is passed and will always launch at the 15 second mark).

The second function is responsible for loading the payload into the appropriate element on the HTML webpage. To accomplish this, we first create the image element of which we want to store and set the appropriate attributes (should we want to we can also make the element invisible and have the same impact on the network traffic). Once the element is created, all that is left is to append it to the appropriate section within the HTML which we created before hand.

```
<script>
  function delay() {
    //Note The timeout occurs at the same time meaning
    //delay1+delay2+delay3 != total delay, it is
    //delay3 = total delay
    var firstDelay = setTimeout(Marker1, 7500);
    var secondDelay = setTimeout(Marker2, 15000);
    var thirdDelay = setTimeout(Marker3, 22500);
  }

  function Marker1() {
    var fiframe = document.createElement("IMG");
    fiframe.setAttribute("src", "Marker3b.jpg");
    fiframe.setAttribute("width", "600");
    fiframe.setAttribute("height", "400");
    document.getElementById("marker1").appendChild(fiframe);
  }

  function Marker2() {
    var fiframe = document.createElement("IMG");
    fiframe.setAttribute("src", "Marker3b1.jpg");
    fiframe.setAttribute("width", "600");
    fiframe.setAttribute("height", "400");
    document.getElementById("marker2").appendChild(fiframe);
  }

  function Marker3() {
    var fiframe = document.createElement("IMG");
    fiframe.setAttribute("src", "Marker3b2.jpg");
    fiframe.setAttribute("width", "600");
    fiframe.setAttribute("height", "400");
    document.getElementById("marker3").appendChild(fiframe);
  }
</script>
```

Figure 42: Torben Implementation Code

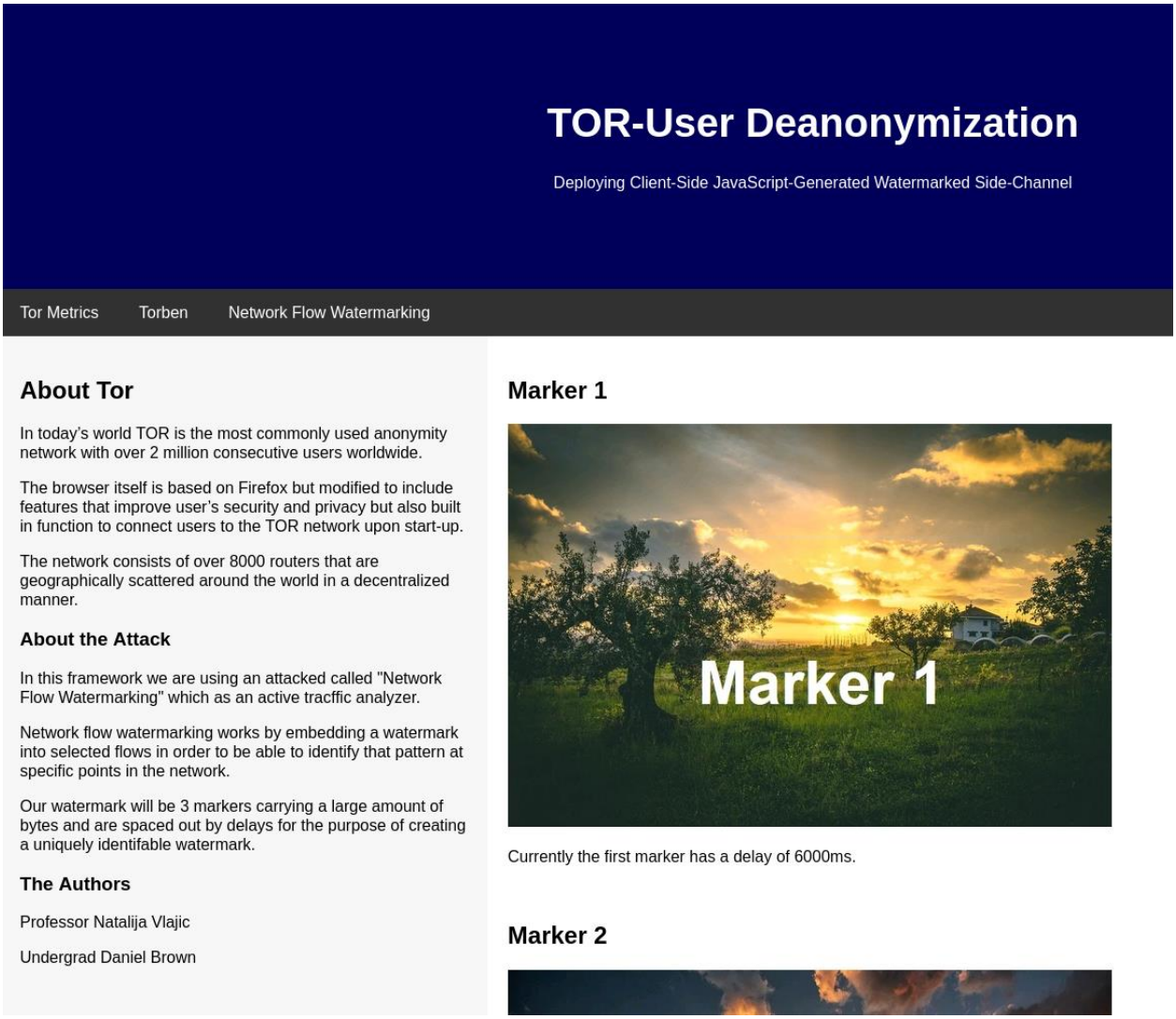


Figure 43: Torben Implementation Webpage

Appendix 2

Dynamic Time Warping (DTW) is a robust algorithm commonly used for measuring the similarity between two temporal sequences that may vary in speed or timing. In the context of watermark comparison, DTW is applied to align and evaluate the similarity between two sequences, such as a reference watermark and a potentially altered version extracted from a medium, see figure 44. Euclidean distance plays a central role in DTW by serving as the local distance measure between corresponding points in the sequences. At its core, DTW computes the cumulative distance between all potential alignments of the sequences by constructing a cost matrix, where each cell represents the Euclidean distance between points in the two sequences and the minimum cumulative distance needed to reach that point. The algorithm identifies the optimal warping path through this matrix, which represents the best alignment between the sequences under time shifts or distortions. The smaller the total warping cost along this path, the greater the similarity between the sequences. In watermark comparison, this approach helps account for non-linear distortions in the watermark due to noise, compression, or other transformations, making DTW particularly effective for robust similarity assessments [81].

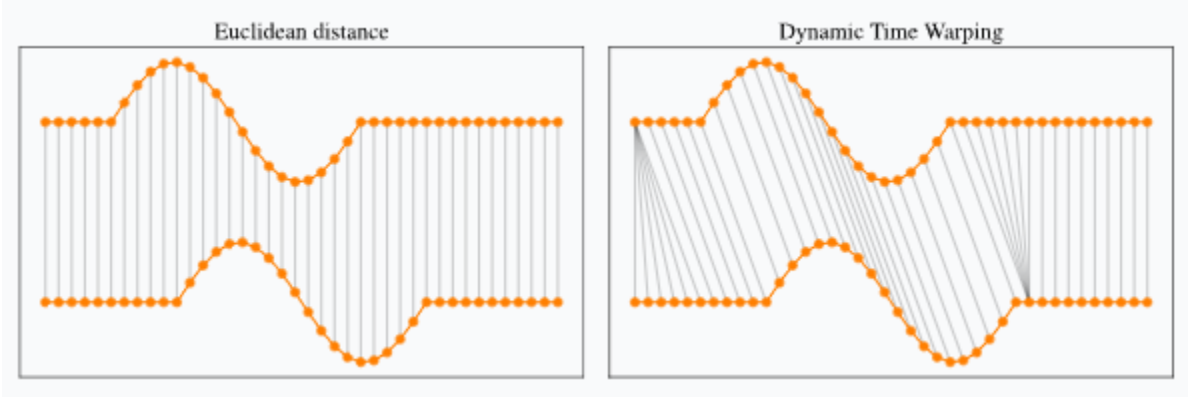


Figure 44: Dynamic Time Warping Sample Figure [82]