NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

MIGUEL SACADURA PAZ DOS SANTOS HORTA
Bachelor in Computer Science

TOR K-ANONYMITY AGAINST DEEP LEARNING WATERMARKING ATTACKS

VALIDATING A TOR K-ANONYMITY INPUT CIRCUIT ENFORCEMENT AGAINST A DEEP LEARNING WATERMARKING ATTACK

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
December, 2022

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

TOR K-ANONYMITY AGAINST DEEP LEARNING WATERMARKING ATTACKS

VALIDATING A TOR K-ANONYMITY INPUT CIRCUIT ENFORCEMENT AGAINST A DEEP LEARNING WATERMARKING ATTACK

MIGUEL SACADURA PAZ DOS SANTOS HORTA

Bachelor in Computer Science

Adviser: Professor Doutor Henrique João Lopes Domingos
Associate Professor, NOVA University Lisbon

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
December, 2022

ToR K-Anonymity against deep learning watermarking attacks

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v6.9.10) [43].

To a new chapter in life.

Acknowledgements

Firstly, I would like to address all the help that my advisor Professor Henrique Domingos provided. He proficiently guided my thoughts and efforts when I most needed and his help was vital towards the development of this dissertation.

I would also like to express my deepest regards to my family for providing me with endless care and support and for always being able to grant me peace of mind.

Finally, I would like to thank my girlfriend for always being present, sharing my toughest moments. To my friends, the old ones and the new ones I made along the way, a big thanks for all the laughs and all the moments we lived together, which I will fondly remember.

“We can’t blame the technology when we make mistakes.”
(Tim Berners-Lee)

Abstract

It is known that totalitarian regimes often perform surveillance and censorship of their communication networks. The Tor anonymity network allows users to browse the Internet anonymously to circumvent censorship filters and possible prosecution. This has made Tor an enticing target for state-level actors and cooperative state-level adversaries, with privileged access to network traffic captured at the level of Autonomous Systems(ASs) or Internet Exchange Points(IXPs).

This thesis studied the attack typologies involved, with a particular focus on traffic correlation techniques for de-anonymization of Tor endpoints. Our goal was to design a test-bench environment and tool, based on recently researched deep learning techniques for traffic analysis, to evaluate the effectiveness of countermeasures provided by recent approaches that try to strengthen Tor's anonymity protection. The targeted solution is based on K-anonymity input covert channels organized as a pre-staged multipath network.

The research challenge was to design a test-bench environment and tool, to launch active correlation attacks leveraging traffic flow correlation through the detection of induced watermarks in Tor traffic. To de-anonymize Tor connection endpoints, our tool analyses intrinsic time patterns of Tor synthetic egress traffic to detect flows with previously injected time-based watermarks.

With the obtained results and conclusions, we contributed to the evaluation of the security guarantees that the targeted K-anonymity solution provides as a countermeasure against de-anonymization attacks.

Keywords: *Internet censorship*; *Anonymization networks*; *Tor Network*; *K-anonymization Tor re-enforcements*; *Covert circuits* ; *Network flow watermarking*; *Tor flow correlation attacks*.

Resumo

Já foi extensamente observado que em vários países governados por regimes totalitários existe monitorização, e consequente censura, nos vários meios de comunicação utilizados. O Tor permite aos seus utilizadores navegar pela internet com garantias de privacidade e anonimato, de forma a evitar bloqueios, censura e processos legais impostos pela entidade que governa. Estas propriedades tornaram a rede Tor um alvo de ataque para vários governos e ações conjuntas de várias entidades, com acesso privilegiado a extensas zonas da rede e vários pontos de acesso à mesma.

Esta tese realiza o estudo de tipologias de ataques que quebram o anonimato da rede Tor, com especial foco em técnicas de correlação de tráfegos. O nosso objetivo é realizar um ambiente de estudo e ferramenta, baseada em técnicas recentes de aprendizagem profunda e injeção de marcas de água, para avaliar a eficácia de contramedidas recentemente investigadas, que tentam fortalecer o anonimato da rede Tor. A contramedida que pretendemos avaliar é baseada na criação de multi-circuitos encobertos, recorrendo a túneis TLS de entrada, de forma a acoplar o tráfego de um grupo anonimo de K utilizadores. A solução a ser desenvolvida deve lançar um ataque de correlação de tráfegos recorrendo a técnicas ativas de indução de marcas de água. Esta ferramenta deve ser capaz de correlacionar tráfego sintético de saída de circuitos Tor, realizando a injeção de marcas de água à entrada com o propósito de serem detetadas num segundo ponto de observação. Aplicada a um cenário real, o propósito da ferramenta está enquadrado na quebra do anonimato de serviços secretos fornecidos pela rede Tor, assim como os utilizadores dos mesmos.

Os resultados esperados irão contribuir para a avaliação da solução de anonimato de K utilizadores mencionada, que é vista como contramedida para ataques de desanonimização.

Palavras-chave: *Censura na Internet*; *Redes de anonimato*; *A rede Tor*; *Reforços do Tor com K circuitos anónimos*; *Circuitos encobertos* ; *Injeção de marcas de água*; *Ataques de correlação de fluxos Tor*.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Context and motivation

The internet grew to be, without a doubt, the most used means of communication. For countries controlled by authoritarian regimes, it is the perfect channel for citizens to exercise their suppressed freedom of speech. Consequently, these totalitarian states invest vast amounts of resources in filtering and blocking their citizen's access to the internet to maintain social and political supremacy [2, 11, 21].

Internet Censorship in Authoritarian Regimes

These *Censors*, attempt to block all content that isn't aligned with their political views, and in many cases religious beliefs. They resort to technical, legal and extra-legal strategies to censor information before it reaches its citizens [11]. When dealing with illegal content being spread through servers from within their jurisdiction, these governments proceed to take down the source through legal action. To deal with the content being spread from the outside, censors use their leverage to control domestic ISPs and employ complex traffic filtering systems [4, 33, 80]. Using several techniques, ranging from IP address and domain name blocking to traffic analysis, these systems regulate the information that enters the censor's jurisdiction.

The Tor network

The Onion Router, or Tor [74], is a volunteer-based network that allows for low-latency, private, and anonymous communication for TCP-based applications, such as web browsing. It was developed with the intent of giving everyone "private access to an uncensored web"and, for many Tor users, this tool is their only means of accessing free, unfiltered internet. Tor is based on a refined implementation of the onion routing technique [66]. Through the use of several cryptographic operations, Tor encapsulates a message within several layers, like an onion, and sends that message through a sequence of relays until it reaches the final destination [13]. Given its privacy and anonymization guarantees, this network allows for citizens, journalists, whistle-blowers and many more, to freely access the internet from inside authoritarian-government controlled countries which perform heavy internet censorship [45, 65].

The Censorship Race

Tor's effectiveness in circumventing censorship systems started getting more attention from censors who started focusing their efforts to block Tor [42, 45, 65, 80]. This gave rise to an "arms race"between the censors and the censorship-resistant tools. The censors have to come up with new strategies for blocking these circumvention systems, and these have to overcome the censors' blocking efforts and research new ways of staying hidden.

Tor vulnerability to end-to-end correlation attacks

The Tor design was implemented to resist an attacker who has control of only a fraction of the network [13]. However, it is vulnerable to de-anonymization attacks from an AS-level adversary with privileged influence on both ends of the connection, for instance, an ISP collaborating with an authoritarian government. End-to-end traffic correlation approaches [19, 38, 49, 70, 84, 85] fall amongst this category of attacks to which Tor is vulnerable, where network flow watermarking attacks [23, 28, 78], especially solutions supported with Machine Learning and Deep-Analysis(ML&DA) algorithms [50, 67], are the main object of study in this dissertation. Several different solutions have been researched but, unfortunately, no approach has been found that completely fends off these attacks. Instead, they serve as mitigation strategies that attempt to stop an adversary from gaining the necessary conditions to deploy such attacks. The more researched approaches are aligned with evading the establishment of circuits from within an AS-level adversary's sphere of influence [1, 12, 16, 35, 39, 59, 69, 70] or Tor protocol reinforcements [3, 51, 56, 60, 71].

1.2 Research opportunity

Due to its strong anonymity properties, Tor has become an interesting target for state-level actors or cooperative state-level entities, for the good and the bad. This includes, for example, the interests of repressive governments in breaking Tor anonymity conditions to prevent political dissidence or reporting activities about violations of civil rights, abuse of authority, or corruption news [2, 4, 11, 21, 33, 80]. On the other hand, Tor de-anonymization is also targeted by state-level adversaries for law enforcement, or to fight against organized online criminal activities [53]. Many attacks have been developed in the nineteen years elapsed since the Tor network's initial launch, in October 2002. As a taxonomy reference in [18], attacks against Tor's anonymity and privacy guarantees can be grouped into seven categories: traffic correlation, traffic congestion, timing-based correlations, fingerprinting, DoS, supportive attacks, and identification of hidden services.

Other authors also classify attacks as passive (based only on traffic analysis and observations) or active (attackers induce pro-actively perturbations in the Tor operation or can manipulate or control Tor resources). Broadly, traffic correlations can be conducted on single-end or end-to-end Tor onion-routed circuits [18]

One of the most significant threats to Tor is correlation attacks [19, 38, 49, 50, 70, 84, 85], regarded as end-to-end attacks where an adversary monitors both edges of a Tor circuit and seeks to establish a relation between flows (potentially asymmetric) observed at the entry and exit points. The success of passive correlation attacks can be significantly improved through the usage of pro-active watermarking inductions[23, 28, 78], with the adversary manipulating "invisible" traffic patterns on inspected Tor circuits, for example introducing predictability in features such as packet-delays, controlled jitter conditions or selective packet-drops to improve the correlation effectiveness. These pro-active watermarking strategies also allow for other detection-like approaches which attempt to distinguish these induced patterns in traffic features, relating outgoing watermarked traffic to its point of origin or even decoding *invisible* embedded data [23, 24, 25, 28, 67, 78]

On the other hand, to enhance the Tor-network resistance, the research community proposed different solutions over time where some of those proposals are now native mechanisms in the more recent versions of Tor (Bridges, Tor Relay Nodes, Tor-Onion-Services/Tor Hidden Services). More recently, other researched solutions include Tor-strengthening mechanisms based on K-anonymity input covert channels [51, 71]. However, the security analysis on such solutions is limited, not including traffic correlation techniques based on pro-actively induced watermarking, and not using the more effective traffic analysis approaches where the correlations are based on deep learning of Tor traffic properties, used in more recent solutions such as [50, 67].

1.3 Goals

Our goal for the thesis elaboration was to design a test-bench environment and tool, based on the more promising researched machine-learning techniques for ML&DA traffic analysis, to evaluate the effectiveness of countermeasures provided by recent approaches that try to strengthen the Tor network with the use of K-anonymity input covert channels. We focused the use of the designed test bench and tool in an extended evaluation of a previous solution that provides TLS tunnelled input circuits, and the possible inclusion of media-streaming encoded covert circuits as an alternative for the K-anonymized covert circuits [71].

The research challenge for our test-bench environment and the designed tool was the ability to launch active correlation attacks, leveraging proactively induced watermarking-based probes and watermark detection strategies through deep learning of marked traffic. This tool can analyse traffic correlations on Tor's synthetic egress traffic flows, considering the Tor network architecture, efficiently injecting and detecting proactively induced watermarks to track down Tor users and Tor Onion Service providers used by such users.

With the expected results, we intended to contribute to the security evaluation of the targeted K-anonymity solution [71] to find new required enforcements and countermeasures against de-anonymization attacks.

1.3.1 Contributions

Considering the thesis goal, we now present the expected contributions for this dissertation:

- A specification proposal for a test-bench environment and tool to conduct deep learning of watermarked traffic flows, with the objective of de-anonymizing communications from Tor users and Tor protected onion services, supported by the targeted pre-existent K-anonymization solution [71];
- The implementation of the specified test-bench environment and tool prototype for performing the de-anonymization of Tor involved users and services when protected by the targeted K-anonymization solution described earlier;
- Validation of the implemented solution, based on an extensive experimental evaluation of the implemented tool's effectiveness and performance, regarding its full parametrization and adaptability against diverse scenarios simulating different network jitter conditions;
- Analysis of the experimental evaluation results and conclusions, to suggest future improvements to the targeted Tor-Strengthening solution against deep learning traffic correlation attacks with proactively induced watermarking techniques performed by AS-level adversaries.

1.4 Report Organization

The remaining chapters are organized in the following way: chapter 2 presents an initial background of the Tor network's architecture and operation, provided security and anonymization features; chapter 3 was organized as a survey, covering typologies of Tor de-anonymization attacks, countermeasures against such attacks and the most relevant solutions involving traffic correlation attack techniques and tools, as well as Tor-enforcement solutions considering state-level or cooperative global adversaries; chapter 4 will provide the tool's conceptual specification, covering its global architecture as well an in-depth description of the tool's components and their operation; chapter 5 covers the deployment specifications and necessary adaptations of the TIR prototype as well as all details regarding our tool's implementation; in chapter 6 we specify the methodology and metrics used in the evaluations of our tool and TIR's security analysis and show the obtained results, also providing security recommendations based on the achieved results;

finally, chapter 7 draws the conclusions for this dissertation, glancing over the main contributions, some open-issues and future implementations regarding a possible extension of this work.

2 Background

2.1 Internet censorship

For totalitarian regimes, it is crucial to have control over the country's means of communication in order to maintain political supremacy [2, 21]. This also applies to countries that have an extreme and intolerant religious influence, where these beliefs are enforced on their citizens. The internet, being the largest and most influential way of communication, is the perfect means for fighting back against these abusing authorities. This drives the big majority of these governments to censor and control their citizens' online communications, which might threaten the government's political control, or even their population's exposure to content that is not be aligned with the country's religious beliefs.

From a list of 45 chosen countries, 26 left evidence of performing some kind of filtering and blocking of internet traffic [11]. The majority of these 26 countries perform filtering on sensitive social and political content, internet tools such as anonymizers, censorship circumvention tools, social media platforms, streaming and P2P file sharing sites and even content that might threaten national security. The type and extent of filtering from these countries suggest that they get rid of any impediments to performing internet censorship and then they turn to extend the filtering to political/social content and censorship circumvention tools.

2.1.1 Content control and filtering approaches

When it comes to content control, censors have several approaches [11]. If the content is being spread from domestic servers, they resort to legal action. If the content is hosted in a foreign source it may be harder to deal with. Sometimes they have to turn to either convincing private companies, or even governments, to take down the propagated content, or just filter incoming traffic from that source. Some more aggressive approaches have been recently growing such as denial-of-service attacks or even targeted hacking [11].

In order to filter high volumes of information, censors use communication *distinguishers* to develop models that categorize allowed traffic and traffic that must be disrupted [33]. They can perform several kinds of traffic control by blocking certain destination IP addresses, HTTP hosts, domain names and blacklisted keywords. They also fingerprint forbidden protocols by creating a statistical model based on traffic flow distinguishers and behaviour [33].

Censors also perform some kinds of direct censorship by, for instance, installing software that disrupts access to information, directly on the user [33]. On the publisher side, censors can corrupt information to be published or just disrupt the whole publication process. Connection-wise, they can: degrade performance by manipulating the link between the user and publisher introducing delays; blocking IP addresses, ports, domain and host names which is quite common; corrupt routing information to disrupt access; corrupt flow content through the injection of fake server responses; and corrupt protocol semantics by manipulating or just tearing down connections [4, 33].

2.1.2 The Censorship race

There are two sides competing in this digital arms race, the censors and the ones who try to counter them, both consistently trying to innovate further than the other. This is evident in the development of China's censorship techniques when trying to block the Tor network. The *Great Firewall of China*(GFW) blocks Tor's public relays and it censors hidden bridges through active probing of suspected servers. Before, the GFW identified the use of the Tor protocol when performing deep package inspection on connections due to Tor's unique TLS cypher list. However, this was later prevented by a traffic obfuscation tool in development at the time [80]. Nowadays, the GFW has adapted to new traffic obfuscating tools and performs newly adjusted probing techniques [42]. Both sides are in constant development, trying to frustrate the other's attempts at producing new techniques that will beat the other side.

2.2 Tor Network Background

Currently, the largest anonymization network is The Onion Routing project, widely known as Tor [74]. With over two million connected users and six thousand public routers, this volunteer-based network allows for low-latency, private, anonymous communication and was developed with the common belief that "all internet users should have private access to an uncensored web".

2.2.1 What preceded Tor

Modern anonymization networks are based on Chaum's untraceable electronic mail from 1981 [10], which introduced a new technique for anonymous and private communication. This technique makes use of a public-key cryptosystem to build a layered structure on top of a message, effectively sealing it. This sealed message is then sent through a sequence of proxies, called *mixers*, that decrypt each layer before relaying it to the next *mixer* until it reaches the final recipient. Several years later, employing and refining some ideas from Chaum's work, the *onion routing* technique was introduced [66]. Users could establish bi-directional, near real-time private connections over a public network, resistant to both traffic analysis and eavesdropping, while maintaining anonymity. The routing network was composed of proxies named *onion routers* that have near-permanent socket connections with the other routers, where connection data is routed through.

2.2.2 The Tor network

The Tor network implements an improved version of the aforementioned *onion routing* technique, providing anonymity for TCP-based applications, like web browsing [13]. It is composed of publicly listed routers called relays, responsible for routing the connections. Each user runs an *onion proxy*(OP), local software that handles the connection with user applications, establishes circuits and fetches the directories with current network information and a list of the public relays, also called onion routers(OR). This proxy uses a SOCKS interface, supporting most TCP-based programs without the need for modifications. Each onion router maintains two different keys: a long-term identity key, to sign TLS certificates and its router descriptor; and a short-term onion key, to decrypt requests, set up circuits and negotiate ephemeral keys. Relays also maintain TLS connections to nodes they have been in recent contact with, which also generates a short-term link key for each connection.

Most traffic that travels through the network is in 512 bytes fixed-size cells that consist of a header and payload. This header carries a circuit identifier(*circId*) that specifies the circuit to which that cell belongs. Cells are divided into two types, each type with its several subtypes: *control* cells, interpreted by the receiving node, used for managing the circuits; and *relay* cells, which have an extra header containing the stream identifier(*streamID*), used for the data streams.

Circuits are the sequence of ORs which traffic is routed through. Before being sent to the first node of the circuit, *relay* cells are encapsulated in several layers of encryption, one for each node of the circuit, using the pre-negotiated keys with each OR. As it travels through the circuit, the encryption layers are decrypted, one layer per relay hop, until it reaches the exit node where it is relayed to the responder. When an OR receives a *relay* cell, it looks up the corresponding circuit and decrypts the cell with the negotiated key for that *circID*, before relaying the message to the next node. When an *onion proxy* receives a *relay* cell, it unwraps it with the session keys shared with each OR on the circuit, from the closest to the farthest. This design guarantees that only the exit node views the contents of the decrypted message and each OR only knows the identity of the previous and next nodes of the circuit. It is important to mention that Tor uses a *leaky pipe* circuit topology, meaning the exit node doesn't need to be the last one added to the circuit, as a way to prevent some passive attacks[13].

2.2.3 Establishment of Tor circuits

In Tor, since circuits take substantial time to be built due to public-key cryptography, each circuit can be shared by multiple TCP streams. To avoid delays, circuits can even be preemptively built by OPs. Users can also configure their applications, personalizing which streams share the same circuit. When creating a circuit, to avoid flooding the network with relays state information, clients get a list of the active relays through a *directory server*. These make a small group of redundant trusted nodes that provide signed directory documents describing connection information from known routers and their current state.

The establishment of Tor circuits is done incrementally wherein each iteration the OP negotiates a symmetric key with the onion router it wants to include in the circuit next. After a circuit is closed, these symmetric keys are deleted which provides perfect forward secrecy. The OP begins by sending a *control create* cell to the entry OR of the circuit, containing the first half of the Diffie-Hellman handshake encrypted to the onion key of that node. Once this first hop is established, the OP extends it by sending *relay extend* cells to the entry router. These cells will contain the encrypted first part of the handshake, also encrypted to the onion key of the node being added, and will specify the address of the next OR so they are routed accordingly. After this iterative process, the circuit is ready to initiate a stream.

The stream data is passed through *relay* cells. To open a data stream to a remote host, the application asks the OP via the SOCKS interface. The proxy chooses an acceptable exit node from the most recent circuit created, or just creates a new one, and sends it a *relay begin* cell passing a new *streamID*. Once the exit note establishes a connection to the remote host, it sends a *relay connected* cell back to the OP, which then notifies the application about the open stream.

2.2.4 Tor Hidden Services

Tor also provides a way to build location-hidden services [13] where the service provider doesn't need to reveal his IP address. To host a hidden service (also called Onion Service), a provider generates the service identity key and chooses its *introduction points*: ORs where he will be waiting for requests. After the points are signed and advertised on the onion lookup service, a user who wants to connect chooses an OR as a *rendevouz point*(RP) and shares it with the provider through an *introduction point*. The provider builds a circuit to the shared RP and the user's OP sends a relay begin cell along the circuit, initiating the data stream.

2.2.5 Tor Anonymity Guarantees

The internet is a highly surveilled network and, as we've seen in 2.1, it suffers from severe content censorship in countries administrated by oppressive regimes. Tor's help is crucial for giving people, censored by these regimes, a way to communicate with the outside. People like journalists, whistle-blowers, citizens affected by oppressive regimes or even people who want to avoid corporate profiling. They are the ones who need the Tor network's privacy and anonymization guarantees to safely access the internet [45, 65]. However, this anonymization also attracts a lot of attention from illegal activities such as the hosting of criminal-orientated websites, pretty popular within Tor's hidden-services [53].

2.2.6 Tor Bridges

Tor's effectiveness in censorship circumvention and anonymity caused it to become a target. Censors want to break it so they can maintain political control[80] and law enforcement wants to stop illegal online activities and services that act through it [20].

In order to cut access to Tor, censors begin by blocking public relays through IP address and TCP port combination, and even the Tor website [62]. This way, users are blocked from entering through the conventional way. In 2007, the Tor project began developing bridges: onion routers that are not publicly listed in the relays directory [63], making them not so trivial to be blocked. This development gave censored users the option to choose bridges as the starting node for their circuits, potentially circumventing the public relay block.

Bridges, however, are still vulnerable when Tor traffic is identifiable. Taking advantage of packet header analysis to find the use of specific protocols or well-known ports or even fingerprinting through specific strings, byte patterns and packet properties are approaches for traffic classification [40] and are used to fingerprint Tor. For example, Tor uses fixed-size cells of 512 bytes, which can make the packet-size distribution of the protocol distinctive and identifiable to some fingerprinting attacks [13]. Another example can be seen in the TLS connections created for relay communication which don't use the standard port for TLS traffic [47]. They use specific ports for communication between relays and communication with directories. Also, in the TLS handshake, the Tor protocol always has present the extension "server_name"which consistently follows the same format. These kinds of patterns can be used to identify Tor traffic and subsequently get the unlisted bridges blocked by the censor.

2.2.7 Attacks against Tor

Over the years, more attacks against Tor have been discovered with more complex and hybrid approaches. Evers [18] collects thirteen years of data about these attacks and arranges into from two different perspectives. Regarding the type, these attacks can be divided into *Active attacks*, where the adversary has the power to manipulate the network's traffic, for example injecting or delaying packets; and in *Passive attacks* where he can only observe such traffic. The attack's coverage can be distinguished from *Single-end attacks*, only the exit or the entry node are monitored/manipulated, or *end-to-end* *attacks*, where both the entry and exit nodes are affected. Evers [18] also categorizes Tor attacks, according to their methods and goals, in seven different types: *Correlation Attacks*, *Congestion Attacks*, *Timing Attacks*, *Fingerprinting Attacks*, *Denial of Service Attacks*, *Supportive Attacks* and *Revealing Hidden Services Attacks*.

Considering the relevance that attacks against Tor have for this dissertation, more specifically traffic correlation attacks, we will cover their different types and respective countermeasures in their dedicated sections of the Related Work chapter, 3.1 and 3.2.

2.2.8 Tor pluggable transports

As mentioned before, bridges can be found, and subsequently blocked, if Tor traffic is identifiable by a censor that analyses network traffic. For this reason, Tor bridges can support traffic obfuscation tools called *Pluggable Transports*(PT) [56]. These tools alter the traffic flow properties from the connection between the user and bridge, so it is no longer identifiable as Tor traffic. It is important to mention that, since the current PT deployment strategy doesn't check for unsafe PTs running, bridges supporting several concurrent pluggable transports may reduce the security of the most secure ones [46]. Also, bridges that run additional non-Tor services, like SSH, are more vulnerable to being tracked down across IP address changes [46]. There are a few PTs available for use, we believe these four are the most relevant to cover:

- **obfs4** is a traffic obfuscation layer for TCP protocols that aims to stop an attacker from finding the protocol being used through the analysis of message contents [3]. The traffic obfuscation works through encrypting all application data with a key derived from both the client's and server's initial keys [56].
- **FTE**, "format-transforming-encryption", is a pluggable transport that encodes data in such a way that, if a censor uses regular expressions to distinguish between allowed and disallowed traffic, FTE makes the data look like allowed traffic [56].
- **meek**. This PT uses a technique called "domain fronting"to trick the censor into thinking the traffic is going to a legitimate server [56, 58], one that would cause too much collateral damage to block (for instance meek-azure). It works by exposing a legitimate domain name "outside"of the HTTPS request, in the DNS query and the TLS Server Name Indication, and "inside", in the HTTP Host header, appears the actual hostname.
- **Snowflake**. This circumvention tool makes use of volunteer snowflake proxies to pass Tor traffic through WebRTC connections [60]. A Tor user, using a snowflake client, starts by establishing a peer-to-peer WebRTC connection to a snowflake proxy. Then, the proxy connects to a Tor relay and starts receiving Tor traffic from the censored user, through the WebRTC connection, and relaying it to the entry node.

3 Related Work

Following the initial background presented in 2, we will now go over a related work analysis with relevant references aligned with the dissertation's goals. This chapter will be structured as such: section 3.1 will organize a typology of Tor attacks, covering four main categories; section 3.2 describes countermeasures that have been proposed as Tor enforcement security mechanisms, considering the attacks covered before; section 3.3 compiles different watermarking detection methods and tool important for the thesis contributions; lastly, section 3.4 is a summary of the above sections.

3.1 Tor Attacks

As mentioned in 2.2.7, attacks targeted at the Tor network have been growing in complexity over the years and several different approaches have been studied. For this study of related work, we consider Tor de-anonymization attacks to be the main target. After reviewing several de-anonymization techniques, we present a taxonomy of de-anonymization approaches, considered the most effective for the state-of-the-art. These will be divided in subsections as such: 3.1.1 - Specific correlation attacks; 3.1.2 - Enhanced correlation attacks; 3.1.3 - Network flow watermarking attacks; 3.1.4 - Attacks leveraged from internet routing level interceptions. Each category will cover possible relevant sub-categorizations.

3.1.1 Specific correlation attacks

A Correlation attack's goal is to link flows observed in two distinct observations sites of a circuit[18]. This flow correlation is based on traffic analysis and observable traffic properties, since the traffic's content can be substantially obfuscated in encrypted payloads. We've mentioned that these attacks may have an active or passive approach, i.e, the adversary may manipulate traffic or only observe it. The means to how an attacker gains these necessary conditions may also differ. It can be done by compromising a node or just running one [8, 17, 76], which is possible due to the volunteering nature of the Tor network. We may also consider network-level threats like Autonomous Systems(ASes) and

ISPs [49], which have inherent access to the possible target links. The correlation method itself may be based on several techniques such as deep learning, probabilistic models, statistical analysis, inherent Tor traffic fingerprinting or even watermarking observations [6, 18].

A lot of passive correlation techniques use downloaded file sizes as a way to correlate flows, performing packet counts from dedicated streams. Since Tor's packets have fixed 512 byte-sized packets, being able to discover file sizes through packet counting becomes a not-so-trivial task but nonetheless, still possible [68]. Furthermore, given that Tor works as a low-latency network, it does not apply countermeasures to mask packet timings (such as packet re-ordering or batching) making these features useful for attacks [85].

As stated earlier, correlation attacks mostly rely upon flows captured from client to entry-node communication and exit-node to destination server. The full communication path doesn't need to be known for the attack to effectively de-anonymize the circuit, since both the entry and exit nodes know, respectively, the client and the destination server IP addresses.

We will now introduce relevant specific correlation attacks, starting from a timing-based correlation approach.

Timing-based traffic correlations

Timing-based attacks are the first line of flow correlation attacks. The main method employed by these attacks consists in performing statistical analysis of traffic flow features to link ingress and egress portions of communication channels from low-latency anonymous systems such as Tor.

These flow features can range from packet arrival timings to traffic volume and other possible flow patterns [18]. For instance, [38] describes the threat that *Timing attacks* pose to low-latency mix systems through detailed simulations and analysis. This study analyses the arrival times of successive packets and shows these timings can effectively link traffic observed in two different relays of a circuit.

In [48], it is applied a relay congestion approach which consists in congesting different relays, one by one, while observing any latency differences in the traffic flow of the target circuit. The idea behind this approach is based on how applying congestion on the said relay, every circuit traversing it will suffer latency differences. Observing a connection and detecting latency variations, after congesting said relay, will confirm that relay's presence in the circuit of the observed connection. This attack is repeated until the circuit is de-anonymized. In order to congest the target relays they need to be publicly listed, so using hidden bridges will surely hamper this approach. Furthermore, this attack may become impractical with Tor growing in scale and circuits being periodically rebuilt.

In [17] the same underlying strategy is maintained but enhanced with a bandwidth amplification technique to deal with a larger-scale network. In another approach, [85] shows a novel statistical analysis of packet arrival timings and frequency that correlates flows in low-latency mix-based networks like Tor. The technique used extracted packet arrival timings and sample sizes, compiling them into a*flow pattern vector*, which can vary depending on the batching strategy used by the network. By correlating these vectors, this attack accurately determines the output node used by traffic that comes through an entry edge.

Other timing attacks, like [8], described an approach against low-latency network anonymity systems that can expose the network identity of network endpoints (users or relays) present in a target connection. It works by inducing fluctuations in the victim's TCP connection that creates traceable bandwidth patterns. In other approaches, [55] suggests a *Replay Attack* that exploits the fact that Tor uses AES in counter mode to encrypt the traffic's layers. By duplicating relay cells at the entry node, encryption and decryption counters would get disrupted resulting in decryption errors detectable at the exit node, effectively linking the flows.

Some defensive mechanisms that try to mitigate *timing-based attacks* have been suggested [48], however, to some degree, they all involve an increase in the network's latency so they are not feasible implementations to Tor. Instead, Tor relays send cells from different streams in a round-robin fashion. This is a critical issue in the research of mitigation solutions for Tor, that avoid *timing-based attacks* while not aggravating the latency and throughput conditions for practical use.

The state-of-the-art flow correlation technique for Tor is DeepCorr [50], which will be covered in 3.3.1 along with other important and enhanced correlation techniques.

Bandwith-based traffic correlations

Previous attacks resorted to traffic analysis based on the size and arrival timings of packets. Although proven successful, all of them required the involvement of a global adversary or cooperative state-level adversaries, possibly needing control over a vast number of relays and observed circuits. Chakravarty [8] introduces another approach to de-anonymize Tor users and even hidden services. The attack was employed through a colluding network endpoint that would introduce propagating bandwidth variations destined to the target. These bandwidth patterns would then be traced using a bandwidth estimation tool, such as LinkWidth [7]. It was concluded in his work [8] that the attack was mildly successful in a controlled real-world setting, representing an open direction for Tor attacks in the future.

3.1.2 Enhanced correlation attacks

Moving through timing and bandwidth-based correlation techniques, we will now go over a class of correlation attacks enhanced through complementary methods. We will start by describing a correlation attack that exploits the victim at the application-level [76], followed by Sybil correlation attacks that require the deployment of several malicious relays [14, 81]. Finally, we will cover the predecessor attack that hinges upon long-term conditions of a link [12, 19, 84].

Application-level traffic correlations

Wang [76] describes a potential application-level attack that can effectively compromise the anonymity of clients. The attack requires an adversary that is positioned at both ends of a Tor circuit by operating malicious relays and it works by detecting a user's web page request and responding with a malicious website. When the victim's browser processes the malicious webpage, it will initiate a deterministic number of malicious web connections to download the web page's contents, creating a pattern that can be recognisable when observing the traffic, and linking flows observed on both endpoints.

Sybil correlation attack

In a Sybil attack, the adversary model conditions include the possible deployment of malicious relays in the Tor network and the creation of an illusion that those nodes pertain to different entities. The goal is to manipulate as many nodes as possible to control a disproportionately large fraction of the network [18], giving the attacker several different vantage points to intercept user traffic. Most attacks on the Tor network gain effectiveness in relation to how much of the network's traffic an adversary can control [81].

Mitigation solutions for these kinds of attacks are not trivial. Douceur [14] suggests an approach where only trusted relays, certified by a central authority, can join the network. However, relying on a single control entity defeats the very purpose of the Tor network, and also exposes the system to other kinds of attacks. We underline that defence strategies against Sybil attacks do not effectively cancel them, they act as mitigation solutions by increasing the attacker's setup effort or costs.

Predecessor sender attack

The *Predecessor sender attack* is a robust traffic analysis approach that aims to identify single or multiple circuit initiators which communicate with the same destination, repeatedly over time. The idea here is to use the nature of Tor's circuit creation and log utilized relays in some repeated communication between two endpoints of Tor, inferring long-term conditions that can be exploited by a traffic analysis-capable adversary [19]. To achieve success, the amount of influence over the network the adversary must have grows linearly with the number of possible communication initiators. A possible solution against this attack is proposed in [84] which assumes a static network model, i.e. nodes do not leave the network. Where to defend against predecessor attacks, the authors propose to fix relays of Tor circuits in certain pre-known positions (e.g. entry, or exit). Tor operation today uses a similar approach that implements guard node (pre-defined entry relay of a circuit) rotation restrictions [12].

3.1.3 Network flow watermarking attacks

In this subsection, we cover attacks employed through traffic correlation, which make use of watermarking or fingerprinting techniques. We start by discussing the essentials of traffic correlation by watermarking and then finish by covering the diversity of watermarking attacks.

Traffic correlation by watermarking

First introduced in 2001 by Wang et al [78], watermarking analysis techniques were initially researched as solutions to support covert channels. Also in Wang's work [78], a *watermark* is defined as a small piece of data or pattern that identifies a connection upon its detection. This means *Network Flow Watermarking* can be seen as a type of traffic analysis technique [27]. It consists of adding a watermark, a specific pattern in the traffic's properties, identifiable by an observer monitoring the traffic who is searching for that specific watermark. It is also worth mentioning flow watermarking techniques have different uses, ranging from thwarting network cyberattacks [22] to launching their own [28].

Watermarking-based traffic analysis is addressed in two stages [27]: conversion of information into a watermark and embedding it into the flow – carried out by a system component called *watermarker*; and observing the flow, identifying it as *watermarked flow* and finally decoding it, retrieving the information from the watermark itself – carried out by watermark detectors. In particular, as stated in Iacovazzi's study [27], a watermarker component must have three main functions: (1) Filtering - selecting which target flows will be embedded with the watermark; (2) Encoding - encoding the information into symbols that will be mapped into a sequence of bits (to be used not necessarily contiguous across packets in traffic flows), which will, in turn, be transmitted through the watermark; (3) Spreading - selecting a diversity scheme for spreading the watermark bits, to improve resistance to interferences; and (4) Embedding - embedding the watermark into the carrier signal by slightly altering some of the carrier's features. The watermark detector only needs two phases: (1) Feature extraction - flow features that could potentially transport watermark bits are extracted and compiled into a descriptor vector; (2) Decoding - the detector computes the value of a function of the extracted features, which dictates if the flow is watermarked or not.

Watermarking diversity approach

Concerning the previous approach, a watermarking technique's type changes according to the kind of carrier chosen to be embedded with the watermark's bit. Iacovazzi [27] distinguishes four kinds of these carriers: *Content-based*, *Timing-based*, *Size-based*, and *Rate-based*. Regarding content-based watermarking, it is a method that injects the watermark directly into the payload or header of exchanged, unencrypted, data. Present in

[78], this kind of watermarking is hardly used since it needs to be executed within an unencrypted communication channel, which doesn't agree with current-day encryption guarantees, like the ones used in Tor. Timing-based watermarking is arguably the most popular. It relies on introducing a controlled degree of delay to selected packets, such that the sequence of arrival times and/or departure times (also called interleaving packet delays or IPD) is the watermark. This sequence is extracted from an observation point and the watermark is calculated as a function. As presented in several works about watermarking [23, 24, 25], this function can be computed as the mean balance of either IPD values, interval centroids or interval packet counts [27]. Regarding size-based watermarking, this technique consists in altering selected packet lengths by padding content sizes according to the watermark being embedded [64]. This method needs to be executed before traffic is encrypted which decreases its overall applicability, similarly to content-based watermarking. Finally, the last covered technique is rate-based watermarking which is done by injecting dummy traffic in an active connection so the rate of real traffic going through becomes a recognizable watermark. Of course, by fluctuating the rate of real traffic going through, this approach might be detected or just degrade the quality of the link. So, we are presented with a trade-off where on one hand, we can increase the scale of the traffic rate fluctuations to ease the watermark's detection and provide better robustness, and on the other hand, increasing fluctuations translates into a bigger probability of the attack's detection and degradation of the communication's quality.

When embedding the watermark in the carrier's signal, watermarks are *spread* across a specific domain, following specific *diversity schemes* in order to gain robustness against possible interferences in the channel which might destroy the watermark. These schemes are closely tied with the type of watermark and, consequently, optimized for the carrier protocol used [27]. In generic terms, we can characterize these diversity schemes in three different approaches [27]: Time diversity, where the watermark is replicated several times at different timings [23, 24, 25], the most used scheme in time-based watermarks; Frequency diversity, a scheme which creates interferences in the flow rate, to embed a pseudo-noise code that allows for the transmission and spreading of the watermark[26]; and finally, the Spatial diversity scheme, which consists in sending the embedded watermark through different channels. This must be accomplished in a lightweight way, otherwise, it could be difficult to detect the watermark in a useful time, and the watermarking signal must be amplified enough for its effective detection [22].

3.1.4 Attacks leveraged from internet routing level interceptions

Raptor

In [70] the authors discuss a set of attacks enabling ASes to compromise Tor user's anonymity. RAPTor is broken down into three different attacks, leveraged by the asymmetric nature of Internet routing and the dynamic aspects of BGP, the globally used Internet Border Gateway Protocol.

Asymmetric traffic analysis: Given the nature of internet routing, paths are often asymmetric. It might be the case that an adversary can only see opposite direction flows in different observation sites. This attack leverages the use of TLS to perform successful correlation attacks [70].

Churn exploits: This attack exploits the natural internet router's churn to increase an AS-level adversary's surveillance capability over time. Through churn injection, this attack could improve the amount of Tor circuits a single observing AS could compromise by 50% [70].

BGP interception: In this attack, a malicious AS hijacked traffic otherwise directed at another AS by falsely advertising its ownership over a range of IP addresses owned by the legitimate one [70].

3.2 Tor Countermeasures

One of the main vulnerabilities of Tor is correlation-based traffic analysis, as initially presented before. These attacks are particularly amplified if we consider global adversaries. A global adversary is an adversarial model in which it is possible to conduct traffic correlations through multiple observation sites which, in the worst scenario, the adversary is capable of eavesdropping on the entire network. This adversary model assumption was considered in the research of Tor strengthening solutions that could be used as global countermeasures. These countermeasures could be targeted by solutions complementing a variety of defence mechanisms that have been studied by the research community.

In our related work analysis, the most interesting solutions are characterized in the following way: Section 3.2.1 - avoidance solutions to circumvent adversary poisoning relay nodes [9, 12, 16, 57, 59, 61, 82]; Section 3.2.2 - circumvention of non-trustable Autonomous Systems [1, 15, 30, 31, 69, 70, 75]; Section 3.2.3 - avoidance of unsafe geographical regions and related GeoIP mappings [35, 37, 39, 61, 79]; and section 3.2.4 - Tor strengthening solutions improving the base Tor circuit's anonymity guarantees with K-anonymization[51, 71].

3.2.1 Circumvention of adversary-poisoning relay nodes

To mitigate the attempts of global adversaries launching poisoned-relay attacks, clients must avoid the use of unsafe relay nodes (controlled by adversaries) in established circuits. The more relevant techniques and solutions proposed in the literature can be summarized as follows:

- **Scan, enumerate and flag the possible relay nodes**. This approach is already used in Tor, in more recent client-enabled Tor software tools that try to decrease the probability of the selection of poisoned nodes. Upon detection of suspect relay behaviour, Tor uses a set of labels designated as control flags [57] which can be disseminated in a P2P fashion through the network. Tor maintainers run a service aimed at verifying the reports of possibly unsafe relays [82] and exit nodes [9, 59].

- **Selective setup from clients**. Tor allows for restrictions regarding the set of entry nodes used by clients in an attempt to mitigate guard rotation weakness [12, 16]. An unlucky client, however, could still select a malicious guard. If clients adopt the behaviour of randomly switching over different guards, using a set of candidate nodes, the chance of regaining anonymity improves. This solution alone is not effective when considering a near-global adversary, as demonstrated in [12]. In [16] the authors also demonstrated that Tor's exaggerated time-based guard rotation presented vulnerabilities to profiling attacks.

3.2.2 Circumvention of Non-Trustable Autonomous Systems

The techniques in 3.2.1 may not be effective when an adversary can observe large zones of the Tor network. A Global Adversary may not even need to gain control over specific relays to de-anonymize Tor traffic. It can just have eavesdropping capabilities on the inter-relay traffic that crosses the network within those large regions controlled by the adversary [30, 31]. To find countermeasures against these adversaries, researchers have proposed enhanced defensive approaches. Typically, such approaches attempt to help Tor clients' to choose paths away from the prying eyes of malicious ASes, by leveraging, for example, the analysis of the Internet topology boundaries and observations in inter-relay latencies [75]. Among the proposed solutions, we can summarize the following ones, as the more representative of these countermeasures:

AS-aware path-prediction

This approach is focused on predictive path selection algorithms for decreasing the chance of an AS-level attacker observing traffic flowing between Tor circuits' endpoints [1, 15, 69, 70].

AS-aware path mandates and diversity of Geo-IPs

In [15] it was suggested the addition of two new requirements to the Tor path selection algorithm: the use of mandates (with considered trust-nodes) and path-establishment using IPs with Geo-IP diversity - avoiding IPs in the same AS. It was observed that these approaches decreased the probability of an AS eavesdropping on both ends of a connection. However, they did not sufficiently mitigate the possibility for a malicious AS to perform traffic correlation (in different circuits). Some solutions related to these strategies are:

• **LAStor**[1]. LAStor is an AS-aware Tor client that can select safe paths between a client and a destination.

• **AS Relay Announcements**. In [70] the authors proposed that each relay must publish, as part of the Tor consensus document, the list of ASes it uses to reach a given relay or a destination. Clients could then use this information to establish safe circuits.

• **Astoria**, described in Starov's work [69], is an AS-aware variant of Tor, which avoids the use of vulnerable circuits using efficient network management based on load balancing circuits across secure paths. Being a god solution in the vision of how to avoid untrustworthy Autonomous Systems, we must mention that this solution alone, cannot resist active BGP interception attacks [70] and might become outdated with the dynamic routing management from AS Relays announcements.

3.2.3 Circumvention of unsafe geographical regions and related Geo-IPs

In the literature, this type of attack circumvention focuses on avoiding entire geographic regions altogether, looking at country-level granularity (and not necessarily at AS-level observation). The main motivation is to evade censorship policies from specific repressive governments. Currently, Tor allows users to select a set of countries to exclude from circuit selection [61], however, the path between safe nodes might still cross these excluded countries [39]. The solution for this purpose was initially proposed by different authors and experimental tools:

DeTor[39]

Uses a technique that proves a Tor circuit, even in between nodes, does not cross excluded regions. To provide the so-called provable geographic locations, DeTor authors borrow the idea of alibi routing(AR) [37] into Tor. Alibi routing is a routing strategy based on packets' round trip times(RTTs) and the speed of light, as a constant to prove that a given packet did not travel within forbidden regions. AR observations use single relays located outside a certain "forbidden region" to confirm that traffic is going from that relay to the destination. While the base AR strategy uses one single relay, DeTor generalizes the solution for three relays. Unfortunately, there are a few limitations regarding DeTor which make it a limited solution in providing provable geographical avoidance conditions:

- DeTor may also use IP geolocation services to find the exact location of Tor relays, not necessarily with further confirmation which hampers the accuracy of DeTor's provided measurements [35, 79].

- In high latency links, it is not possible to measure the packets' travel times accurately, solely relying on RTT measurements.

- Third, DeTor assumes only a symmetric routing approach which doesn't correspond with the real operation of ASes and Routing.

TrileraTor

This approach was proposed in [35], as a possible improvement to overcome DeTor's main limitations. The solution is based on a new measurement technique that derives a circuit's end-to-end timing from the handshake in Tor's circuit establishment process. To prevent the use of fake GeoIP information in its measurements, TrilateraTor leverages a distributed measurement infrastructure and protocol which obtains accurate estimates for the locations of relays.

3.2.4 Tor strengthening using K-Anonymization Input Circuits

As solutions designed for the mitigation of correlation-based traffic analysis on Tor, considering a global adversary model, some recent proposals address the possibility of exploring traffic disaggregation using multipath-based circuits. Those solutions have been studied in the context of K-anonymization, particularly focusing on Tor input circuits. The following proposals are aligned in this perspective:

TorK [51]

This system is a contribution focused in providing defensive countermeasures against global adversarial attacks. The essential approach in TorK is to strengthen the anonymity set associated with the source IP address of a given Tor circuit from one single user to a set of k plausible users. This idea is represented in the figure 3.1.

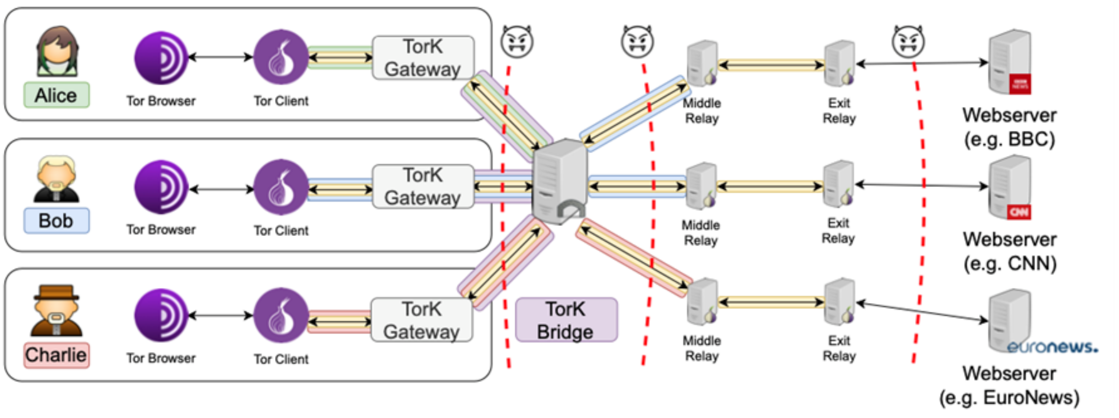

Figure 3.1: TorK architecture model with established k-anonymized circuits.

In the represented figure, we observe that the TorK network model includes two components: the TorK client and the TorK bridge. Both TorK components implement a specific pluggable transport and must be fully compatible with the existing Tor components (Tor bridges and input relays) and client-side software applications. For the TorK operation represented in figure 3.1, Alice leverages TorK to communicate with a given web server (represented on the right), through a standard Tor vanilla circuit. The access from the client to the entry node is proxied by the TorK bridge. TorK strengthening intends that when considering a global-level adversary that can observe all exchanged network packets, using standard vanilla Tor circuits, the adversary would be able to de-anonymize Alice's client by correlating the entry and exit flows. However, TorK prevents this attack by allowing k-1 additional users (Bob and Charlie) to collaborate with Alice so that the adversary will not be able to distinguish who amongst them is the real originator of the traffic associated with this circuit.

To achieve this, prior to the circuits' establishment, the k users open connections, called TorK segments, from their client to the bridge. Each segment acts as a covert tunnel between the local client and the bridge, such that the local client can send arbitrary traffic through it. However, while Alice's segment will be used to transmit Tor packets, the segments of the supporting users will transmit chaffing payloads. The bridge discards the chaff and only forwards Alice's actual traffic to the entry node. To prevent the global adversary from distinguishing which segment carries the Tor circuit's cells (achieved by observing differences in the volume and timing properties of the observed traffic), the traffic of all participating segments is encrypted and modulated according to a common traffic shaping function. Thus, even though the attacker can capture and inspect the traffic from each of the three users, it cannot correlate the message to the original sender due to indistinguishability between segments. He would have to randomly guess the sender having a 1/k probability of succeeding: the core idea behind K-anonymity condition. This is why TorK provides k-anonymity by allocating k-sized groups, introducing the term "k-anonymized circuit" to refer to a coordinated setup of k segments protecting Tor input circuits.

TIR Nodes [71]

In this proposal, a solution based on TorK was enhanced with the use of a more robust pre-staging solution in which several gateways (special Tor bridges called TIR nodes) implement overlaid TLS-enabled TorK segments, that can connect clients and these Tor bridges. These bridges can work in a stand-alone fashion or can be components coupled to Tor clients. Moreover, TIR nodes can forward client packets according to a multipath routing strategy involving K TIR nodes. The figure 3.2 represents a generic environment, with an operation of k TIR nodes.

The underlying support for the multi-staged circuits between TIR nodes is provided by a standard parametrizable S-TUNNEL solution [83], allowing all the TLS 1.2 or TLS 1.3 parametrizations and cypher suites to be used. Furthermore, the TIR nodes are also responsible for the traffic fragmentation of original client packets in parametrizable Tor-compliant cells. As in the TorK solution, TIR nodes also include chaffed traffic and are used to decouple the original input traffic in several flows which are sent to multiple Tor entry nodes, by collaborative proxied connections. So, even if the attacker observes all flows, it can only de-anonymize a set of k clients using the network.

The proposal in [71] involved the analysis of the effectiveness of the proposed solution, focused on processing observations of Tor input traffic and Tor-output traffic using powerful Machine Learning algorithms analysing packet-lengths, interleaving timing of packets or duration of traffic-sessions, as observable multi-features. The tool used to evaluate the correlation between chaffed and covert traffic is XGBoost, a distributed gradient library that implements a *gradient boosting* machine learning algorithm. After evaluating four scenarios, starting with a single TIR node and incrementing one for each iteration, the classifier demonstrated lower values of separability of flows when more TIR nodes were in use, as expected. It presented AUC values as low as 0.59 when four TIR nodes were used, really close to random guessing, proving the solution's effectiveness against this tool.

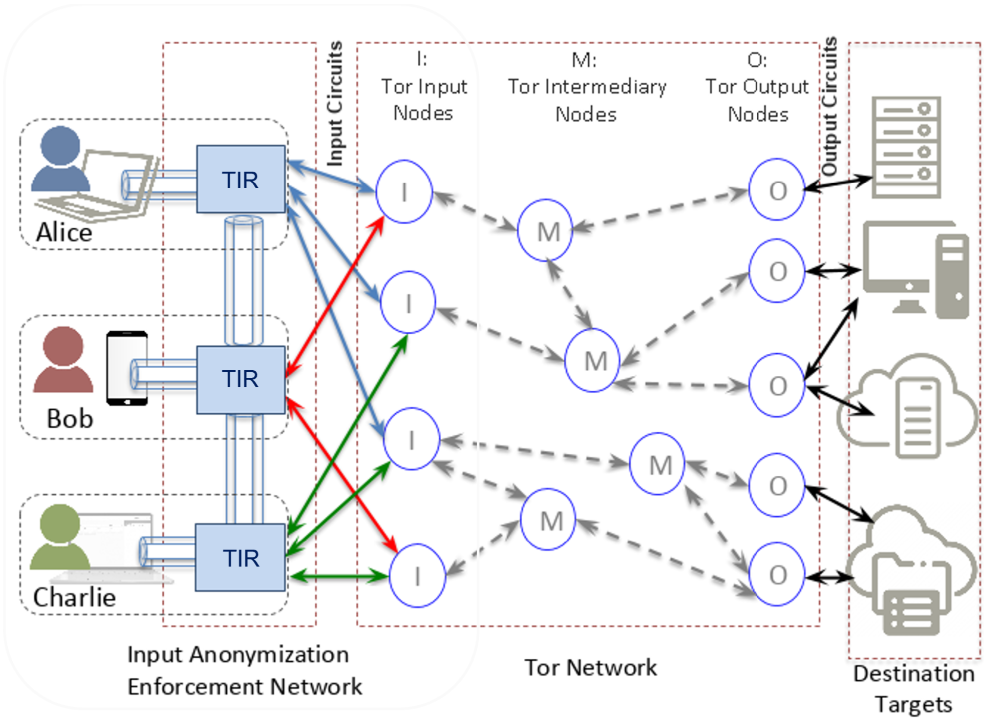

Figure 3.2: TIR nodes architecture model with established k-anonymized circuits.

Finally, an enhancement of the TIR solution is ongoing, providing more diverse support for the multi-staging TIR-based network using the K-anonymity TIR nodes. For this diversity, the solution will include the support of TIR to TIR circuits to use real-time steganographic media streaming covert channels [5]. In this case, the packets interchanged by the TIR nodes can be hidden as "covert" bits embedded in standard media traffic (such as Web RTC channels). For an adversary, these circuits will be undetectable (or unobservable) by using the studied attack typologies.

3.3 Watermarking Detection Methods and Tools

When performing watermark detection the methodologies are relatively straightforward. As mentioned in 3.1.3, the watermark only depends on the encoding type and diversity scheme used for the support carrier. As an approach to building a watermarking tool, the detector will collect the traffic's features and form a descriptor vector for the observed flow that is used. Such vector is then used as the starting element to compute the value of a function, which will decide whether a flow is watermarked or not. Watermark decoding algorithms may be distinguished into different families: non-blind algorithms [23, 24, 25] or blind algorithms [77]. Non-blinding means that the analysis is dependent from a known-carrier data watermarking encoding. Blind algorithms are those in which the analysis is independent of the carrier data provided by the watermarker. We will describe next the techniques and tools we found in the related-work analysis.

Advanced time-based watermarking tools. Some methods and tools implemented more advanced techniques for watermarking analysis. For example, in the Rainbow approach [25], the authors proposed a watermarking scheme that would be able to use delays, hundreds of times smaller than previous techniques resulting in inherent invisibility for detection and more robustness to interference. When the delays were very small, Rainbow [25] had to be non-blind – the packet IPD values extracted had to be compared with ones recorded at the watermarker. The watermark extraction in Rainbow used normalized correlation to account for network jitter and has a pre-processing step before it to render the system robust to packet addition/removal. Performed evaluations led the authors to conclude that the system was not only invisible to detection but also presented false error rates for short observations. Nevertheless, we noticed possible drawbacks, such as possible visibility conditions when comparing with more recent counter flow watermarking techniques [41, 44].

Scalable watermarking with resilience against packet-losses. Swirl [23]was proposed as a new watermarking scheme, more scalable, invisible and resilient to packet losses. Swirl's watermark pattern was specifically adaptable to the characteristics of each flow being marked. Similarly to Rainbow [25], Swirl also uses watermarked flows by introducing very small delays, with the same base invisibility to detection tools. However, there is an essential difference since Swirl proceeded to divide a flow into intervals, subintervals, and yet again into slots. Two intervals, the base and the mark were selected. The base interval was used to derive the necessary parameters for watermark embedding. According to parameters derived from the base and a specific watermark key, packets from the mark interval were delayed as to be permuted to specific slots, in said interval. Then, the detection consisted of analysing the base, extracting the necessary parameters, and checking if the packets in the mark followed the imposed watermarking distribution. The results using this approach showed an improved accuracy under ideal conditions. For relatively large flow lengths it is necessary to have some minutes to achieve ideal conditions. Unfortunately, other studies revealed this approach to be vulnerable to more recent counter-watermarking techniques [41, 44].

Inverse-Flow Watermarking. In a more recent contribution, the authors proposed the *Inflow* approach and related experimental tool [29]. The technique here is to identify Tor Hidden Services using an inverse flow watermarking strategy. Inflow follows a time-based diversity scheme and carrier. The rationale is to conduct an attack leveraging network congestion mechanisms, to induce Hidden Services' traffic patterns from a watermarker located client-side. In particular, TCP acknowledgement packets from the client to the service provider were dropped in short bursts to temporarily stop and resume packet transmissions from the provider, creating traffic gaps. In the evaluation methodology, the adversary controls both edges of the circuit and can detect the watermarking gaps. The watermark detector, located Hidden-Service-side, had previous knowledge of the estimated periodicity and duration of the gaps and analysed IPDs to identify and extract the specific watermark from the flow. With appropriate parameter tuning, the system was able to achieve 96% true positive and 0% false-positive rates when correlating flows. The authors also tested Inflow against known counterflow watermarking techniques (the ones described in [27]) concluding its successful robustness against such attacks.

Exploitation of intrinsic traffic congestion of the Tor protocol. This approach, addressed by the Duster technique [28], implements a traffic analysis attack through watermarking focused on de-anonymizing Tor Hidden Services. Following the author's description, the Duster's novelty was because: it exploits Tor's congestion protocol mechanism intrinsically; it was hidden from the target endpoint; and, did not affect network performance. In normal conditions, Tor uses SENDME cells as acknowledgements, signalling a Hidden Service that a client received a fixed amount of relay cells. Tor generates one SENDME every 50 cells and another every 100 cells – and then is ready for the next batch. Duster leveraged this mechanism and watermarked flows by sending a large batch of SENDME cells to the Service Provider while refraining from sending anymore before receiving all related data. In the approach, a watermark detector located Hidden Service side listened for the SENDME cell batch and silence pattern and was able to identify the flow as watermarked. For the Hidden Service not to notice the watermarking behaviour, the detector kept track of all data cells sent by it and forwarded the expected SENDME cells to clients, essentially hiding the watermark. With appropriate tuning, the authors found through experimental evaluation, the true positive rate(TPR) could be as high as 98% with false-positive rates(FPR) only 3%. In 2020, Tor set version 1, SENDME cells are used as the consensus default. This version introduced authenticated SENDME cells, which prevents this attack altogether.

3.3.1 Blind watermarking with deep-learning techniques

The use of blind watermarking techniques together with deep-learning algorithms is a more recent promising technique. We will analyse in this category of techniques and experimental observation tools, two of the more advanced state-of-art proposals: DeepCorr

[50] and Finn [67].

3.3.1.1 DeepCorr [50]

DeepCorr is a state-of-the-art correlation attack approach supported by deep learning. In the DeepCorr attack model, the core idea is to train a convolutional neural network(CNN) to learn the intrinsic signature of traffic in the Tor network – called the network's correlation function. In fact, the complex nature of noise in Tor and the unpredictability of natural (or eventually induced) perturbations in the network, were pointed out as reasons for the existing correlation attacks' inefficiency, in a real-world scenario. A CNN implementation was chosen since this type of neural network is known for having good performance on time series. In DeepCorr, the CNN is composed of two layers of convolution and three fully connected layers. It can be seen as a pipeline where inputted flow pairs get returned values of 0 and 1. Each flow is represented by four vectors – two for upstream and downstream inter-packet delays and two for upstream and downstream packet sizes – and the flow pair is represented by an eight-row matrix constructed from both flows' vectors. The output represents the probability of the two flows being correlated. The first convolution layer is intended to capture the similarities between adjacent rows of the previously mentioned eight-row matrix while the second has the goal of capturing overall traffic features from the combination of timing and size information. To train the CNN, the authors presented it with two large, labelled, sets of flow pairs, where one had correlated flows and the other did not. For this tool's experimental validation [50], the authors performed evaluations using flows collected by accessing the top 50000 Alexa websites, via Tor. In the evaluation, half of the collected flows were used for training purposes and the other half for testing. From each flow, only the first 300 packets were used, applying padding to shorter flows. By observing the system's performance for a month, the authors concluded that retraining would be necessary approximately every three weeks. In comparison to previous presented state-of-the-art systems, such as Raptor [70], DeepCorr was able to achieve much better accuracy for long observations and significantly better accuracy for shorter observations. With only 900 packets, the system achieved 96% accuracy (comparing for example with 4% achieved with Raptor. While Raptor was able to present 96% accuracy with 100MB of data, DeepCorr was able to reach 100% accuracy, but with only 3MB. Regarding processing time, DeepCorr was roughly two times slower than Raptor for same size observations. However, since DeepCorr needed significantly smaller observation times to achieve the same level of accuracy as previous systems, it ended up being much faster than the others. It is still necessary to conduct more observations and analyses to understand the necessary criteria for the periodic retaining of the CNN.

3.3.1.2 Finn [67]

Finn was presented as an enhancement of the DeepCorr system [50]. Following a time-based diversity scheme and carrier, Finn's watermarks get embedded via packet delays in target flows. However, the new insight introduced is that it leverages neural networks to avoid a "manual" process for embedding and extracting the watermarks. Finn's watermarker is made up of a neural network with four hidden layers. As input, it takes the watermark to be embedded (a vector of all zeros and a single one) and the network noise at the target flow. The output is a vector of delays to be added to the flow's packets whose size is equal to the total number of packets in the flow. The generated delays take into account the network noise expected to be introduced to the flow, such as to prevent it from damaging the watermark. The decoder consists of a network of two convolution layers and two fully connected ones. As input, it takes the noisy and watermarked IPDs extracted from the received flow. The two convolution layers extract the encoded noise as well as watermarked one. The fully connected layers extract the watermark itself by returning a vector of probabilities of the same size as the number of bits in the watermark. Since the watermark is made up of a single one, the highest probability taken reveals the one's "location".

The model is trained with data regarding different IPDs, watermarks, watermark delays and network noise. The evaluation conducted on the live network, explained in [67], allowed the authors to determine that, after proper parameter tuning, the system was able to achieve TPR between 93% and 97% and an FPR as low as 1%. If we compare the results with those obtained in the experimental evaluation of DeepCorr, it is noticeable that Finn has slightly better results with much fewer data (50 packets compared to DeepCorr requiring 300). The conducted evaluation also showed that Finn is also robust against real-time noise and embeds a watermark extremely difficult to detect. In the state-of-the-art, Finn seems to be the more advanced technique and tool, obtaining very high accuracy rates with a small data volume, while exhibiting considerable robustness and invisibility. With this approach, it is expected that jitter fluctuations negatively impact the system's performance and retraining is recommended when jitter conditions change.

3.4 Summary and Critical analysis

This chapter covers different dimensions of the related work and literature, around the Thesis goals. We report different Tor de-anonymization attacks and traffic correlation techniques (section 3.1), in a taxonomy that covers different attack types, with particular attention to de-anonymization approaches based on traffic-correlation and watermarking techniques and tools (particularly addressed in section 3.1.3). Among these attacks, more particularly related to the thesis goal, we noticed the most promising and effective state-of-art techniques (described in section 3.3) with tools based on deep learning of traffic properties [50, 67] that can be applied to Tor-traffic de-anonymization goals, potentially exploitable by global adversaries. These techniques, particularly the ones using convolutional neural networks [50, 67], are inspiring solutions to build better traffic de-anonymization tools. Moreover, these tools can be used in scenarios of traffic analysis conducted by state-level or cooperative global-level adversaries focused on the de-anonymization of users and Tor onion-protected services, during sessions in which such users interact with the targeted services. However, such techniques can have issues dealing with the trade-off on the efficiency and the effectiveness of the detection ability, for real-time and high volumes of inspected traffic flows from multiple probing locations. Furthermore, these discussed state-of-the-art techniques and tools will be particularly interesting when analysing ways to build traffic correlation tools deployed in test-bench environments, as a way to evaluate some recently proposed solutions for Tor strengthening, used as countermeasures against traffic correlation attacks (as the ones addressed in section 3.2.4). Among these, we consider solutions with pre-staged circuits that decouple users from Tor Bridges or Tor input relay nodes, using K-anonymity covert circuits [51, 71].

4 TorMarker: System model and architecture

The following chapter gives a conceptual description of TorMarker's architecture and how its different components operate.

It's divided into 5 different sections: section 4.1 contains the tool's macro architecture and its modules; section 4.2 describes the threat model; section 4.3 covers all of TorMarker's components in a more in-depth design description; section 4.4 describes the tool's operation in a TIR environment; section 4.5 is a summary of the chapter.

4.1 Architecture overview

TorMarker is divided into 2 modules:

Watermarking module

This module's purpose is to manipulate traffic from targeted connections, inducing a temporal watermark, while also acting as the client that opens the connection. To achieve this, the client's data is sent by this module, which has previously crafted all of the artificial delays that compose the watermark, embedding the mark into the data as it is injected into the network.

Detection module

The detection module is responsible for detecting a watermark in a given egress Tor flow. The detection is done through a Convolutional Neural Network(CNN), trained to detect the intrinsic time patterns of the watermarks, which outputs the probability of a flow being watermarked or not.

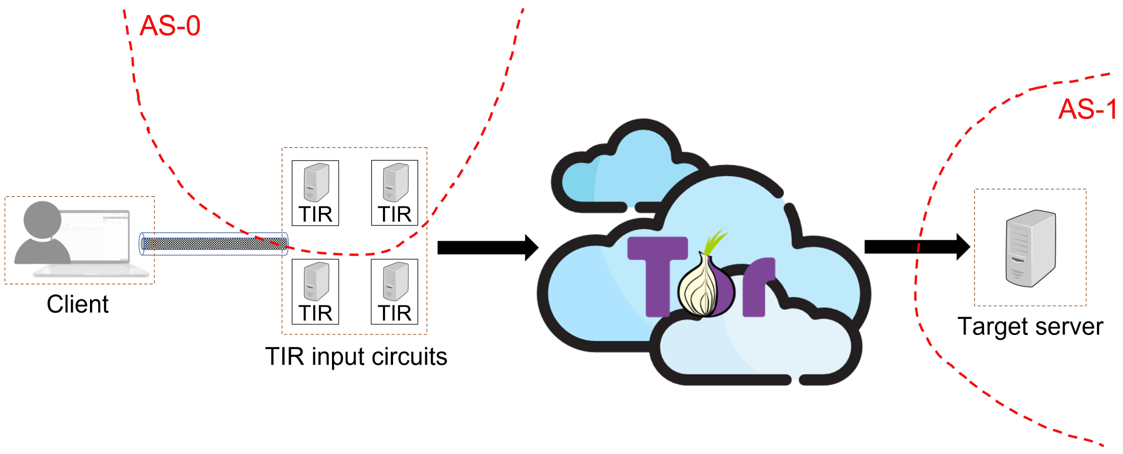

Figure 4.1: Threat model of the thesis' case study.

4.2 Threat model

Figure 4.1 shows the setting studied in this dissertation. The **Client**, which is run by the Watermarking component, has an open connection with the **Custom server**. This connection is established over Tor and is supported by geographically distributed TIR input circuits. Since this is a private, anonymous and low latency connection, an adversary can only use extracted traffic features to employ an attack, payload comparisons are not possible due to all data being encrypted.

For this case study, the adversary aims to perform a de-anonymization attack on the connection by processing egress Tor flows arriving at the custom server. The type of adversary we study in this context has an AS-level influence, meaning it has privileged control or authority over different autonomous systems to listen and alter connections that travel through these zones. Depending on the type of evaluation, covered in chapter 6, the attacker's privilege level and influence over the network change, according to the scenario we wish to analyse.

Nonetheless, figure 4.1 shows an overall representation of the adversary's threat model, where the controlled ASes are highlighted in red:

- **AS-0** covers the client that the adversary attempts to de-anonymize and has partial influence over the TIR input circuits. For this dissertation, we study a strong adversary which may have enough influence over the TIR input circuits to aggregate all traffic sent from an arbitrary connection, while having access to its whole packet sequence. Furthermore, the attacker may also be able to implant software in the victim's machine which marks sent traffic, or just have the option to mark traffic travelling through a certain TIR segment.

- **AS-1** covers the other end of the connection resulting in all data arriving at the target server being observed by the adversary.

This attack can be characterized as an active de-anonymization attack. Connections which pass through the attacker's sphere of influence are eavesdropped on and manipulated to carry certain patterns of watermarks. In a real setting, the marking of the traffic can be employed in any segment of the Tir input circuits that fall inside the attacker's sphere of influence but in this instance, traffic is injected into the network already marked.

4.3 Components and their design

This section provides an in-depth description of the modules' architecture and the reasoning behind some architectural choices.

4.3.1 Watermarking module

This module's architecture is divided into 2 sub-modules:

- The first one is the *.pcap* reader. To simulate a real user connecting through an application we used a dataset [36] of real Tor flows, stored in several *.pcap* files. This sub-module reads the chosen dataset's *.pcap* file iteratively, sending the packets and the original inter-packet delay information to the other sub-module until the file ends or the desired number of packets to be sent is reached.

- The other sub-module can be defined as the sender. Naturally, it is responsible for sending each packet received by the sender to the chosen TIR node. Before transmitting each packet, the component sleeps a certain amount of milliseconds, more specifically, it waits for the computed artificial delay, that makes up the watermark, plus the original IPD, related to the last packet sent.

4.3.2 Correlation module

The correlation module is composed of a convolutional neural network(CNN) which is how the tool computes the probability of a certain flow being watermarked. The inspiration for the model of this neural network is our implementation of Finn's [67] CNN, a state-of-the-art flow correlation tool with high true positive and low false positive rates.

The goal of this neural network is to compute if a certain flow is marked by processing its packet IPDs. This is possible due to the CNN being trained to distinguish the natural noise of the Tor network and the artificial delays introduced by the TIR input circuits, from the added delays which carry the watermark. After the flows are captured, the IPDs need to be calculated from the arrival timestamps of each two packets and stored as an array of values according to the CNN's input structure.

The CNN's model is described in the table 4.1 which specifies the layer type and optimized parameters. This model was based on the network used in Finn [67] but it

Layer	Details
Convolution Layer 1	# of Filters : 500, Kernel size: (1, 10), Stride: (1,1), Activation: Relu
Max Pool 1	Window size: (1,5), Stride: (1,1)
Convolution Layer 2	# of Filters: 100, Kernel size: (1, 10), Stride: (1,1), Activation: Relu
Max Pool 2	Window size: (1,5), Stride: (1,1)
Fully connected 1	Size: 3000, Activation: Relu
Fully connected 2	Size: 800, Activation: Relu
Fully connected 3	Size: 100, Activation: Relu
Output Layer	Size: 1, Activation: Sigmoid

Table 4.1: Model of the correlation module's neural network.

is optimized to our implemented watermarks and data type. It features 2 convolutional layers and 3 fully connected layers:

- The first layer contains a kernel with a size of (1,10) and 500 filters. It is trained to find patterns from up to 10 consecutive IPDs. Its stride is (1,1) to cover all combinations of 10 successive arrival timings. This layer's purpose is to find correlations between IPDs of successive packets which mind indicate the presence of a watermark.

- The second layer has a size of (1,10) and 100 filters. It is aimed at capturing more complex functions from the combination of the delays processed in the first layer.

- The final 3 fully connected layers process the output from the final convolutional layer and give us a number between 0.0 and 1.0 which translates to the probability of the processed flow being watermarked or not.

The values for the parameters specified in 4.1, such as the number of kernels or kernel size, were optimized during the implementation phase and is further discussed in chapter 5, as well the network training methodology.

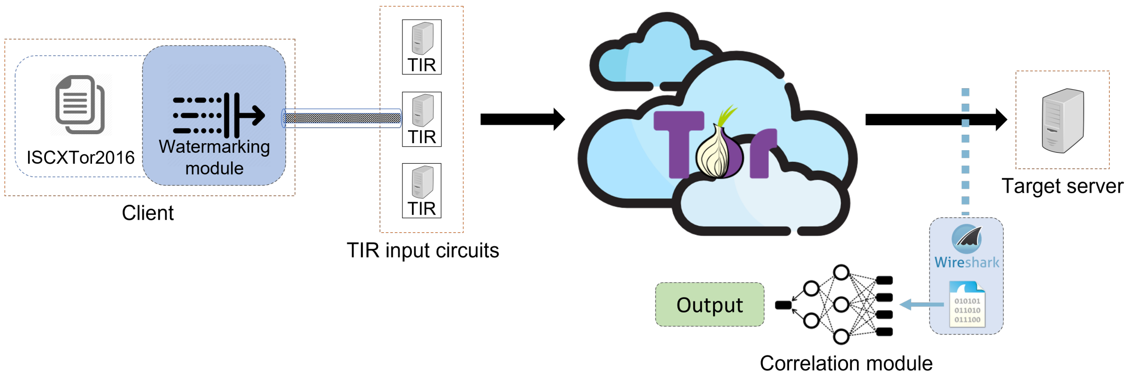

Figure 4.2: Simple model of TorMarker's operation.

4.4 TorMarker's purpose and operation

TorMarker's objective lies in de-anonymizing a Tor connection between an unknown user and server, supported by anonymous input circuits provided by TIR [72]. Our tool's operation, regarding the steps taken when attempting to de-anonymize the used endpoints of a targeted connection, begins by watermarking traffic leaving one edge of the connection, embedding a specific watermark previously chosen. TorMarker then processes flows captured at the other connection's endpoint in an effort to detect this specific watermark. In case of marked traffic being detected, the adversary has successfully linked the flows from both endpoints and de-anonymized the targetted connection.

To successfully employ this attack, the Watermarking module is set up to open a connection to a specified destination. The traffic that will be sent needs to be available as a *.pcap* file. At the other end of the connection, traffic destined to the specified destination is captured as a *.pcap* file. To evaluate if the traffic is watermarked, contiguous segments of captured traffic, or flows, are fed to the Detection module which returns a percentage of how probable the processed flows carry a watermark.

Figure 4.2 shows an overview of TorMarker's operation. The **Client**, which is run by the Watermarking component, opens a connection to the **Target server** using the TIR input circuits. Data exchanged in this connection goes through the TIR input circuits, into the Tor network, where it follows the relay circuit until it reaches its final destination.

4.4.1 Watermarker operation

In this thesis, to simulate an active attacker which can interrupt and manipulate traffic in its sphere of influence, the client itself is be the watermarker simulating malicious software operating in the target's machine. Positioned at the client's end of the connection, as seen in figure 4.2. This component injects the *ISCXTor2016* [36] dataset, as the client's data. We chose this dataset because it was publicly available for researchers and the data was generated by capturing real outgoing Tor traffic from the user's location, while using several applications (Web Browsing, Email, Chat, Streaming, File Transfer, VoIP, P2P). It is used in several works and its diversity and size are representative of real-world traffic, making it the perfect candidate for generating artificial Tor flows.

Since this module acts as the client, it opens the connection to the custom server. After choosing the *.pcap* file which contains the connection's data and selecting the watermark parametrizations, this module will start acting as the user's client, sending data to the defined destination.

TorMarker can embed two different types of time-based watermarks and allows for different amplitude values to be chosen. They are embedded while injecting the connection's data into the network by adding specific timing delays between certain packets before sending them. The two implemented watermarking strategies are:

- RAINBOW, a strategy which generates the mark using cumulative delays based on Houmansadr's work [25]
- ICBW, an interval centroid-based watermark inspired from another Houmansadr's work [24]

The implementation of these strategies is described in greater detail in section 5.2.1 in the next chapter.

Detection operation

As mentioned before, this component takes a captured network flow as input and outputs the probability of such flow being watermarked.

Naturally, the convolutional neural network needs to be trained to be able to compute the probability. So, for this purpose, several watermarked flows and regular network flows must be captured by a traffic analysis tool and stored as *.pcap* files. Flows used for training consist of egress Tor traffic, where in this study they correspond to traffic arriving at the mentioned target server. These flows are stored together, labelled as regular or watermarked flows, and randomized to then be used for training the CNN. Since we use different watermarking strategies, for the component to be able to process flows marked with a certain watermark type it needs to be trained beforehand using flows marked the same way, following a sort of semi-blind watermarking approach.

After training, the component is able to process network flows and output if it is marked or not.

4.5 Summary

Chapter 4 presented the inner workings of TorMarker while also specifying the different components' operation and the tool's operational context. We begin by describing the tool's overall architecture in section 4.1, glancing over the two tool's modules. This is followed by a conceptual description of the tool's deployment conditions (section 4.2), where we analyse the properties of the environment and the type of privileged influence an adversary should have when deploying TorMarker. We then went over the two components of TorMarker (section 4.3, providing an in-depth description of the watermarking component and the CNN that makes up the correlation component. Finally, in section 4.4, we glanced over the tool's operation, the different watermarking parametrizations and the necessary training process.

5 Prototype implementation and environment setup

This chapter addresses the process and specifications behind the implementation of TorMarker, focusing on the modules conceptually described in chapter 4. It also describes all of the necessary adaptations we had to make to the original TIR prototype when building our test-bench environment. The prototype's source code is available publicly in `https://github.com/MiguelSHorta/TorMarker`.

It is organized in 4 sections: section 5.1 describes the TIR environment and all the adaptations made; sections 5.2 and 5.3 specify all implementation details of its related module. It finishes with section 5.4 where we provide a global summary of the chapter.

5.1 TIR environment

Since this dissertation is aimed at evaluating the effectiveness of the TIR input circuits, we followed as much as possible the implementation instructions specified in Teixeira's work [72]. However, to properly evaluate the TIR input circuits as a more secure solution when compared to vanilla Tor, we had to make some small changes to the inner workings of TIR whilst maintaining the solution's overall behaviour and watermarking countermeasures.

5.1.1 TIR nodes

Regarding the setup of the TIR nodes, we had an issue caused by the type of traffic which is accepted by this solution. Teixeira's implementation of TIR could only deal with REST requests where a new SOCKS connection would be opened with the specified Tor endpoint, for each request. This meant that only web browsing-type applications were supported by the TIR input circuits and made it impossible to inject the *ISCXTor2016* [36] dataset into the network as the user's data. To be able to use the dataset, which allows us to study how our tool behaves when dealing with data from different types of applications, we reprogrammed the TIR nodes. The changes performed made sure that TIR didn't re-open TCP and SOCKS endpoints for each new request. Instead, the nodes maintain a single TCP connection to the client and a single SOCKS connection to the specified Tor endpoint which stays alive as long as the communication is active. This guarantees that we are able to freely stream traffic through the open connection while not forcing new SOCKS connections, which would stress the Tor application and cause malfunctions.

5.1.2 Stunnel

Stunnel tunnels a connection through the TLS protocol so traffic becomes encrypted and less distinguishable to a possible attacker. It works as a connection proxy and allows for a wide variety of TLS parametrizations.

In the original TIR setup, Stunnels are used to tunnel the connection segments in between the TIR input circuits multipath network, however, they were not used in this test-bench environment. Since we are using the *ISCXTor2016* [36] dataset, all artificially generated Tor flows were already following the TLS protocol which makes the installation of Stunnels not needed.

5.2 Watermarking module

As said in chapter 4, this module acts as the client which has an open connection to the target server that the adversary attempts to de-anonymize. Since this module needs to read the packets sent as user data and re-send them according to certain crafted timings, we opted for a Java implementation (Java 17.0.1). This allowed us to have control over packet payloads and packet arrival timings while maintaining the system's complexity lower.

Figure 5.1 shows a deeper look into this module's design:

- Regarding the data we used to simulate a real connection, the **ISCXTor2016** data was divided into several *.pcap* representing flows from different applications. Since we were working with small-scale timing intervals, we needed to reduce as much as possible the overload from reading the *.pcap* files. To facilitate this process, we generated an associated *.csv* file which contained the lengths and arrival timings of all packets and allowed us to access these packet arrival timings in a quicker and simpler way.

- The **PcapManager** class is responsible for reading the *.pcap* and *.csv*, passing the packet payloads and timings to the **Watermarker** class. We used the library Pcap4J [54] to open and inspect the *.pcap* files and decided to use opencsv [52] to read the *.csv* files since it is one of the fastest alternatives and allows for an easy implementation. The watermark timings are also crafted in this class, right after opening *.pcap* and *.csv* files.

- The **Watermarker** is the main class of the system. It is responsible for sending the data through a *Java.net.Socket* according to the watermark-embedded timings. All network configurations are read from a file called *config.properties*, present in the same directory as the rest of the system.

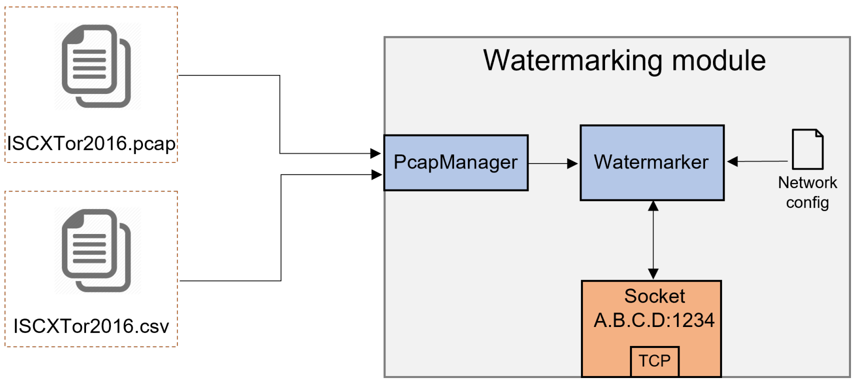

Figure 5.1: Watermarking module's design.

5.2.1 Watermark crafting

As mentioned before, TorMarker allows the traffic to be marked in two different ways. The following 2 sections describe how we implemented these watermarking strategies in more detail.

RAINBOW

This watermarking strategy is the implementation of [25], a robust and invisible watermark embedded through cumulatively adding or subtracting a chosen delay value.

Suppose we have n IPDs from unwatermarked traffic where the ith packet IPD is represented by $\tau_i^u = t_i^u - t_{i-1}^u$, t_i being the arrival timing of the ith packet (we use u to denote unwatermarked traffic and w as watermarked). To embed this watermark, each ith packet suffers a delay by the amount w_i such that $\tau_i^w = \tau_i^u + w_i$.

To calculate $w_i = W_0 + a_i$ we first generate the a_i calculated as $a_i = a_{i-1} \pm a$ for each packet where $\pm a$ will take the values $+a$ or $-a$ with the same probability. The *amplitude*, a, is a positive parameter of the watermark chosen beforehand. Since it is not possible to delay a packet for a negative amount of time, W_0 or *max amplitude* is another parameter chosen beforehand which is always bigger than the minimum value obtained from the sequence of a_i. W_0 is added as a delay to all sent packets so that w_i is never a negative value. The *max amplitude* value controls the overall intensity of the mark and the *amplitude* controls the granularity of how the mark is applied.

Figure 5.2 shows the pattern formed by the artificial delays of the Rainbow watermark.

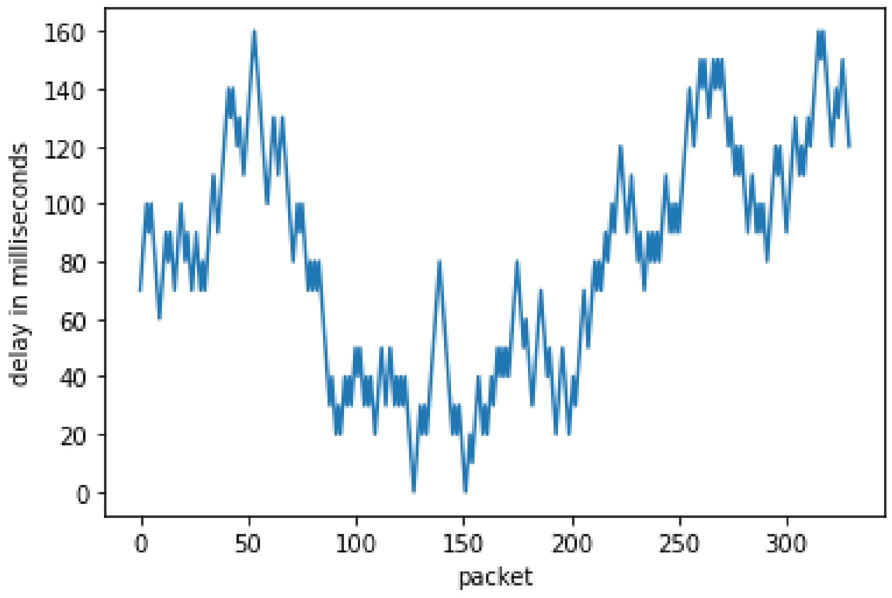

Figure 5.2: Plot of the artificial delays from the Rainbow watermark.

ICBW

The ICBW watermark was implemented based on [24]. It is presented as a variation of an interval centroid-based watermark resistant to multi-flow attacks and a non-blind approach which allowed an arbitrary number of bits to be encoded, however, we only need to make the watermarked traffic distinguishable from regular traffic so the implementation can be simplified.

We begin by looking at the original IPDs of the packets which will be sent. This stream of packets is divided into n intervals of T milliseconds, without overlapping each other. The *amplitude*, represented by a, is chosen as a parameter and added as a delay to all packets which fall inside these intervals. The result is a distinct pattern which is shown in figure 5.3

5.3 Correlation module

All parts of this module are implemented in *Python 3.6.13*. The framework we used to implement the neural network is *Keras 2.6.0* [32] which is an interface for *TensorFlow 2.6.2* [73]. We used a single machine running Windows 10 for both training and testing. Since TensorFlow has advanced support for CUDA-enabled GPUs, we set up a Nvidea RTX 2070 to run all network-related computations, substantially improving training and prediction times.

To implement the CNN's model we used the layer interface of Keras which, in comparison to TensorFlow's, it allows for a simpler implementation with much better code readability.

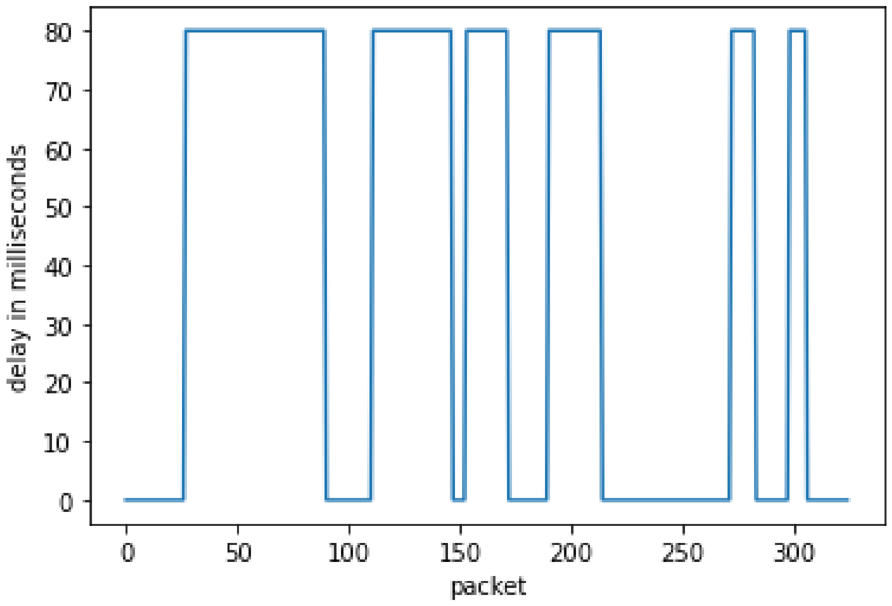

Figure 5.3: Plot of the artificial delays from the ICBW watermark.

5.3.1 Training

The network is trained according to the watermark it will detect and the amplitude used when crafting it. It requires 2 types of traffic flows:

- Regular traffic flows: sequences of packets from regular network behaviour which in this case will be data from the chosen dataset.
- Watermarked flows: traffic which is already marked with the same watermark the tool is attempting to detect.

The training dataset has the same proportion of regular and watermarked flows.
In table 5.1 are specified the main training parameters:

- The *optimizer* is the algorithm which updates the network's weights proportionally to the learning rate to minimize the model's loss. Our model uses the same optimizer used by Finn: Adam [34], a gradient descent-based algorithm well known for its use in neural networks due to its efficiency and scalability.
- The *loss function* is the function used to calculate the model's error regarding the predicted and expected results after processing the training dataset. Since our model classifies data in two possible classes, regular flow or watermarked flow, the only suited loss function would be binary cross entropy.
- The *Learning rate* is the rate of how much the network's weights change in each update and the *Epochs* value is the number of times the model processes the whole training set. We used values seen in works like Finn and DeepCorr as a starting point and fine-tuned them to the data used.

Parameter	**Values**
Optimizer	Adam
Loss Function	Binary cross entropy
Learning rate	0.0001
Epochs	200

Table 5.1: Training parameters.

Number of layers:	**FPR**	**TPR**	**ACC**
1 Layer (size 100)	0,4079	0,6593	0,6266
2 Layers (sizes 800,100)	0,4378	0,8312	0,6927
3 Layers (size 3000,800,100)	0,3555	0,8145	0,7250

Table 5.2: Validation set results when performing fully-connected layer optimization.

5.3.2 Parameter optimization

To fine-tune the network's hyperparameters we trained the network with generated samples of regular traffic and traffic watermarked with different watermarking strategies however values shown in the following tables are taken from processing flows embedded with an early-stage RAINBOW watermark.

Our aim was to create a precise tool that provides accurate but most importantly reliable results. To maximize the tool's reliability, when optimizing the CNN we focused on having a lower false positive rate(FPR) and then a higher overall accuracy and true positive rate(TPR).

Using Finn's kernel size and the number of filters as a starting point, we worked through several models tweaking the hyperparameters according to the results obtained for the validation set.

Firstly, we optimized the amount of fully connected layers the model should finish with. As shown in table 5.2, the network displayed the lowest FPR and highest accuracy when using 3 fully connected layers. The size of each layer was inspired by DeepCorr's model [50] because, in the original work, the fully connected part of the model shares the same purpose as this one.

We then compared results when changing the number of filters used by the 2 initial convolutional layers obtaining table 5.3. It shows better overall values when using 500 and 100 filters in the first and second convolutional layers, respectively. Higher values caused the model to overfit the training set.

We also experimented with different Kernel sizes, table 5.4. The results didn't vary significantly although the original sizes for Finn's model ended up working the best overall.

This optimization phase resulted in the current refined model of TorMarker's neural network shown in table 4.1.

Number of filters	**FPR**	**TPR**	**ACC**
1st layer: 200, 2nd layer: 50	0,1530	0,8474	0,8393
1st layer: 500, 2nd layer: 100	0,0816	0,8703	0,8937
1st layer: 600, 2nd layer: 150	0,1016	0,8764	0,8906

Table 5.3: Validation set results when optimizing the filter numbers of the convolutional layers.

Kernel sizes	**FPR**	**TPR**	**ACC**
1st kernel: (1,20), 2nd layer: (1,20)	0,1770	0,7758	0,7969
1st kernel: (1,10), 2nd layer: (1,10)	0,0816	0,8703	0,8937
1st kernel: (1,10), 2nd layer: (1,5)	0,0875	0,9240	0,9203
1st kernel: (1,5), 2nd layer: (1,5)	0,0951	0,8919	0,8984

Table 5.4: Validation set results when optimizing kernel sizes of the convolutional layers.

5.4 Summary

In chapter 5 we have described the overall implementation process behind the creation of TorMarker, as well as some adaptations that had to be the TIR prototype. We began by going over the changes regarding the TIR prototype's operation and the covert channels' parametrizations (section 5.1). We then started specifying the implementation process behind the prototypes' modules, first going over the Watermarking module's used technologies, the system's internal architecture and the logic behind the watermark types (section 5.1). In the final section 5.3, we specified the CNN's layers and their parametrizations, the training process of it, and finishing with the layers and hyperparameters optimization.

6

Experimental evaluation

In order to validate our tool's effectiveness, we needed to perform extensive experimental evaluations simulating real-world scenarios to obtain results that would mirror a real deployment. This chapter will cover all evaluations performed on our tool and present the obtained effectiveness and performance results.

This chapter will first cover the evaluation methodology in section 6.1 where we specify the different scenarios in which TorMarker was deployed and evaluated. Sections 6.2, 6.3 and 6.4 describe the results of TorMarker's evaluation regarding it's effectiveness and reliability when deployed in different conditions. Section 6.5 presents the tool's time performance evaluation taking into account the size and amount of packets used for training and testing. Then we have section 6.6 which sums up the main findings discovered in the previous sections. This is followed by a comparative analysis of TorMarker and the most influential state-of-the-art tools for flow correlation, in section 6.7. To conclude the chapter, we have section 6.8 which presents an overall summary.

6.1 Evaluation methodology

As mentioned before, we aimed to build a tool which can confidently and precisely identify timing-based watermarks carried by traffic flows within a reasonable processing time. To assess TorMarker's: effectiveness when embedding different watermark types in Tor traces; accuracy regarding the detection of said watermarks' presence; and performance of the prototype; we conducted extensive evaluations on our tool by deploying it in a controlled environment where we ran the tool through different scenarios emulating real-world cases.

6.1.1 Environment and scenario specification

The environment was composed of 4 machines in total. The main machine was running a Nvidea RTX 2070 with 8GB VRAM, an AMD Ryzen 5 CPU with 6 cores at 3.80 GHz and 16GB of RAM. This machine ran both components of the tool, was situated in Lisbon and had access to a bandwidth of 60Mbps. The other 3 machines had Intel CPUs with

8 cores at 2.40 GHz, 32GB of RAM and a bandwidth of 2Gbps. Of these 3, the one situated in London UK hosted the target server for all connections and the other, situated in Strasbourg FR and Gravelines FR, hosted the TIR nodes.

To fully evaluate TorMarker's precision and effectiveness, we came up with 3 main scenarios with a few ramifications that allowed for the tool to be studied under varying conditions regarding the attacker's strength, the distance of the connection and the path followed by the traffic.

Tor vanilla scenario evaluation

For this scenario, as shown in 6.1, the client opens a SOCKS connection through the Tor network to the target server without any support from TIR input circuits. It serves as a control scenario so we know the precision of the tool when deployed in an environment without any countermeasures against watermarking, other than Tor's natural jitter. Other than serving as a base comparison for tougher scenarios, the *Tor vanilla* scenario evaluation also provides results which can be compared to other state-of-the-art flow correlation tools that focus on de-anonymizing *vanilla* Tor endpoints, providing further analysis regarding TorMarker's validation.

This scenario simulates a type of adversary which targets a *vanilla* Tor connection and has enough network influence to listen and manipulate segments of both ends of the connection. The test set has the same amount of regular flows as watermarked flows.

TIR scenario evaluation

For this second scenario, the connection is supported by TIR input circuits where the client connects to a TIR node which relays traffic to the Tor network destined to the target server. In this case, in addition to Tor's jitter, TIR has countermeasures to watermarking and traffic has to follow a bigger path with one extra *stepping stone*. This countermeasure, implemented in all TIR nodes, consists of storing packet arrival timings observed earlier in the same connection and adding them in a randomized fashion, as additional artificial delays.

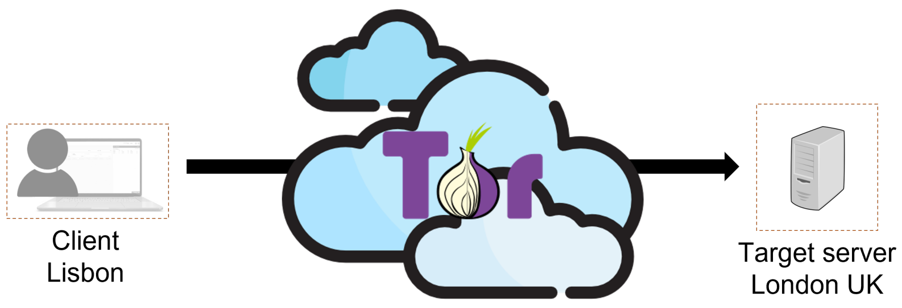

Figure 6.1: Model of the connection used in *Tor vanilla* scenario.

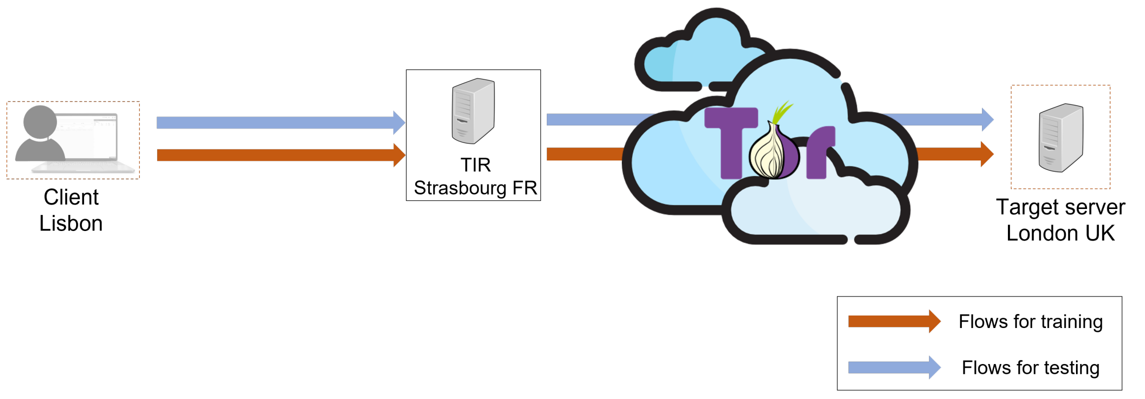

Figure 6.2: Model of the connection used in the *TIR* sub-scenario 1.

This scenario simulates a real-world attack employed by an adversary which is strong enough to aggregate the traffic partitioned by the different input circuits. This means that all flows from the victim's connection are watermarked even if sent through different TIR segments. For this evaluation, much like the first one, we want to assess the tool's effectiveness so the test set is equally populated with regular and watermarked flows.

In light of other works that perform evaluations regarding their tool's adaptability when correlating flows from different Tor circuits [50], for this scenario, we performed two types of evaluations on TorMarker.

- **Sub-scenario 1** - as shown in figure 6.2, the traffic flows for training (represented in orange) and testing (represented in blue) of regular and watermarked traffic were captured from a connection using the TIR node in Strasbourg. This helps the performance of watermark detection since all flows go through the same Tor circuits and the same TIR entry node and should follow similar jitter conditions.

- **Sub-scenario 2** - as shown in figure 6.3, the training flows (represented in orange) are generated from a connection supported by the Strasbourg TIR node and the testing flows (represented in blue) from a connection using the Gravelines TIR node. In comparison to the first one, this scenario should produce closer results to the real-world deployment of our tool because the processed flows are generated from diverse locations and circuits, which varies the jitter conditions they are exposed to.

TIR unaggregated flows evaluation

After assessing the tool's effectiveness in the first two scenarios, we now change the focus of our evaluations to how the tool's reliability and usefulness scales with different numbers of TIR nodes. The best watermarking parametrizations will be used to test how the tool performs when de-anonymizing a connection supported with K input circuits.

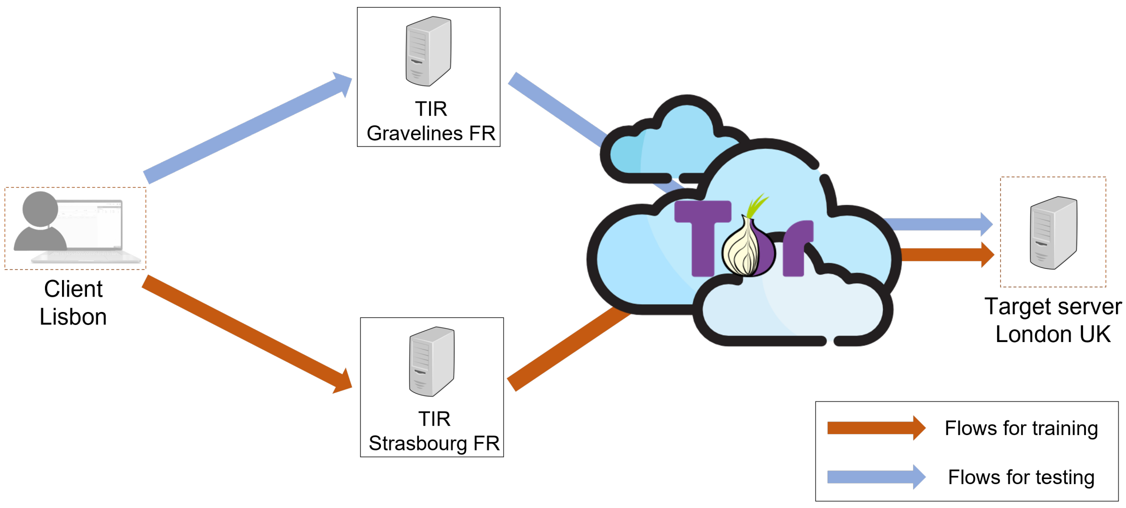

Figure 6.3: Model of the connection used in the *TIR* sub-scenario 2.

The adversary simulated by this evaluation is unable to aggregate the flows being partitioned by the several input circuits. It is only able to watermark flows from one segment of an entry node while listening to the partitioned connection as it arrives in the target server.

Following TIR's original evaluations [72], TorMarker will perform predictions from 4 different test sets where each one corresponds to a setup with 1, 2, 3 and 4 TIR nodes. Figure 6.4 shows a representation of this scenario when TorMarker is deployed against a 3 TIR setup. Since the prototype swaps entry nodes in a constant time interval, we will assume that each TIR processes the same amount of flows, resulting in a connection partitioned equally through the input nodes. This behaviour is reflected in the test set, the ratio of regular to watermarked flows varies accordingly to the amount o TIR nodes used because the adversary is only able to watermark the flows that pass through one of the entry TIR nodes.

The 4 test sets contained a total of 100 flows each, divided as such:

- **1 TIR setup** - all 100 flows are watermarked.
- **2 TIR setup** - 50 flows are watermarked and the other 50 are regular flows.
- **3 TIR setup** - 34 watermarked flows and 66 regular flows.
- **4 TIR setup** - 25 watermarked flows and 75 regular flows.

6.1.2 Methodology and metrics

Watermark variations and flow size

TorMarker was deployed in all the aforementioned scenarios and evaluated according to its effectiveness. To better understand how the tool's predictions vary proportionally to

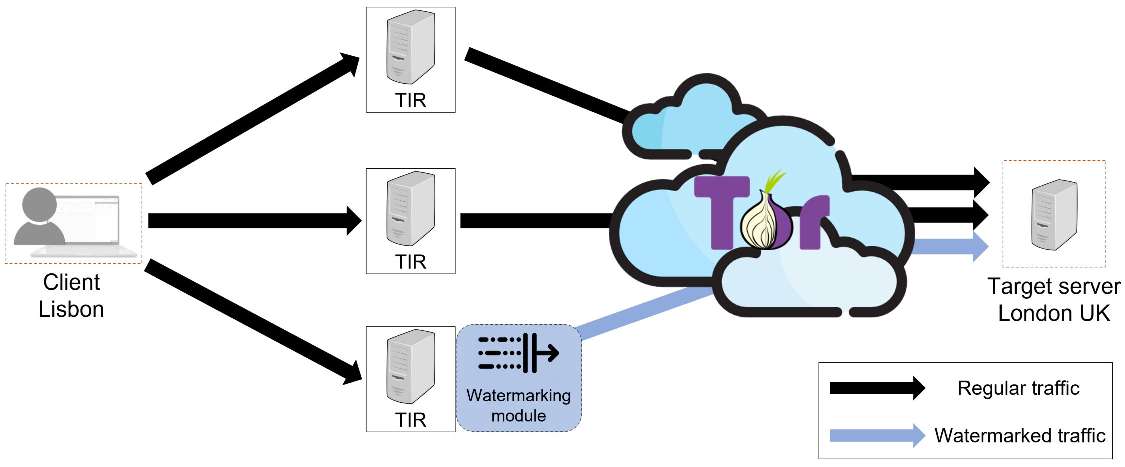

Figure 6.4: Representation of a TorMarker deployment in a 3 TIR setup.

the size of the flows, we performed predictions of 50, 100 and 150 packet-sized flows for the first two scenarios. Different types of traffic have different frequencies of packets which means analysing the trade-off between flow size, FPR and accuracy might show us better strategies for watermark detection. For 50 packet-sized flows, the kernel sizes shown in table 4.1 of the convolutional layers can't process flows that small so they were changed to a size of (1, 5) in both layers.

Regarding the watermarks, to understand the effectiveness of both strategies:

- ICBW watermarks were tested with *amplitude* values of 40ms, 80ms and 120ms in the *TIR* scenario evaluation. We only tested amplitude values of 40ms and 80ms in the control scenario because testing higher amplitudes, which correspond to more robust marks, would be unnecessary regarding the lack of countermeasures. This allowed us to save time since dataset generation is a very time-consuming process;

- RAINBOW watermarks were tested with a constant *max amplitude* value of 80 and varying *amplitude* values of 10 and 40. Only varying the *amplitude* will let us know which is the best granularity of delays applied in this watermarking strategy while also making it possible to be compared to the best parametrization of the ICBW watermark.

The watermarking detection for the first two scenarios was evaluated according to two metrics: false positive rate(FPR) which refers to the number of regular flows identified as watermarked flows and the global accuracy of the predictions. Since our priority is to build a tool with a low rate of mislabelled flows, we can verify the model's reliability through the FPR, while observing its overall performance through the accuracy metric. As for the third scenario, we didn't use false positive or accuracy rates. We have an unbalanced test set with a varying ratio of marked and regular flows, the ratios used previously wouldn't provide useful information regarding this scenario. Instead, we used the precision metric which provides the necessary information to understand at what point the tool loses its reliability, given that it shows the ratio of correctly labelled watermarked flows to all flows labelled as watermarked.

Dataset generation and partitioning

Regarding the dataset generation, we wanted to recreate real Tor and TIR flows as close as possible since we aim to evaluate the effectiveness of embedding and detecting watermarks in different network conditions. So, as mentioned before, we replayed real Tor traces into our controlled environment using the dataset ISCXTor [36]. After these traces are captured arriving at the target server, we compiled them into different datasets of real tor traces partitioned according to the scenario followed. Furthermore, we generated these evaluation datasets with two types of application data: Skype traffic and P2p traffic. This allows us to study the behaviour of the tool when dealing with data with different properties. For example, as shown in figure 6.5, P2p traffic has a higher frequency of packets per second which translates to lower IPDs but less consistency when compared to Skype traffic.

For each watermark tested with a specific *amplitude*, we generated a dataset with an average of 70K packets, where the training set is equally populated with regular and watermarked traffic. This dataset is split into 70% for training and 30% for testing.

Performance

For the performance evaluation, we analysed the tool's training time, the times it takes to be ready to perform predictions, and testing time, the time it takes to perform those predictions, comparing these timings to the flow size used.

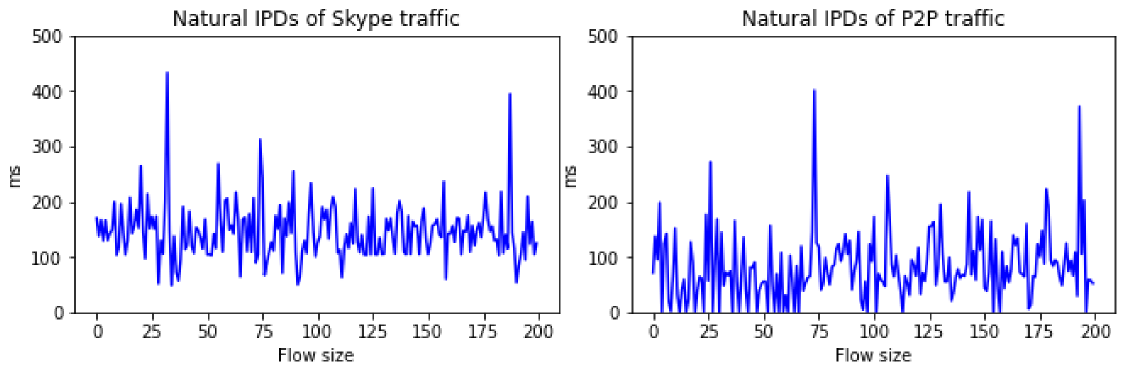

Figure 6.5: Plots of the natural IPDs found in regular flows from Skype traffic, left, and P2P traffic, right.

6.2 *Tor vanilla* scenario evaluation

This section documents the evaluations done on TorMarker when following the *Tor vanilla* scenario. This evaluation covers a scenario where there are no countermeasures to flow watermarking other than the natural jitter of the Tor network, caused by its onion routing architecture. It will analyse a control scenario, providing us with base results to be compared to the other evaluations, while giving us an idea of how the tool is effective when attempting to de-anonymize a *vanilla* Tor connection.

Skype traffic

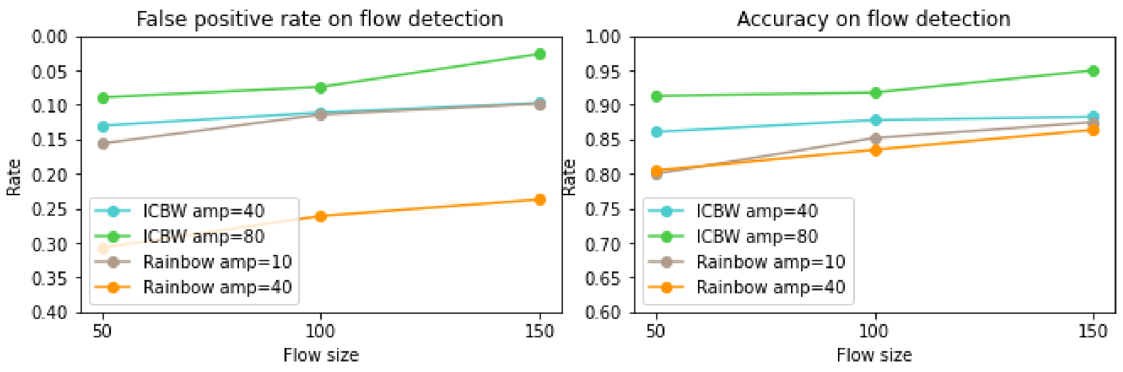

Figure 6.6: Plots obtained from watermarked flow detection of Skype Tor traffic, following scenario 1

Figure 6.6 shows two plots for our tool's FPR and accuracy when tested with Skype flows. The graphs clearly show that the ICBW watermark performed better in comparison to the Rainbow mark. When crafted with an *amplitude* of 80ms (green line), it reached an FPR of 2.5% and the accuracy was 95%. Such strong results were expected since there are no countermeasures to watermarking in this scenario, as explained before. Furthermore, the parametrization with the biggest *amplitude* got the best results because in the ICBW watermark the ampitude is directly correlated with the mark's intensity and consequently with its distinguishability.

It is also worth mentioning that the Rainbow watermarked performed better when the mark was applied with a smaller granularity, *amplitude* value of 10ms, obtaining an FPR of 9.8% and accuracy of 87.5% which is practically the same performance obtained by the ICBW mark with an *amplitude* of 40ms.

Finally, we can see that bigger flows provide a proportionally better prediction which is expected because having more packets naturally helps the detection of watermarking patterns. This observation should remain consistent throughout the rest of the evaluations.

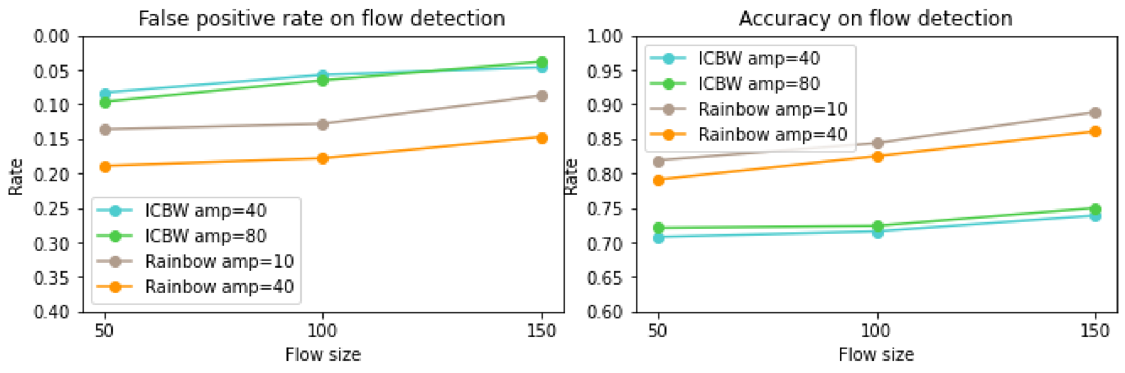

Figure 6.7: Plots obtained from watermarked flow detection of P2p Tor traffic, following scenario 1

P2p traffic

Moving on to P2P traffic, figure 6.7 plots the aforementioned metrics for predictions of P2P flows. Overall, when compared to Skype traffic, these graphs show us slightly worst results regarding false positive rates for both marks. When looking at the accuracy, the Rainbow watermark shows similar rates but the ICBW strategy's results drop substantially which leads us to believe that P2P traffic impairs the detectability of the ICBW watermark. In a more specific analysis, the ICBW watermark continues to show better results in the FPR metric with rates up to 4.6% but fell short in accuracy when compared to the Rainbow mark which obtained an accuracy of 88.9% when using 10ms of *amplitude*, compared to an accuracy of 75% from the ICBW watermark. This is consistent with our last observation about the detectability of the ICBW mark and shows that the Rainbow watermark can potentially perform better in tougher scenarios when using this type of traffic.

6.3 *TIR* scenario evaluation

This section will present the results obtained from the *TIR* scenario evaluation. Our aim here is to assess the effectiveness of our tool in embedding different watermarks with diverse parametrizations and the consequent detection of said watermarks on the other end of the connection. As mentioned in section 6.1.1, the adversary we emulate in this scenario has enough influence to aggregate all flows partitioned through the TIR input circuits, being able to completely watermark all of them.

Skype traffic

Sub-scenario 1

Firstly we will analyse the results obtained from processing the Skype traffic flows, shown in the plots of figure 6.8, where training and test flows are generated from the same TIR

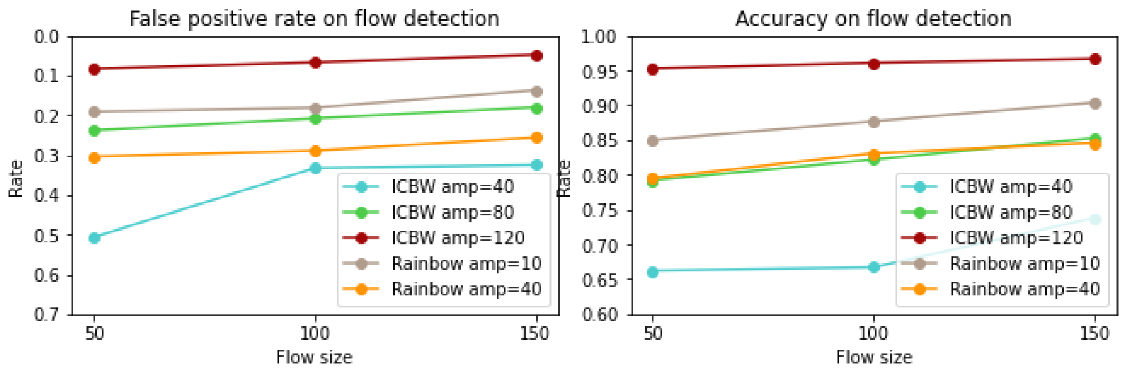

Figure 6.8: Plots obtained from watermarked flow detection of Skype Tor traffic, following sub-scenario 1

entry node. This specific sub-scenario aims to obtain an overall idea of how robust the different watermarks are when confronted with TIR's countermeasures.

As shown in the graphs, the ICBW watermark using an *amplitude* of 120ms obtained the best results in both metrics. With an FPR of 4.8% and an accuracy rate of 96.7%, it showed better results than its other parametrizations as expected. The rainbow watermark using an *amplitude* of 10ms is still consistently better than its other parametrization, obtaining false positive and accuracy rates of 13.7% and 90.4% respectively. In this instance, this Rainbow parametrization obtained better rates in both metrics than the ICBW parametrizations with *amplitude* levels of 40ms and 80ms which tells us that this watermark deals better with TIR countermeasures than the less robust versions of the ICBW watermarks.

Figure 6.9 plots the results from this scenario (solid lines) and the control scenario (dotted lines). Both watermarks have worse FPR, explained by the lack of countermeasures to watermarking in the control scenario. Furthermore, the Rainbow mark loses less FPR than the ICBW watermark and it gets even better accuracy levels in this scenario which validates the deduction that this watermark type works better against TIR's artificial jitter.

Sub-scenario 2

The plots in figure 6.10 show the results of the two best watermarking parametrizations when applied in both sub-scenarios specified in section 6.1.1. The dotted lines are the results from TorMarker when trained and tested with different entry nodes and the continuous lines represent the formerly analysed sub-scenario. The rationale behind the evaluation of TorMarker in this second sub-scenario is to assess how well the tool and its different parametrizations behave when tested with more diverse data.

In this sub-scenario, since the test data is routed through a whole different path and relayed through different circuits than the training one, the performance drop shown in the graphs is expected. The ICBW watermark got a non-substantial improvement in FPR

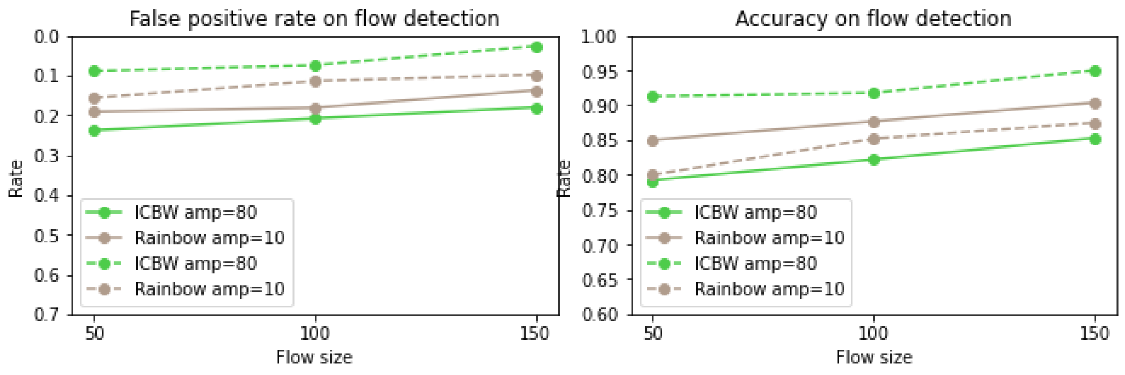

Figure 6.9: Plots obtained from watermarked flow detection of Skype Tor traffic, control scenario comparison

than the last sub-scenario in all three packet sizes but lost a lot in accuracy obtaining a maximum percentage of 85.4%. The Rainbow watermark only performed slightly worse obtaining a minimum FPR of 14.3% and an accuracy of 89.8%. The fact that the Rainbow watermark only obtained a slight drop in performance suggests that the watermark is more robust to different jitter conditions than the ICBW strategy.

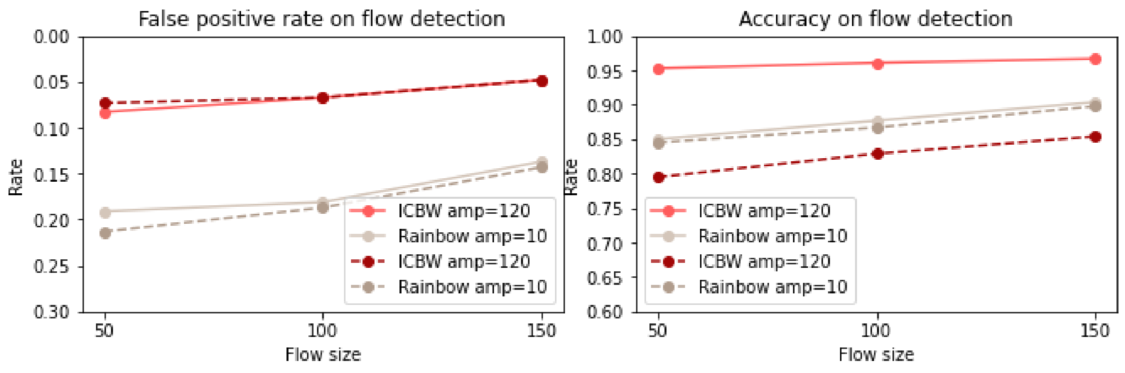

Figure 6.10: Plots obtained from watermarked flow detection of Skype Tor traffic, sub-scenarios comparison

The most aggressive parametrization of the ICBW watermark obtained the best results in the toughest scenario with an FPR of 4.4% and an accuracy rate of 85.4%. TorMarker's watermarks should be robust against network jitter and in this case TIR's additional delays. However, the trade-off between robustness and invisibility is an important key to building a good watermark [67]. When our tool obtains such a substantially low FPR against dedicated countermeasures and volatile jitter conditions, it leads us to believe that an *amplitude* of 120ms in the ICBW watermark strategy is excessive and might hurt the watermark's invisibility properties.

P2P traffic

Sub-scenario 1

Following the same experiment order, figure 6.11 plots the false positive and accuracy rates obtained by our tool after processing P2P flows from sub-scenario 1.

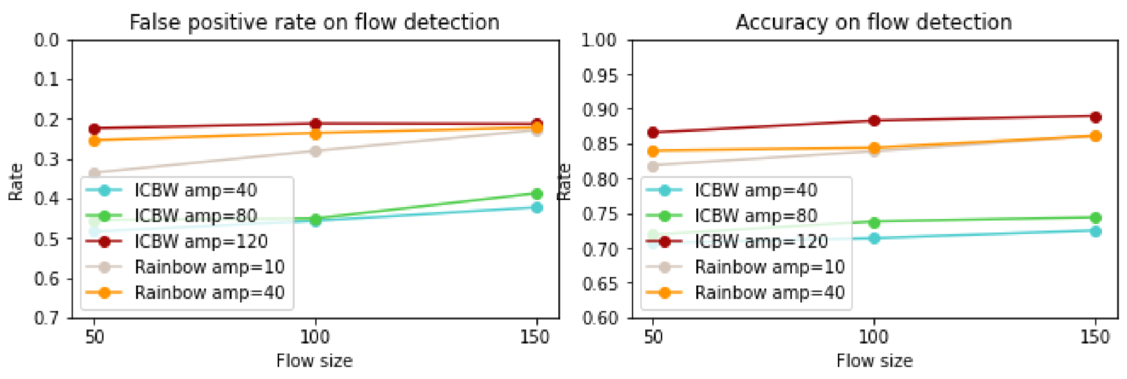

Figure 6.11: Plots obtained from watermarked flow detection of P2P Tor traffic, following sub-scenario 1

As shown in the graphs, the first plot shows that both parametrizations of the rainbow watermark and the most robust version of the ICBW mark obtained practically the same FPR and close accuracy percentages. However, the ICBW watermark was still slightly ahead in both metrics obtaining values of 21.3% and 89% in FPR and accuracy respectively. The ICBW parametrizations with lower *amplitudes* of 40ms and 80ms, obtained substantially bad results with FPRs over 40% which shows us *amplitudes* under 80ms are clearly not robust enough for this scenario and traffic type. These watermarks are being destroyed by the jitter and artificial delays added by TIR and the system is incapable of detecting them efficiently.

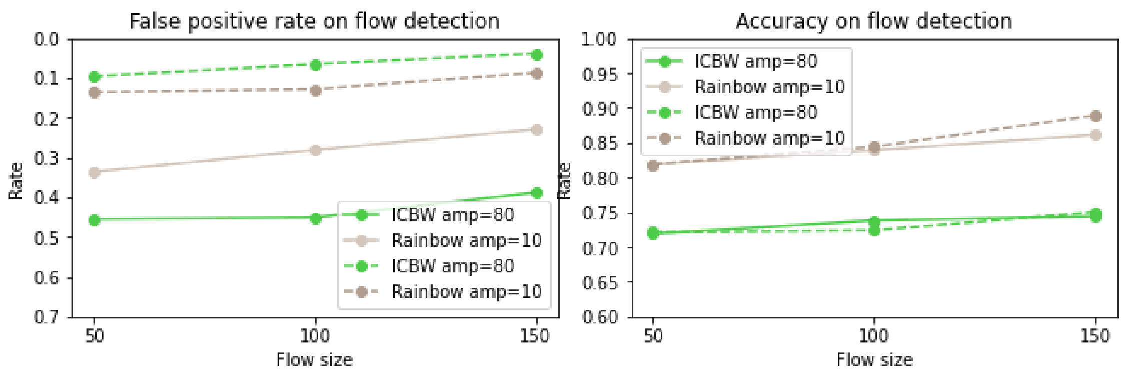

Figure 6.12: Plots obtained from watermarked flow detection of P2P Tor traffic, control scenario comparison

Moving on to the control scenario comparison, figure 6.12 shows plots comparing the results obtained in the control scenario (dotted lines) and this scenario (continuous line).

Much like we saw in the Skype traffic analysis, both marks obtained worse metrics in this scenario, as expected. For this traffic type, it is even more evident that the Rainbow mark is more robust against TIR's countermeasures by showing an FPR increase from 8.7% to 22.9%, a substantially smaller difference than the ICBW mark which showed an FPR increase from 3.8% to 38.8%.

Sub-scenario 2

Although the two Rainbow parametrizations obtained very similar results in the first sub-scenario, making it logical to compare them in this scenario, we opted to not show the results from the 40ms parametrization since the plots would become too difficult to read and this parametrization attained worse results than the other Rainbow mark.

With that being said, the evaluation results from the second sub-scenario are shown in figure 6.13. Values obtained from this second sub-scenario are represented with dotted lines and values from the first one are shown with a continuous line. Analysing the graphs, the ICBW watermark shows a slight drop in precision, within the expected amount. The Rainbow mark, however, presents overall better values in this second sub-scenario and it even surpasses the ICBW mark. The only explanation for this must rely on the volatile jitter conditions of the Tor network when the dataset was generated. Since TIR's countermeasures are somewhat consistent, the jitter on the Tor circuits at the time must have been substantially smaller than when the test sets for sub-scenario 1 were created.

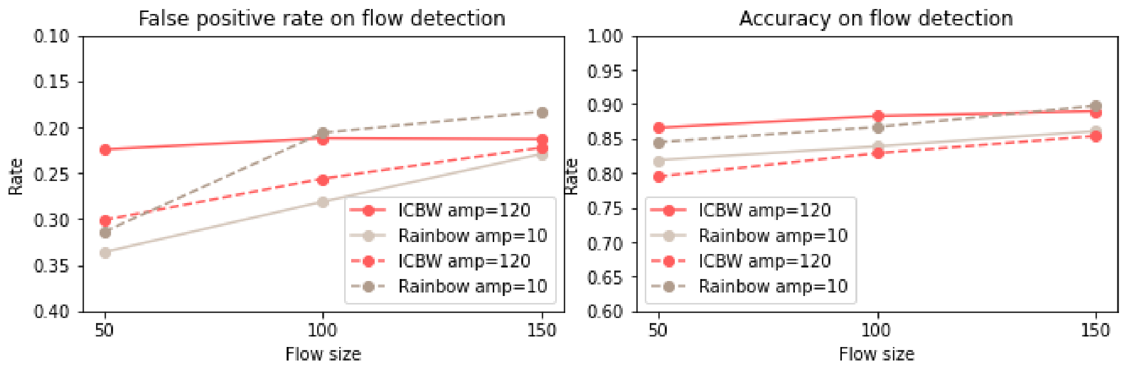

Figure 6.13: Plots obtained from watermarked flow detection of P2P Tor traffic, sub-scenarios comparison

To perform a final analysis of this scenario, we compared the results obtained in the second sub-scenario by the best two parametrizations for each traffic type, as shown in figure 6.14. TorMarker was more effective when marking and detecting Skype traffic which tells us that embedding watermarks in traffic with a lower frequency of packets per second results in a more reliable detection. The rainbow watermark obtained overall better accuracy results, which means that, for different circuits, this watermark type might potentially be more resistant to perturbations.

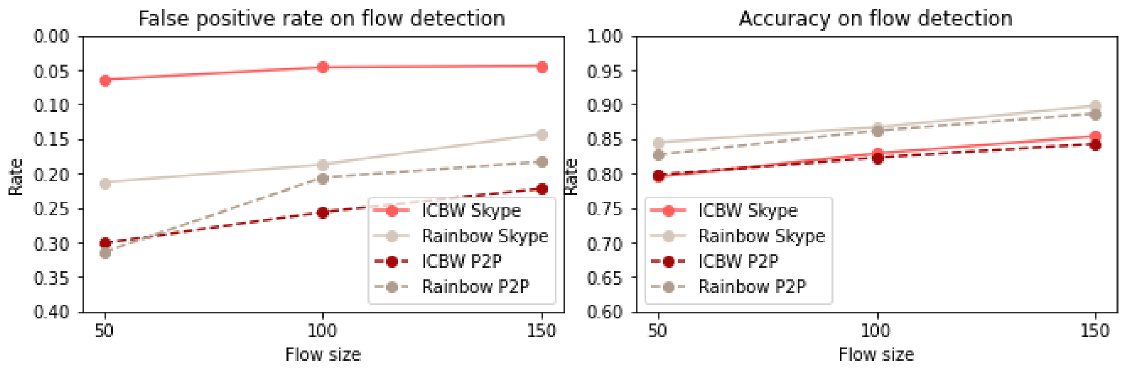

Figure 6.14: Plots obtained from watermarked flow detection of Skype and P2P Tor traffic

6.4 *TIR* partitioned flows evaluation

The purpose of this evaluation, as specified in section 6.1.1, lies in understanding the reliability of TorMarker when confronted with a connection partitioned through K input circuits. This scenario will emulate a real TIR connection where traffic from the targetted connection is partitioned through several different circuits.

Our tool was tested using the two watermarking parametrizations which obtained the best results in the former evaluations: the ICBW watermark, with an *amplitude* of 120ms; and the Rainbow watermark, with an *amplitude* of 10ms. Regarding the flows used, we decided to use traffic flows generated according to sub-scenario 2 since it will give us the results that best translate into a real-world instance.

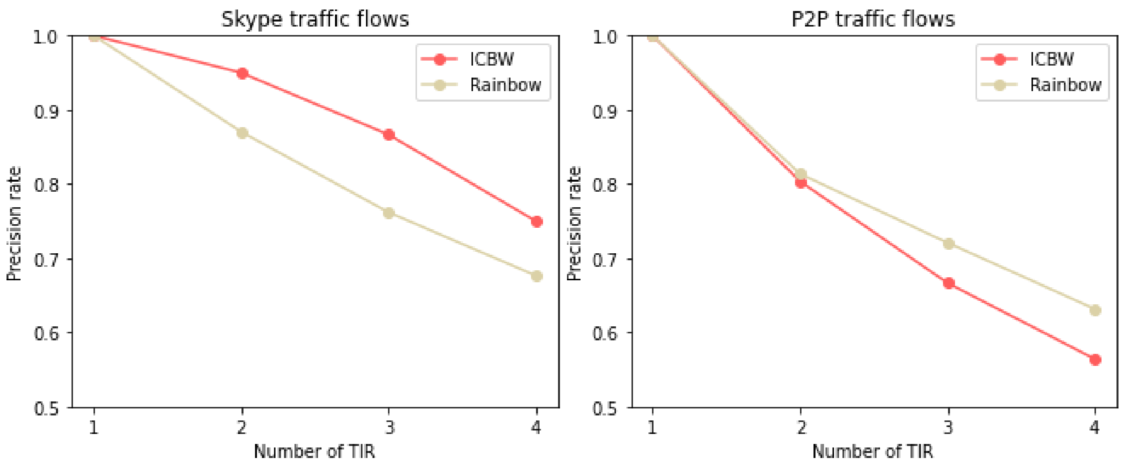

Figure 6.15: Plots showing the precision rate obtained by TorMarker against different numbers of TIR nodes.

Figure 6.15 shows the precision rates obtained by TorMarker when attempting to detect watermarked flows, in a connection partitioned by a varying number of TIR nodes. Overall, the predictions made on Skype flows obtained better precision when compared to the P2P traffic type which makes sense according to previous evaluations. In both

Flow size:	Training time in seconds:	Testing time in seconds:
150	6,174	0.144
100	6,214	0.221
50	7,036	0.120

Table 6.1: Training and testing times for 70K packets.

traffic types, precision rates drop when more TIR nodes are used which is expected since with more nodes we have more regular flows leading to more false positive predictions. When using a 2 TIR setup, both watermarks for both traffic types were able to obtain rates over 80% and for the 3 TIR setup, only the ICBW mark in P2P flows couldn't obtain a precision score over 70% and the best score was 86.2% obtained by the ICBW watermark for Skype traffic. With the 4 TIR setup, the ICBW watermark with Skype traffic got a precision rate of 75% and was the only mark to obtain a score over 70%.

Concerning these observations, it is clear that a TIR setup with 1 or 2 nodes still allows TorMarker to perform predictions with good confidence and a 3 TIR setup starts to substantially lower the tool's precision. With 4 nodes, to perform predictions with decent precision, the watermark and its parametrizations need to be specially chosen according to the traffic type present in the connection the tool is attempting de-anonymize. Even if the tool is against setups with as many as 4 TIR, performing various captures for TorMarker to process will allow the attacker to detect enough watermarked flows to confidently de-anonymize targeted connections in a reasonable observation time.

These results show that TIR setups with up to three nodes are vulnerable to TorMarker's watermarking attacks, and four TIR setups are also weak against this approach. We recommend that, in order to guarantee K-anonymization, the focused TIR solution must be deployed with at least four nodes and it's watermarking countermeasures must be updated to resist more robust watermark types.

6.5 Performance evaluation

As we mentioned in section 6.1.2, we're using 70K packets to perform each evaluation. From these, around 20K are used for testing and the other 50K for training. The training test itself is divided 75% for training the model and 25% for validation but to facilitate the observations we will not distinguish these subsets of the training set.

Table 6.1 presents the timings obtained when TorMarker is trained with 50K packets and performs predictions over 20K packets, amounts used in the previous evaluations. Regarding the training times, we can see that for smaller flow sizes the network takes a little bit more time to converge and finish training because smaller flow sizes will translate into more flows extracted from the dataset which are then processed. Testing times are not substantially different and don't seem to follow a pattern other than being extremely short.

Overall, these times are very short. The CNN model is very light since we use few layers and only have to process one feature type, using a powerful GPU to perform all calculations allows for training and testing times to be very small. We understand that training TorMarker with only 50K packets for each evaluation may limit our tool's scores since reaching amounts such as 100K or even 200K packets for training may substantially enhance the tool's reliability and effectiveness. Despite that, we are pretty confident about the results obtained from TorMarker and by not generating bigger training sets for each evaluation, we were able to test more watermark types and parametrizations and how these dealt with different types of traffic.

6.6 Main findings

Taking into account the results from the described experimental evaluations:

- TIR's countermeasures against timing correlation attacks proved to be more effective on flows originating from Skype Tor traffic. Nonetheless, TorMarker performed considerably better when watermarking and detecting watermarks from Skype-type flows.

- The ICBW watermark showed good results, specifically against Skype traffic, obtaining a high detection accuracy while maintaining low false positive rates. However, the more robust parametrizations of this watermark might be hurting its invisibility properties in exchange for good distinguishability.

- The Rainbow watermark obtained better results when its parametrizations followed a lower granularity with lower *amplitude* values. This watermark also showed good compatibility with P2P traffic flows when compared to its alternative.

- When confronted against partitioned flows, TorMarker was successful when attempting to detect watermarked flows and presented reasonable precision results from processing flows from a 3 TIR setup. To accomplish the same in 4 TIR setups was only possible with specific parametrizations for specific traffic types.

- Training and testing times are short because the model used in the detection component is light and the GPU used to perform these tasks was powerful. Furthermore, since train and test times were substantially short, training TorMarker with bigger and more diverse datasets is a feasible possibility and might allow the tool to perform with better scores.

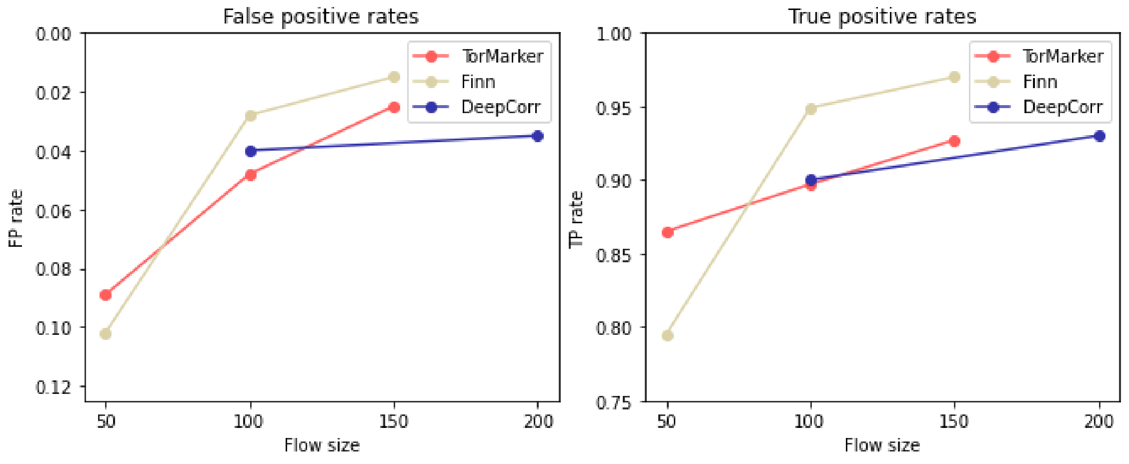

Figure 6.16: Plots showing the false positive and true positive rates obtained by TorMarker, Finn and DeepCorr.

6.7 State-of-the-art tools comparison

Considering all the research work done previously to the implementation of TorMarker, there are two state-of-the-art tools that should be analysed performance-wise in a comparative matter to our tool. These flow correlator tools are DeepCorr [50], which works through deep learning correlations of traffic properties, and Finn [67], which performs the detection of watermarked flows and subsequent extraction of watermark-embedded data. Both of these tools are recent implementations that obtained remarkable performance results when compared to earlier works, proving that a comparative analysis between them and TorMarker should contribute to the validation of this tool.

In figure 6.16 are presented two plots with the metrics obtained by the three tools, FPR on the left and TPR on the right. These metrics were extracted from the official studies of both tools [50, 67]. Finn metrics were explicit however, for DeepCorr rates, the only documented flow sizes appropriate for this comparison are of 100 and 200 packets and since the work doesn't present fixed FPRs we retrieved approximate values for a target TPR that was closest to TorMarker's results because our tool is the focus of comparison.

Regarding the comparison analysis, we can see that Finn leads both in FPR and TPR for flow sizes of 150 and 100 obtaining FP rates as low as 1.5% and TP rates up to 97.0%. For flows of size 50, TorMarker was able to achieve the better rates than Finn meaning it has potentially the best performance for small-sized flows. If we look at DeepCorr's rates for 200 packet-sized flows, it got an approximate FPR of 3.5% for a target TPR of 93% (since our tool achieved a TPR of 92.7 for 150 packet-sizes flows we extracted DeepCorr's results regarding a TPR as close as possible to TorMarker's). Compared to TorMarker's FPR of 2.5% with a TPR of 92.7%, for processing flow sizes of 150 packets, we can conclude that our tool managed to achieve slightly better results than DeepCorr, presenting a smaller FPR, for approximately the same TPR, for the detection of 50 packets smaller flow sizes.

However, for flows of size 100, DeepCorr achieves a substantially lower FPR of 4.0% when compared to TorMarker's rate of 7.4%. It is important to mention that both of these state-of-the-art tools learn from training datasets containing up to 500K packets, which compared to the 50K packets used to train TorMarker shows a solid justification for these results. A more robust training set would possibly allow for TorMarker to surpass Finn's achieved metrics.

In conclusion, this comparative analysis shows that TorMarker is capable of competing with two of the best flow correlation tools in the state-of-the-art, DeepCorr and Finn, and it could potentially outperform them if a bigger training set is supplied.

6.8 Summary

Chapter 7 presented all evaluations performed on TorMaker, validating this prototype as an effective and reliable de-anonymizer of Tor flows. It started by describing the evaluation methodology used for all experimental testing (section 6.1) . Firstly, the section glanced over the specifications of the machines that make up the full environment and then provided with a description about the full extent of the evaluations, going over the experimental roadmap and the several different scenarios it followed. It also provided information about the testing methodology and metrics used to base such evaluations. The sections that followed present the evaluation results of TorMarker against each scenario: the first section described the evaluation against Tor *vanilla* flows (section 6.2), providing with results that served as the base for the comparative analysis against tougher scenarios and other tools; the two sections that followed contained the most relevant evaluations of the dissertation since they described the tool's experimental results against TIR, without traffic partitioning (section 6.3) and with traffic partitioning of the targeted connection, performed by varying numbers of TIR (section 6.4). This last section also provided a security analysis of TIR with recommendations for a secure deployment of this prototype against watermarking attacks from TorMarker. Section 6.5 analyses the performance of TorMarker regarding processing times of the training and testing stages. The main findings of all experimental evaluations were summed up in section 6.6 and were followed by the final section of the chapter that performs a comparative analysis of TorMarker against other relevant state-of-the-art tools (section 6.7).

7 Conclusions

To maintain social and political dominance, totalitarian regimes often implement strict approaches towards blocking content being shared that goes against their interests. Circumvention strategies for these behaviours are necessary to provide some form of uncensored means of communication to people affected by these abusive governments. The Tor network was created precisely with this intent: providing everyone with access to an uncensored internet. This voluntary-based distributed network implements a protocol that allows for private, anonymous, low-latency communication through the world wide web. Despite all efforts devoted to reinforcing this network's security, many types of attacks leveraging AS-level influence have been successfully deployed, proving it is not invulnerable.

In this dissertation, we have designed a test-bench environment and prototype to evaluate a candidate Tor reinforcement, offering valuable information regarding the security provided by K-anonymity enforcement solutions. TorMarker is a system capable of embedding timing watermarks in inbound Tor traffic and performing the detection of said watermarks in outbound Tor traffic, consequently de-anonymizing the watermarked connection. Deploying our tool in a Tor environment, reinforced with traffic partitioning through K-input circuits, allowed us to perform security evaluations on the TIR prototype: a Tor reinforcement which provides K-anonymity through the addition of TLS tunnelled, covert input circuits. This evaluation further elaborates TIR's security analysis by testing the prototype against new approaches based on proactively induced traffic watermarking and watermark detection through ML&DA of marked Tor traffic.

7.1 Main contributions

The main contributions of this dissertation were as follows:

- We designed and implemented TorMarker, a system capable of de-anonymizing connections made over the Tor network, and deployed it in our test-bench environment that comprehends Tor connections supported by the Tor-reinforcement and K-anonymization solution: TIR. TorMarker detects actively embedded watermarks in Tor flows leveraging state-of-the-art techniques, such as traffic time-based watermarking and deep learning of traffic time properties through convolutional neural networks.

- We performed experimental evaluations on the full extent of TorMarker's watermarks and their different parametrizations against *vanilla* Tor flows and TIR-supported Tor flows. We showed that the tool can perform the detection of different types of watermarks embedded in two different Tor traffic types. With the proper parametrization, TorMarker obtained low false positive detection rates with good accuracy and precision rates, in short train and test times.

- We conducted a new security analysis on TIR using state-of-the-art techniques and recommended deployment specifications to improve the targeted Tor-strengthening solution against our implemented tool. TIR's traffic partitioning was tested against the precision of TorMarker by processing flows from setups with different numbers of TIR nodes, showing that the best watermark parametrizations were able to obtain reasonably good precision rates in setups with up to 4 TIR.

7.2 Open issues

In the relation between the initial dissertation's goals and the finished study, some open issues are derived as posterior work. The most evident one was the temporal performance analysis which could be further enriched by having larger captures. Flow generation is very time-consuming, following performance analyses of TorMarker can focus more time on dataset generation to obtain information regarding the trade-off between precision scores in watermark detection and the number of flows used for training, providing new strategies regarding the compromises that can be made to increase precision at the cost of performance.

The second issue would be the parametrization diversity, more specifically the robustness granularity at which the watermarks are evaluated. Different network jitter conditions require diverse levels of watermark robustness and more robustness hurts the watermark's unobservability properties. Testing the watermark's robustness at a lower granularity would further contribute to finding the ideal robustness level that achieves reasonable precision scores while maintaining a sufficient degree of unobservability.

This issue takes us to the last point concerning the analysis of a watermark's unobservability criteria. Further analysis of TorMarker could focus on the embedded watermark's unobservability level in comparison to the regular traffic used to embed the mark. Together with the tool's detection metrics, this analysis can also contribute to the calibration of a watermark's ideal robustness.

7.3 Future implementations

All efforts dedicated to TorMarker's design, implementation, experimental evaluation and background research allowed us to identify potential enhancements that could be implemented in this prototype. They are presented in the following section:

- TorMarker's parametrizations could be extended by enhancing the watermarking component, allowing it to embed more watermark types. With more parametrization options, the tool should gain more adaptability and perform better against more diverse traffic types. Extending the tool's parametrization is also possible by adding diversification to the nature of the watermark's carrier, for example, the implementation of *rate-based* watermarks would provide an adversary with another attack vector potentially boosting his chances of success.

- Regarding TorMarker's experimental evaluations, the tool's watermarks were tested against two types of traffic that would not suffer any content transformations. As a way to extend our tool's effectiveness evaluations, TorMarker watermark embedding and detection could be tested against other Tor reinforcement solutions which provide traffic morphing options, for instance, RTC traffic-based morphing channels. Since TorMarker relies on time-based carriers for embedding its watermarks, testing this carrier type's robustness against type-morphing traffic would contribute greatly to understanding the effectiveness of this class of Tor reinforcement systems.

- Another enhancement of TorMarker that would greatly improve the tool's interest in a real-world deployment would be the implementation of real-time time connection de-anonymization. Implementing TorMarker as a centralized system, coordinated with strategically placed probes that perform the watermark embedding, would provide the adversary with the ability to link Tor endpoints in real time. This would also allow for the tool, with enough allocated resources and watermarking probes, to watermark different Tor circuits simultaneously, which could better the chances of watermark detection and possibly brute-force through some of Tor's countermeasures, such as periodic guard node rotation.

Bibliography

[1] M. Akhoondi, C. Yu, and H. V. Madhyastha. "LASTor: A low-latency AS-aware Tor client". In: *2012 IEEE Symposium on Security and Privacy*. IEEE. 2012, pp. 476–490 (cit. on pp. 2, 19, 20).

[2] M. Alimardani and S. Milan. "The Internet as a global/local site of contestation: The case of Iran". In: *Global cultures of contestation*. Springer, 2018, pp. 171–192 (cit. on pp. 1, 2, 7).

[3] Y. Angel. *obfs4*. `https://github.com/Yawning/obfs4/blob/master/doc/obfs4-spec.txt`. (Visited on 2022-02-03) (cit. on pp. 2, 12).

[4] S. Aryan, H. Aryan, and J. A. Halderman. "Internet censorship in Iran: A first look". In: *3rd {USENIX} Workshop on Free and Open Communications on the Internet ({FOCI} 13)*. 2013 (cit. on pp. 1, 2, 8).

[5] D. Barradas et al. "Poking a hole in the wall: Efficient censorship-resistant Internet communications by parasitizing on WebRTC". In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 2020, pp. 35–48 (cit. on p. 24).

[6] L. Basyoni et al. "Traffic analysis attacks on Tor: a survey". In: *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*. IEEE. 2020, pp. 183–188 (cit. on p. 14).

[7] S. Chakravarty, A. Stavrou, and A. D. Keromytis. "LinkWidth: a method to measure link capacity and available bandwidth using single-end probes". In: (2008) (cit. on p. 15).

[8] S. Chakravarty, A. Stavrou, and A. D. Keromytis. "Traffic analysis against low-latency anonymity networks using available bandwidth estimation". In: *European symposium on research in computer security*. Springer. 2010, pp. 249–267 (cit. on pp. 13, 15).

[9] S. Chakravarty et al. "Detecting traffic snooping in tor using decoys". In: *International Workshop on Recent Advances in Intrusion Detection*. Springer. 2011, pp. 222–241 (cit. on pp. 19, 20).

[10] D. L. Chaum. “Untraceable electronic mail, return addresses, and digital pseudonyms”. In: *Communications of the ACM* 24.2 (1981), pp. 84–90 (cit. on p. 8).

[11] J. D. Clark et al. “The shifting landscape of global internet censorship”. In: (2017) (cit. on pp. 1, 2, 7).

[12] R. Dingledine et al. “One fast guard for life (or 9 months)”. In: *7th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2014)*. 2014 (cit. on pp. 2, 16, 19, 20).

[13] R. Dingledine et al. “Tor: The second-generation onion router (2014 DRAFT v1)”. In: *Cl. Cam. Ac. Uk* (2014) (cit. on pp. 1, 2, 9–11).

[14] J. R. Douceur. “The sybil attack”. In: *International workshop on peer-to-peer systems*. Springer. 2002, pp. 251–260 (cit. on p. 16).

[15] M. Edman and P. Syverson. “AS-awareness in Tor path selection”. In: *Proceedings of the 16th ACM conference on Computer and communications security*. 2009, pp. 380–389 (cit. on pp. 19, 20).

[16] T. Elahi et al. “Changing of the guards: A framework for understanding and improving entry guard selection in Tor”. In: *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*. 2012, pp. 43–54 (cit. on pp. 2, 19, 20).

[17] N. S. Evans, R. Dingledine, and C. Grothoff. “A Practical Congestion Attack on Tor Using Long Paths.” In: *USENIX Security Symposium*. 2009, pp. 33–50 (cit. on pp. 13, 14).

[18] B Evers et al. *Thirteen Years of Tor Attacks*. 2016 (cit. on pp. 2, 3, 11–14, 16).

[19] D. R. Figueiredo, P. Nain, and D. Towsley. “On the analysis of the predecessor attack on anonymity systems”. In: *Computer Science Technical Report* (2004), pp. 04–65 (cit. on pp. 2, 3, 16).

[20] K. Gallagher. “How Tor helped catch the Harvard bomb threat suspect”. In: (2013). (Visited on 2022-02-03) (cit. on p. 11).

[21] S. Golkar. “Liberation or Suppression Technologies? The Internet, the Green Movement and the Regime in Iran.” In: *International Journal of Emerging Technologies & Society* 9.1 (2011) (cit. on pp. 1, 2, 7).

[22] A. Houmansadr and N. Borisov. “BotMosaic: Collaborative network watermark for the detection of IRC-based botnets”. In: *Journal of Systems and Software* 86.3 (2013), pp. 707–715 (cit. on pp. 17, 18).

[23] A. Houmansadr and N. Borisov. “SWIRL: A Scalable Watermark to Detect Correlated Network Flows.” In: *NDSS*. 2011 (cit. on pp. 2, 3, 18, 25).

[24] A. Houmansadr, N. Kiyavash, and N. Borisov. “Multi-flow attack resistant watermarks for network flows”. In: *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2009, pp. 1497–1500 (cit. on pp. 3, 18, 25, 36, 42).

[25] A. Houmansadr, N. Kiyavash, and N. Borisov. “RAINBOW: A Robust And Invisible Non-Blind Watermark for Network Flows.” In: *NDSS*. Vol. 47. Citeseer. 2009, pp. 406–422 (cit. on pp. 3, 18, 25, 36, 41).

[26] J. Huang et al. “Long PN code based DSSS watermarking”. In: *2011 Proceedings IEEE INFOCOM*. IEEE. 2011, pp. 2426–2434 (cit. on p. 18).

[27] A. Iacovazzi and Y. Elovici. “Network flow watermarking: A survey”. In: *IEEE Communications Surveys & Tutorials* 19.1 (2016), pp. 512–530 (cit. on pp. 17, 18, 26).

[28] A. Iacovazzi, D. Frassinelli, and Y. Elovici. “The {DUSTER} attack: Tor onion service attribution based on flow watermarking with track hiding”. In: *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*. 2019, pp. 213–225 (cit. on pp. 2, 3, 17, 26).

[29] A. Iacovazzi, S. Sarda, and Y. Elovici. “Inflow: Inverse network flow watermarking for detecting hidden servers”. In: *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE. 2018, pp. 747–755 (cit. on p. 26).

[30] A. Johnson et al. “Users get routed: Traffic correlation on Tor by realistic adversaries”. In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 2013, pp. 337–348 (cit. on pp. 19, 20).

[31] J. Juen et al. “Defending tor from network adversaries: A case study of network path prediction”. In: *Proceedings on Privacy Enhancing Technologies* 2015.2 (2015), pp. 171–187 (cit. on pp. 19, 20).

[32] Keras. *Keras*. `https://keras.io/`. (Visited on 2022-02-17) (cit. on p. 42).

[33] S. Khattak et al. “SOK: Making sense of censorship resistance systems”. In: *Proceedings on Privacy Enhancing Technologies* 2016.4 (2016), pp. 37–61 (cit. on pp. 1, 2, 7, 8).

[34] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on p. 43).

[35] K. Kohls et al. “On the Challenges of Geographical Avoidance for Tor.” In: *NDSS*. 2019 (cit. on pp. 2, 19, 21, 22).

[36] A. H. Lashkari et al. “Characterization of tor traffic using time based features.” In: *ICISSp*. 2017, pp. 253–262 (cit. on pp. 33, 35, 39, 40, 52).

[37] D. Levin et al. “Alibi routing”. In: *ACM SIGCOMM Computer Communication Review* 45.4 (2015), pp. 611–624 (cit. on pp. 19, 21).

[38] B. N. Levine et al. “Timing attacks in low-latency mix systems”. In: *International Conference on Financial Cryptography*. Springer. 2004, pp. 251–265 (cit. on pp. 2, 3, 14).

[39] Z. Li, S. Herwig, and D. Levin. "{DeTor}: Provably Avoiding Geographic Regions in Tor". In: *26th USENIX Security Symposium (USENIX Security 17)*. 2017, pp. 343–359 (cit. on pp. 2, 19, 21).

[40] Z. Li, R. Yuan, and X. Guan. "Traffic classification-towards accurate real time network applications". In: *International Conference on Human-Computer Interaction*. Springer. 2007, pp. 67–76 (cit. on p. 11).

[41] Z. Lin and N. Hopper. "New attacks on timing-based network flow watermarks". In: *21st USENIX Security Symposium (USENIX Security 12)*. 2012, pp. 381–396 (cit. on pp. 25, 26).

[42] P. Liubinskii. "The Great Firewall's active probing circumvention technique with port knocking and SDN". English. Master's thesis. Aalto University. School of Electrical Engineering, 2021, p. 65. URL: `http://urn.fi/URN:NBN:fi:aalto-202101311842` (cit. on pp. 2, 8).

[43] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: `https://github.com/joaomlourenco/novathesis/raw/master/template.pdf` (cit. on p. iii).

[44] X. Luo et al. "Exposing invisible timing-based traffic watermarks with BACKLIT". In: *Proceedings of the 27th Annual Computer Security Applications Conference*. 2011, pp. 197–206 (cit. on pp. 25, 26).

[45] A. Macrina and E. Phetteplace. "The Tor browser and intellectual freedom in the digital age". In: *Reference and User Services Quarterly* 54.4 (2015), pp. 17–20 (cit. on pp. 2, 11).

[46] S. Matic, C. Troncoso, and J. Caballero. "Dissecting tor bridges: a security evaluation of their private and public infrastructures". In: *Network and Distributed Systems Security Symposium*. The Internet Society. 2017, pp. 1–15 (cit. on p. 12).

[47] P. Mayank and A. Singh. "Tor traffic identification". In: *2017 7th International Conference on Communication Systems and Network Technologies (CSNT)*. IEEE. 2017, pp. 85–91 (cit. on p. 11).

[48] S. J. Murdoch and G. Danezis. "Low-cost traffic analysis of Tor". In: *2005 IEEE Symposium on Security and Privacy (S&P'05)*. IEEE. 2005, pp. 183–195 (cit. on pp. 14, 15).

[49] S. J. Murdoch and P. Zieliński. "Sampled traffic analysis by internet-exchange-level adversaries". In: *International workshop on privacy enhancing technologies*. Springer. 2007, pp. 167–183 (cit. on pp. 2, 3, 14).

[50] M. Nasr, A. Bahramali, and A. Houmansadr. "Deepcorr: Strong flow correlation attacks on Tor using deep learning". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 1962–1976 (cit. on pp. 2, 3, 15, 27–29, 44, 49, 62).

[51] V. Nunes. "Hardening Tor against State-Level Traffic Correlation Attacks with K-Anonymous Circuits". MA thesis. IST, Universidade de Lisboa, 2021 (cit. on pp. 2, 3, 19, 22, 29).

[52] Opencsv. *Opencsv*. `http://opencsv.sourceforge.net/`. (Visited on 2022-02-17) (cit. on p. 40).

[53] G. Owen and N. Savage. "Empirical analysis of Tor hidden services". In: *IET Information Security* 10.3 (2016), pp. 113–118 (cit. on pp. 2, 11).

[54] Pcap4j. *Pcap4j*. `www.pcap4j.org/`. (Visited on 2022-02-17) (cit. on p. 40).

[55] R. Pries et al. "A new replay attack against anonymous communication networks". In: *2008 IEEE International Conference on Communications*. IEEE. 2008, pp. 1578–1582 (cit. on p. 15).

[56] T. T. Project. *AChildsGardenOfPluggableTransports*. `https://gitlab.torproject.org/legacy/trac/-/wikis/doc/AChildsGardenOfPluggableTransports`. (Visited on 2022-02-03) (cit. on pp. 2, 12).

[57] T. T. Project. *How to Report Bad Relays*. `https://blog.torproject.org/how-report-bad-relays/`. (Visited on 2022-02-03) (cit. on p. 19).

[58] T. T. Project. *meek*. `https://gitlab.torproject.org/legacy/trac/-/wikis/doc/meek`. (Visited on 2022-02-03) (cit. on p. 12).

[59] T. T. Project. *Reporting Bad Relays*. `https://gitlab.torproject.org/legacy/trac/-/wikis/doc/ReportingBadRelays`. (Visited on 2022-02-03) (cit. on pp. 2, 19, 20).

[60] T. T. Project. *Snowflake*. `https://gitlab.torproject.org/tpo/anti-censorship/pluggable-transports/snowflake/-/wikis/Technical%20Overview`. (Visited on 2022-02-03) (cit. on pp. 2, 12).

[61] T. T. Project. *Tor FAQ*. `https://2019.www.torproject.org/docs/faq.html`. (Visited on 2022-02-03) (cit. on pp. 19, 21).

[62] T. T. Project. *Torproject.org Blocked by GFW in China: Sooner or Later?* `https://blog.torproject.org/torprojectorg-blocked-gfw-china-sooner-or-later/`. 2008. (Visited on 2022-02-03) (cit. on p. 11).

[63] T. T. Project. *What is a bridge*. `https://support.torproject.org/censorship/censorship-7/`. (Visited on 2022-02-03) (cit. on p. 11).

[64] D. Ramsbrock, X. Wang, and X. Jiang. "A first step towards live botmaster traceback". In: *International Workshop on Recent Advances in Intrusion Detection*. Springer. 2008, pp. 59–77 (cit. on p. 18).

[65] J.-F. Raymond. "Traffic analysis: Protocols, attacks, design issues, and open problems". In: *Designing privacy enhancing technologies*. Springer. 2001, pp. 10–29 (cit. on pp. 2, 11).

[66] M. Reed, P. Syverson, and D. Goldschlag. “Anonymous connections and onion routing”. In: *IEEE Journal on Selected Areas in Communications* 16.4 (1998), pp. 482–494. DOI: `10.1109/49.668972` (cit. on pp. 1, 9).

[67] F. Rezaei and A. Houmansadr. “FINN: Fingerprinting Network Flows using Neural Networks”. In: *Annual Computer Security Applications Conference*. 2021, pp. 1011–1024 (cit. on pp. 2, 3, 27–29, 33, 56, 62).

[68] Y. Shi and K. Matsuura. “Fingerprinting attack on the tor anonymity system”. In: *International Conference on Information and Communications Security*. Springer. 2009, pp. 425–438 (cit. on p. 14).

[69] O. Starov et al. “Measuring and mitigating AS-level adversaries against Tor”. In: *arXiv preprint arXiv:1505.05173* (2016) (cit. on pp. 2, 19–21).

[70] Y. Sun et al. “{RAPTOR}: Routing attacks on privacy in tor”. In: *24th USENIX Security Symposium (USENIX Security 15)*. 2015, pp. 271–286 (cit. on pp. 2, 3, 18–21, 27).

[71] J. Teixeira. “Strengthening of Tor Against Traffic Correlation with K-Anonymity Input Circuits”. MA thesis. FCT/UNL, 2021 (cit. on pp. 2–4, 19, 23, 24, 29).

[72] J. Teixeira and H. Domingos. “TIR - Strenthening Tor with Stunneled K-Anonymized Input Circuits – TIR Prototype and Technical Report”. In: *DI/FCT/UNL – NOV LINCS Research Center* (2021) (cit. on pp. 35, 39, 50).

[73] TensorFlow. *TensorFlow*. `https://www.tensorflow.org/`. (Visited on 2022-02-17) (cit. on p. 42).

[74] *Tor project*. `www.torproject.org` (cit. on pp. 1, 8).

[75] C. Wacek et al. “An Empirical Evaluation of Relay Selection in Tor.” In: *NDss*. 2013 (cit. on pp. 19, 20).

[76] X. Wang et al. “A potential HTTP-based application-level attack against Tor”. In: *Future Generation Computer Systems* 27.1 (2011), pp. 67–77 (cit. on pp. 13, 15, 16).

[77] X. Wang and D. S. Reeves. “Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays”. In: *Proceedings of the 10th ACM conference on Computer and communications security*. 2003, pp. 20–29 (cit. on p. 25).

[78] X. Wang et al. “Sleepy watermark tracing: An active network-based intrusion response framework”. In: *IFIP International Information Security Conference*. Springer. 2001, pp. 369–384 (cit. on pp. 2, 3, 17, 18).

[79] Z. Weinberg et al. “How to catch when proxies lie: Verifying the physical locations of network proxies with active geolocation”. In: *Proceedings of the Internet Measurement Conference 2018*. 2018, pp. 203–217 (cit. on pp. 19, 21).

[80] P. Winter and S. Lindskog. *How China Is Blocking Tor*. 2012. arXiv: `1204.0447 [cs.CR]` (cit. on pp. 1, 2, 8, 11).

[81] P. Winter et al. "Identifying and characterizing Sybils in the Tor network". In: *25th USENIX Security Symposium (USENIX Security 16)*. 2016, pp. 1169–1185 (cit. on p. 16).

[82] P. Winter et al. "Spoiled onions: Exposing malicious Tor exit relays". In: *International Symposium on Privacy Enhancing Technologies Symposium*. Springer. 2014, pp. 304–331 (cit. on pp. 19, 20).

[83] W. Wong. "Stunnel: SSLing Internet Services Easily". In: *SANS Institute, November* (2001) (cit. on p. 23).

[84] M. Wright et al. "Defending anonymous communications against passive logging attacks". In: *2003 Symposium on Security and Privacy, 2003.* IEEE. 2003, pp. 28–41 (cit. on pp. 2, 3, 16).

[85] Y. Zhu et al. "Correlation-based traffic analysis attacks on anonymity networks". In: *IEEE Transactions on Parallel and Distributed Systems* 21.7 (2009), pp. 954–967 (cit. on pp. 2, 3, 14).