LOOPTOR

Fighting traffic correlation on the Tor network

A Degree Thesis
Submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
by

Jaume Planas i Planas

In partial fulfilment of the requirements for the degree in
Telecommunications Technologies and Services Engineering

Supervised by
Carmela Troncoso
École Polytechnique Fédérale de Lausanne
and
Oscar Esparza
Universitat Politècnica de Catalunya

June 2020

Abstract

Tor is a general-purpose anonymous communication network. It is low-latency to support a wide variety of applications, including web browsing, at a low bandwidth overhead. Nevertheless, this makes Tor vulnerable to traffic correlation attacks. Passive attackers can link communication partners by observing traffic entering and exiting the Tor network. Based on a review of the literature, there are two main solutions for this issue: either introduce significant delays to destroy timing patterns or add a large amount of cover traffic to hide real communications. These solutions, however, either require the deployment of new infrastructure or impose a high overhead on the users and the network.

This thesis focuses on the problem of traffic correlation attacks and proposes LoopTor. LoopTor is a medium-latency communication scheme built on top of Tor that covers user data by establishing a loopback connection between the Tor client and an onion service set up by the client itself. This connection is used to send padding traffic together with the real data so that the observable pattern remains a constant rate, regardless of whether the user has active communications or not. Different implementations of LoopTor are proposed and compared. To test the performance, a Tor network is simulated and traffic is captured both with and without real data being sent. Machine learning classifiers try to distinguish both situations.

Experimental results for the classification task show that the implementations of LoopTor that set a limit for the rate at which data is sent to the Tor network make it impossible for some classifiers to identify communications. Nevertheless, other classifiers are still capable of detecting patterns. Further research is needed to identify other factors that could strengthen the effectiveness of LoopTor and whether the conclusions of the performed simulations apply to the live Tor network.

Keywords: Anonymity, Onion Routing, Tor Network, End-to-end Traffic Correlation Attacks, Global Passive Adversary, Unobservability, Sender-Receiver Unlinkability, Privacy, Secure Communications, Machine Learning Classification.

Resum

Tor és una xarxa de comunicacions anònimes multipropòsit. És de baixa latència per tal de donar suport a una àmplia varietat d'aplicacions, inclosa la navegació web, introduint un baix sobrecost d'amplada de banda. Tanmateix, això, fa que Tor sigui vulnerable a atacs per correlació de tràfic. Un atacant passiu pot relacionar els dos integrants d'una comunicació observant el tràfic que entra i surt de la xarxa Tor. Segons la literatura consultada, hi ha dues solucions habituals per a aquest problema: introduir retards significatius per tal de destruir els patrons temporals o afegir una gran quantitat de tràfic de cobertura per ocultar les comunicacions reals. No obstant això, aquestes solucions, o bé requereixen el desplegament de nova infraestructura o bé imposen una sobrecàrrega elevada als usuaris i a la xarxa.

Aquesta tesi se centra en el problema dels atacs per correlació de tràfic i proposa LoopTor. LoopTor és un sistema de comunicacions de latència mitjana sobreposat a Tor que busca amagar les comunicacions mitjançant l'establiment d'una connexió bucle entre el client Tor i un servei ocult ofert pel propi usuari. Aquesta connexió s'utilitza per enviar tràfic de farciment juntament amb les dades reals de manera que el patró observable sigui sempre un ritme constant, independentment de si l'usuari té o no comunicacions actives. Es proposen i es comparen diferents implementacions de LoopTor. Per avaluar el funcionament, se simula una xarxa Tor i es captura tràfic amb i sense que s'enviïn dades reals. Diversos classificadors d'aprenentatge automàtic intenten distingir ambdues situacions.

Els resultats experimentals en la tasca de classificació mostren que les implementacions de LoopTor que estableixen un límit per a la velocitat amb què s'envien les dades a la xarxa Tor fan impossible que alguns classificadors detectin comunicacions. No obstant això, altres classificadors encara són capaços de detectar patrons. És necessària més investigació per a identificar altres factors que puguin reforçar l'eficàcia de LoopTor i si les conclusions de les simulacions realitzades apliquen a la xarxa Tor real.

Paraules clau: Anonimat, Encaminament Ceba, Xarxa Tor, Atacs per Correlació de Tràfic Extrem a Extrem, Adversari Passiu Global, Inobservabilitat, Desvinculació Emisor-Receptor, Privacitat, Comunicacions Segures, Classificació per Aprenentatge automàtic.

Resumen

Tor es una red de comunicaciones anónimas multipropósito. Es de baja latencia para dar soporte a una amplia variedad de aplicaciones, incluida la navegación web, introduciendo un bajo sobrecoste de ancho de banda. Sin embargo, esto hace que Tor sea vulnerable a ataques por correlación de tráfico. Un atacante pasivo puede relacionar los dos integrantes de una comunicación observando el tráfico que entra y sale de la red Tor. Según la literatura consultada, hay dos soluciones habituales para este problema: introducir retrasos significativos para destruir los patrones temporales o añadir una gran cantidad de tráfico de cobertura para ocultar las comunicaciones reales. Sin embargo, estas soluciones, o bien requieren el despliegue de nueva infraestructura o suponen una sobrecarga elevada a los usuarios y la red.

Esta tesis se centra en el problema de los ataques por correlación de tráfico y propone LoopTor. LoopTor es un sistema de comunicación de latencia media sobrepuesto a Tor que busca ocultar las comunicaciones mediante el establecimiento de una conexión bucle entre el cliente Tor y un servicio oculto ofrecido por el propio usuario. Esta conexión se utiliza para enviar tráfico de relleno junto con los datos reales de manera que el patrón observable sea siempre un ritmo constante, independientemente de si el usuario tiene o no comunicaciones activas. Se proponen y se comparan diferentes implementaciones de LoopTor. Para evaluar el funcionamiento, se simula una red Tor y se captura tráfico con y sin que se envíen datos reales. Diversos clasificadores de aprendizaje automático intentan distinguir ambas situaciones.

Los resultados experimentales en la tarea de clasificación muestran que las implementaciones de LoopTor que establecen un límite para la velocidad con que se envían los datos a la red Tor hacen imposible que algunos clasificadores detecten comunicaciones. Sin embargo, otros clasificadores todavía son capaces de detectar patrones. Es necesaria más investigación para identificar otros factores que puedan reforzar la eficacia de LoopTor y si las conclusiones de las simulaciones realizadas aplican a la red Tor real.

Palabras clave: Anonimato, Encaminamiento Cebolla, Red Tor, Ataques por Correlación de Tráfico Extremo a Extremo, Adversario Pasivo Global, Inobservabilidad, Desvinculación Emisor-Receptor, Privacidad, Comunicaciones Seguras, Clasificación por Aprendizaje Automático.

Dedicated to everyone in the fight against Covid-19.

Acknowledgements

It fills me with unspeakable joy to express my gratitude to everyone who contributed to the completion of this thesis and the successful accomplishment of my Bachelor's Degree.

First and foremost, I would like to express my deepest appreciation to my supervisors, Professor Carmela Troncoso and Professor Oscar Esparza, who directed my thesis. I would also like to thank Dr. Wouter Lueks for guiding me throughout the project. My appreciation extends to all the faculty members of ETSETB and EPFL whom, with their wisdom, support, advice, encouragement, and constructive criticism, have inspired me during my academic life, enriching my growth both as a person and as a student.

Finally, I would like to thank my family and friends for their unconditional support. Without them, I would have never been able to complete this journey successfully.

Revision history and approval record

Revision	Date	Purpose
0	15/04/2020	Document creation
1	18/06/2020	Document completion
2	21/06/2020	Document revision
3	27/06/2020	Document approval
4	28/06/2020	Document submission

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Jaume Planas	jplanas0909@gmail.com
Oscar Esparza	oscar.esparza@entel.upc.edu
Carmela Troncoso	carmela.troncoso@epfl.ch
Wouter Lueks	wouter.lueks@epfl.ch

Written by:	
Date	June 27, 2020
Name	Jaume Planas
Position	Project Author

Supervised by:		**Reviewed and approved by:**	
Date	June 17, 2020	**Date**	June 27, 2020
Name	Carmela Troncoso	**Name**	Oscar Esparza
Position	Project Supervisor	**Position**	Project Supervisor

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> *"Begin at the beginning," the King said, gravely, "and go on till you come to the end; then stop."*
>
> – Lewis Carroll, *Alice in Wonderland*

1.1 Project background

Anonymity, described by the Cambridge English Dictionary as "the situation in which someone's name is not given or known" [1], has been an object of debate for psychologists, lawmakers and society in general all along history.

Derived from Ancient Greek's ἀνωνυμία (anōnumía), anonymity can be a consequence of the loss of information, like happens with anonymous masters from bygone eras, whose work is still appreciated today but their names are long forgotten. Nevertheless, it is not uncommonly an act of free will.

There are a lot of reasons why someone may want to hide their identity, and the one that typically comes to our minds is to avoid legal prosecution. Everyone is familiar with concepts like DNA or fingerprints, and how murderers and robbers will go to the extremest lengths to hide their acts from law enforcement. Despite the fact that most crimes are typically solved, others are still a mystery, and infamous murderers like *Jack the Ripper* or *The Zodiac Killer* remain anonymous to haunt the imagination of the generations to come.

The idea of a perfect murderer is also recurrent in fiction, and like in real life, the motivations behind the crimes are completely diverse. From the aesthetic intellectual exercise poorly executed by two brilliant Harvard students who want to prove that it is possible to commit a crime that cannot be solved in Patrick Hamilton's *Rope* to the passionate greed that leads a former tennis player to blackmail an old colleague into attempting an assassination on his unfaithful and extremely wealthy wife, immortalized by Grace Kelly in Alfred Hitchcock's *Dial M for Murder*, where the hitman ends up being the victim in what appears to be a perfect crime. For some characters, even giving their own life seems a reasonable price to pay in their pursuit of perfection. This is the case of Justice Wargrave in Agatha Christie's *And Then There Were None*, whose own death is the final sacrifice that he willingly makes to create an unsolvable mystery.

But anonymity is not only pursued by criminals, and we can find examples in the most diverse fields that prove the opposite. Hiding the identity and using secret

symbols was fundamental for early Christians to survive Roman persecution until finally legalized by Emperor Constantine in 313 AD; even some of the works from that time that compose the Bible remain anonymous. Later on, with society still riddled with prejudice, many women writers were forced to use male pseudonyms to have their works published, some of the most celebrated being Mary Ann Evans (George Eliot), Amantine Lucile Aurore Dupin (George Sand), Louisa May Alcott (A.M. Barnard) or The Brontë Sisters (Currer, Ellis and Acton Bell). Still nowadays, some artists and writers avoid signing their work to avoid fame or worship.

Living in the middle of the digital era, one could argue that today anonymity plays a bigger role than ever. It was french psychologist Gustave Le Bon in his 1895 work *The Crowd: A Study of the Popular Mind* [2] who described anonymity as the first step towards Deindividuation. According to Le Bon, anonymity prevents people from being isolated or identified, and this leads to a feeling of being untouchable and to a loss of a sense of personal responsibility. A century later, we cannot think of a better illustration of that behavior than the one we get by browsing social networks like Twitter, where we are doomed to encounter millions of users that believe that using a pseudonym makes them unidentifiable, and in consequence uncountable for their actions and comments. Although this sense of anonymity is mostly based on the lack of knowledge of the fact that in most cases the identification is possible, there are some situations in which real anonymity can almost be obtained by expert users. In some contexts, like online voting, to protect this anonymity is fundamental.

One of the techniques that make online anonymity possible is onion routing. Onion routing is a system for anonymous communication in which messages are encapsulated by several layers of encryption, each of which is removed at a different node until the final destination is reached. By doing that, the sender remains anonymous, and each intermediary only knows the location of the immediately preceding and following nodes. The most famous onion routing network is Tor.

But even secure environments like the Tor network have weaknesses. One of the most relevant are end-to-end traffic correlation attacks. The idea behind these attacks is simple, instead of attempting to decrypt the content of packets, an attacker who can observe both ends of the communication can look for patterns in the traffic to match outgoing and incoming data in order to identify users. Tor assumes that a global passive adversary (GPA), capable of eavesdropping anywhere in the network, is not realistic, and that it is very difficult for an attacker to observe both endpoints of the communication. Nevertheless, this is not always true. For example, when both communicating parties reside within the same country or network.

To fight this vulnerability, the Security and Privacy Engineering laboratory (SPRING) at the École Polytechnique Fédérale de Lausanne (EPFL) proposes *LoopTor* [3]. LoopTor is a medium-latency communication scheme built on top of Tor that protects users against end-to-end traffic correlation attacks at the cost of increasing latency and decreasing throughput. It is designed with two main goals to ease deployability: introduce minimal changes to the infrastructure and impose the minimal cost in terms of bandwidth or processing for the onion routers.

Theoretically, LoopTor would ensure that an attacker always observes a constant rate of packages entering the Tor network without any distinguishable pattern. This would make it impossible to perform end-to-end traffic correlation attacks. Nevertheless, the practical implementation shows that machine learning classifiers are still capable of identifying when data is being sent with an accuracy close to 100%.

1.2 Statement of purpose

The objective of this thesis is to evaluate the LoopTor implementation, introduce several modifications and assess how they affect the traceability of the user.

This project was carried out during the Spring semester 2020 at the School of Computer and Communication Sciences of the École Polytechnique Fédérale de Lausanne (EPFL). Additionally, it was supervised by the Barcelona School of Telecommunications Engineering (ETSETB) of the Universitat Politècnica de Catalunya (UPC). Furthermore, this project contributed to the Security and Privacy Engineering laboratory (SPRING) research group from EPFL. The work plan detailing the process that has been followed to complete this thesis as well as a report of the incidences that have occurred can be found in Appendix A.

1.3 Specifications and security goals

LoopTor aims to provide security against passive end-to-end traffic correlation attacks. The security properties, inspired by the formal definitions from AnoA [4], that LoopTor offers are:

Sender-Receiver Unlinkability: Unauthorized parties should not be able to link senders and receivers. This means that an adversary cannot determine if two users are communicating with each other.

Sender unobservability: Whether or not senders are sending data should be hidden from a passive attacker. This means that an adversary cannot decide whether a specific sender is communicating with any receiver.

Receiver unobservability: Whether or not receivers are receiving data should be hidden from a passive attacker. This means that an adversary cannot decide whether a specific receiver is communicating with any sender.

If features of sender/receiver unobservability are achieved, it means that a passive adversary cannot determine if a user is communicating with *any* other user. As a result, it cannot determine if two specific users are communicating with each other, guaranteeing sender-receiver unlinkability.

1.4 Document outline

This thesis is organized in the following way: After this introduction, the reader will be presented in the next chapter with a theoretical introduction to the Tor Network and end-to-end traffic correlation attacks and defenses. With this background, the reader should be able to follow the remainder of the thesis. Chapter 3 describes the methodology followed during the development of the project, while Chapter refres presents the results of the different experiments that have been made. These results are discussed and analyzed in Chapter 5. Afterwards, Chapters 6 and 7 analyze the economical, social and environmental implications of this research. The document closes with Chapter 8, which summarizes the important findings and sets up the basis for future research in this field.

Chapter 2

Theoretical framework

Learn continually. There's always "one more thing" to learn.

– Steve Jobs

This chapter establishes the theoretical framework upon which this thesis is based and provides the reader with the necessary knowledge to understand the following chapters. The first part of the chapter introduces the reader to the environment in which all the experiments will be developed: onion routing networks, in particular Tor, and details the threat to the anonymity that end-to-end traffic correlation attacks present. The second part presents the adversaries that will be used to evaluate whether the presented solutions are capable of solving the problem: correlation techniques and machine learning binary classifiers. Finally, previous research on Tor vulnerabilities and traffic correlation attacks is reviewed in order to identify related work that can be relevant to this thesis.

2.1 Onion routing

Developed by the U.S. Naval Research Laboratory in the mid-1990s to protect U.S. intelligence communications online, *Onion routing* is a design for a low-latency system to support private anonymous communications over a public network. It is based on the establishment of bidirectional connections that are resistant to both eavesdropping and traffic analysis [5]. As opposed to conventional networks where socket connections are made directly to the responding device, in Onion routing the connection is established through a sequence (circuit) of nodes called Onion Routers (OR). Typically, three different OR are chosen to build a circuit: an entry/guard node, a middle node and an exit node [6].

Each client runs a software called Onion Proxy (OP). The function of an OP is to route traffic through the Tor network to a Service Provider (SP) anonymously. To do that, it first builds a circuit choosing 3 nodes and connecting to each of them sequentially. All communications are made over TLS [7] sessions established between two consecutive nodes in the circuit. All circuits have to follow 4 guidelines [8]:

- The same node cannot be chosen twice for the same path.
- Only one router can be chosen in a given /16 subnet or family.
- Non-running or non-valid routers cannot be chosen.

- If Guard nodes are used, the first node of a circuit must be selected from the set of guards. Guard nodes are further explained in Section 2.1.2.

Figure 2.1 shows an example of a network based on Onion routing.

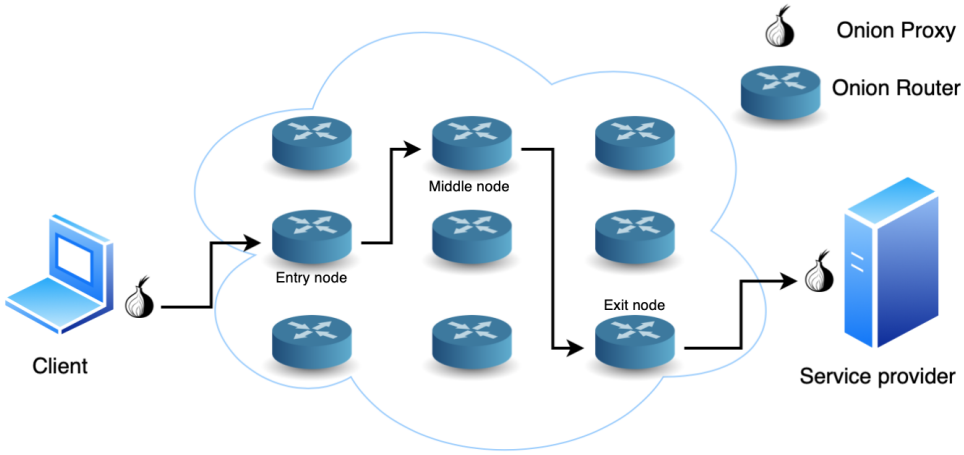

Figure 2.1: Onion routing circuit

During the set up of the circuit, the user exchanges a key with each of the nodes. Once the circuit is complete, the user starts sending data to the first node of the circuit. Analogously to the layers of an onion, the data is sent by the client with a layer of encryption for each node. When the message reaches an OR, the OR removes its corresponding layer by decrypting it with its key and forwards the data with the remaining layers of encryption to the next hop. This process is repeated at each node until all layers have been removed. Then, the exit node forwards the raw data to its destination. This ensures that none of the nodes are aware of both the source and the destination. The entry node knows who is accessing the network but can see neither the raw data nor the destination while the exit node knows the destination and sees the raw data, but does not know who sent the message. The intermediate nodes only know their predecessor and successor on the circuit but are completely unaware of both the sender, the receiver or the content of the data [5].

Figure 2.2 [9] is a representation of the several layers of encryption that are used in Onion routing networks.

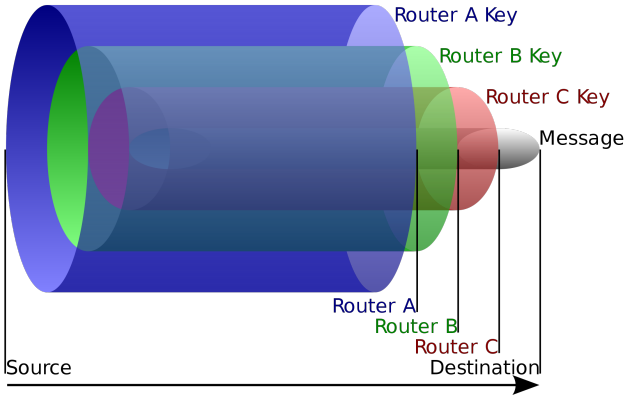

Figure 2.2: Layers of encription in onion routing networks

2.1.1 Tor

Tor is the most famous and widely used implementation of the concept of onion routing. Initially released on 20 September 2002, it includes some significant improvements to the original Onion routing proposal such as congestion control, perfect forward secrecy and the inclusion of onion services [10]. The main goal of Tor, and Onion routing implementations in general, is to protect against attackers trying to link communication partners or link multiple communications to or from a single user. Within this main goal, however, several aspects are crucial in determining Tor's evolution: [11]

- **Deployability:** Tor must be deployed and used in the real world. In consequence, it cannot be expensive to run and it must be easy to implement.
- **Usability:** In order to have increase anonymity, it is fundamental for Tor to have a high number of users, and in consequence, it must be easy to use and implementable on all platforms.
- **Flexibility:** The protocol must be flexible and well-specified so that Tor can serve as a test-bed for future research.
- **Simple design:** Tor aims to deploy a simple and stable system that integrates the best-accepted approaches to protect anonymity.

The design of Tor is based on two fundamental elements, cells and circuits.

Cells

The communication between ORs and between an OR and an OP is done via TLS [7] connections with ephemeral keys to conceal the data with perfect forward secrecy. This also provides integrity to the data by preventing an attacker from modifying it or impersonating an OR. All traffic is sent through these connections in fixed-size cells of 512 or 514 bytes. Each cell is composed of a header and the payload. Since TLS connections allow multiplexing of several circuits, the header of each cell includes a circuit identifier (circID) that indicates to which circuit the cell belongs. Circuit identifiers are connection-specific: each circuit has a different circID on each OP/OR or OR/OR connection [11]. The header also includes a command that describes which action has to be done with the cell's payload. According to their command, cells are classified into control cells and relay cells.

Control cells are cells that have to be interpreted by the node that receives them. According to their command, control cells can have 5 different functions, as shown in Table 2.1. Figure 2.3 presents the structure of a control cell.

Table 2.1: Control cells classification

Cell	Function
Create/d	Create a new circuit and acknowledge
Destroy	Terminate a circuit
Padding	Keep alive
Padding Negotiate	Padding negotiation
Netinfo	Information about time and address

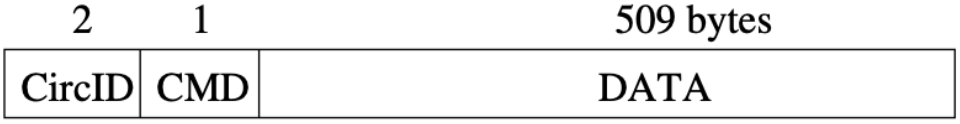

Figure 2.3: Structure of a control cell

On the other hand, **relay cells** carry end-to-end stream data. They include an additional header at the beginning of the payload which contains a streamID that identifies a specific stream multiplexed into a circuit, an end-to-end checksum for integrity checking, the length of the relay payload and a relay command. According to this command, there are several kinds of relay cells (Table 2.2). The relay header and the relay cell payload are encrypted or decrypted together as the relay cell moves along the circuit, using the 128-bit AES [12] cipher in counter mode to generate a cipher stream [8, 11]. Figure 2.4 presents the structure of a relay cell.

Table 2.2: Relay cells classification

Cell	**Function**
Relay data	Data traveling through the stream
Relay begin	Request to open a stream
Relay end	Request to close a stream
Relay teardown	Request to close a broken stream
Relay connected	Inform the OP that a relay begin has succeeded
Relay extend/ed	Extend the circuit by a hop and acknowledge
Relay truncate/d	Tear down only part of the circuit and acknowledge
Relay sendme	Congestion control
Relay drop	Implement long-range dummies

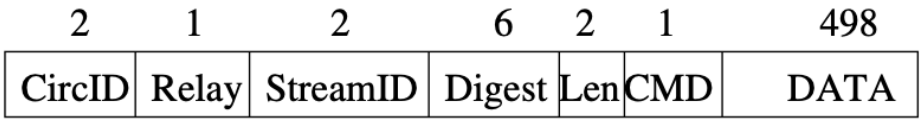

Figure 2.4: Structure of a relay cell

Circuits

In Tor, circuits are created one hop at a time by request of an OP. To start the process, the OP (Alice) sends a *create* cell to the first OR in the path to reach a server. This cell contains an unused CircID and the first half of the key exchange handshake. After receiving it, the OR responds with a *created* cell containing the other half of the key exchange. To extend a circuit, the OP sends a *relay extend* cell instructing the last OR in the circuit to send a *create* cell to the following hop [8]. When a *created* cell is received by the last node, it is transformed into an *extended* cell and sent back to the OP. This process is repeated until the final destination is reached. Figure 2.5 represents the setup process of a circuit. At each

hop, a new encryption key is generated which will only be known by the OP and the corresponding OR. At the end of the process, the OP will be able to encrypt the data with as many layers as ORs constitute the circuit. Tor typically uses the TAP Handshake, which is based on the Diffie-Hellman key exchange [13] (Equation 2.1).

$$(g^a \mod p)^b \mod p = (g^b \mod p)^a \mod p \tag{2.1}$$

One of the main improvements that Tor introduced to the original Onion routing design [5] was the possibility to send multiple TCP streams over a single circuit. This meant a great improvement in terms of performance, since it reduced significantly the frequency of a complex procedure like the circuit establishment. Despite this, to ensure anonymity OPs close old circuits when they no longer have open streams and periodically build new ones [11].

To terminate a circuit, the OP sends a *Destroy* cell which is transmitted through all the ORs of the circuit (Figure 2.6).

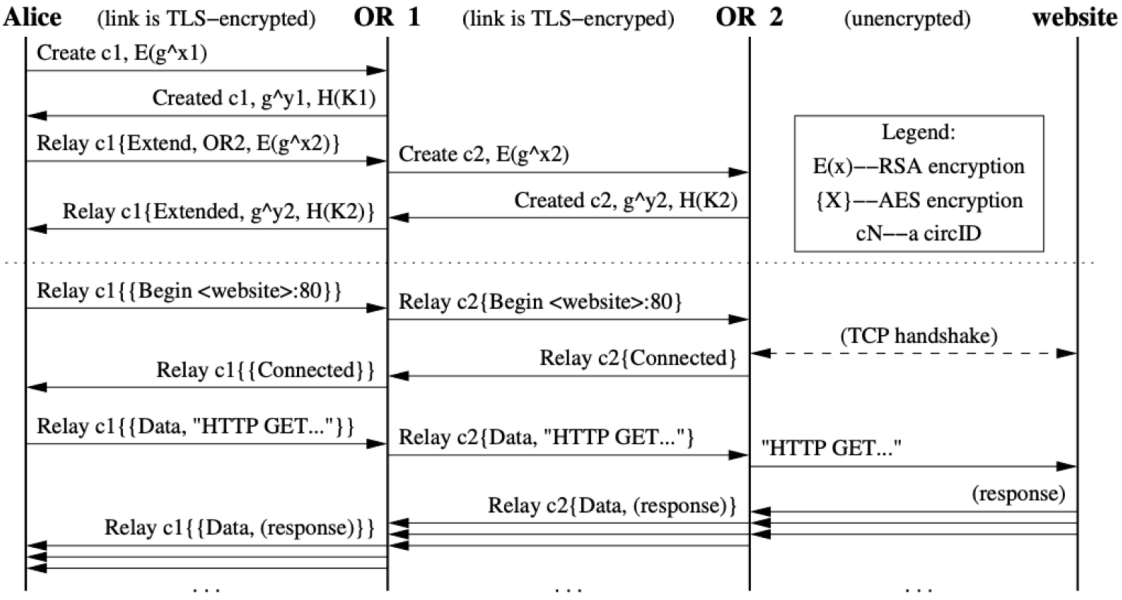

Figure 2.5: Establishment of a two-hop circuit

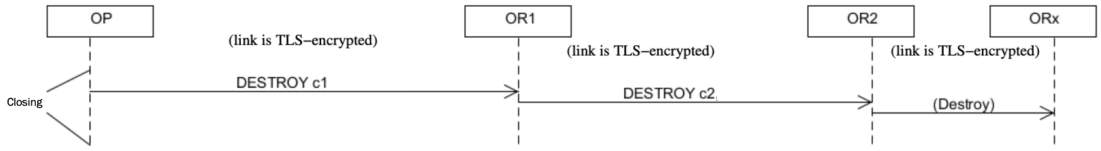

Figure 2.6: Termination of a two-hop circuit

Onion services

Originally known as *hidden services*, onion services are TCP services offered by servers that, unlike normal servers, do not require the user to know an IP address and a port to connect. These servers are configured to only allow inbound connections through the Tor network. To allow clients to access to one of these services,

Tor identifies onion servers with a *.onion* address and establishes a connection procedure between the client and the server through a *rendezvous point* (RP). With this procedure, Tor users can connect to onion services each without knowing the other's network identity. This also protects against DDoS attacks [14] by providing access control to the service provider; the server filters the incoming requests thus no longer being vulnerable to a flood of connections. To ensure robustness, the service provider must be able to migrate it across ORs. This allows keeping a long-term pseudonymous identity even in the presence of router failure [11, 15].

Onion services are advertised on several onion routers, known as introduction points. When the client wants to connect to an onion service, an OR is chosen to act as the rendezvous point and notifies one of the introduction points. If the server accepts the connection, it connects to the rendezvous point and starts offering the service to the client. The detailed procedure of how a connection to an onion service trough a rendezvous protocol is established can be found in Appendix B.

2.1.2 End-to-end traffic correlation

Onion routing is an effective tool to preserve the anonymity of its users. Nevertheless, it is designed assuming that an attacker cannot observe both the entry and the exit nodes simultaneously. As a result, anonymity can be compromised by performing an *end-to-end traffic correlation* or *traffic confirmation* attack. In these attacks, the adversary observes both the traffic that a user is sending to the Tor network and the one another user is receiving shortly thereafter. If the volume and timing of the traffic on both points are highly correlated, it means that there is an active connection between the two clients. The main traffic correlation techniques will be reviewed in the following chapter.

Traffic correlation attackers can be classified into two groups: **Active attackers**, which try to enhance the attack by deleting, modifying, adding or replaying data and **passive attackers**, that only observe traffic without interfering with it. Although active attacks make correlation easier to establish with fewer data, they are generally difficult to carry out because the attacker has to be able to manipulate traffic without being detected. In passive attacks, however, the adversary only needs to observe traffic entering and leaving the Tor network to perform the attack [16, 17].

Since an attacker has to control or observe both the entry node and the exit node of a circuit, the probability of a circuit being compromised if an attacker controls C out of N nodes can be estimated by Equation 2.2. This estimation assumes that all nodes are randomly selected, which is only partially true. Some restrictions apply when choosing the nodes of a circuit, as explained in previous sections.

$$\frac{C^2 - C}{N^2 - N} \tag{2.2}$$

Provided that a user is constantly creating new circuits, even controlling a low number of nodes, a compromised circuit will eventually be established and anonymity will be broken.

If the attacker can only observe the entry link, an end-to-end traffic correlation attack is not possible. Nevertheless, the attacker can still carry out a *website fingerprinting attack* to identify which webpages the user is accessing. To perform this attack, the adversary captures sequences of packets received by the client and

uses machine learning classification to determine the webpage from which they came from [18, 19].

Traffic correlation attacks against onion services

A particular case of traffic correlation attack is the one that is performed to capture the IP address of the provider of an onion service. In this case, the attacker assumes the role of the client instead of the external attacker mentioned before, and only requires the control of the exit node.

As described in [8], onion services (OS) publish contact information on a database distributed on the Tor network. Subsequently, any client who wants to use the service gains access to the database and uses that information to contact with the OS through a rendezvous point, where both of them establish a Tor circuit, to avoid being traced. From this position, the adversary can reveal the OS identity following 4 steps, represented in Figure 2.7 [20–22]:

1. A node controlled by the attacker is selected as an entry node on behalf of the OS provider.
2. The attacker, acting as a client, establishes a connection with the OS and designs a controlled node as RP. This connection is associated with a cookie created by the client which is also transmitted to the OS. The OS uses the RP to distinguish between connections.
3. When the service connects with the malicious RP, the RP can associate this connection initiated by the OS with the connection started by itself, since it contains the same cookie. The RP sends an anomalous pattern of traffic towards the service provider, followed by a petition to close the connection.
4. If the entry node receives the same pattern of packages sent by the RP followed by the petition to shut down the connection, the entry node controlled by the attacker associates the received connection from the OS provider with the rendezvous node connection to the OS and anonymity is broken.

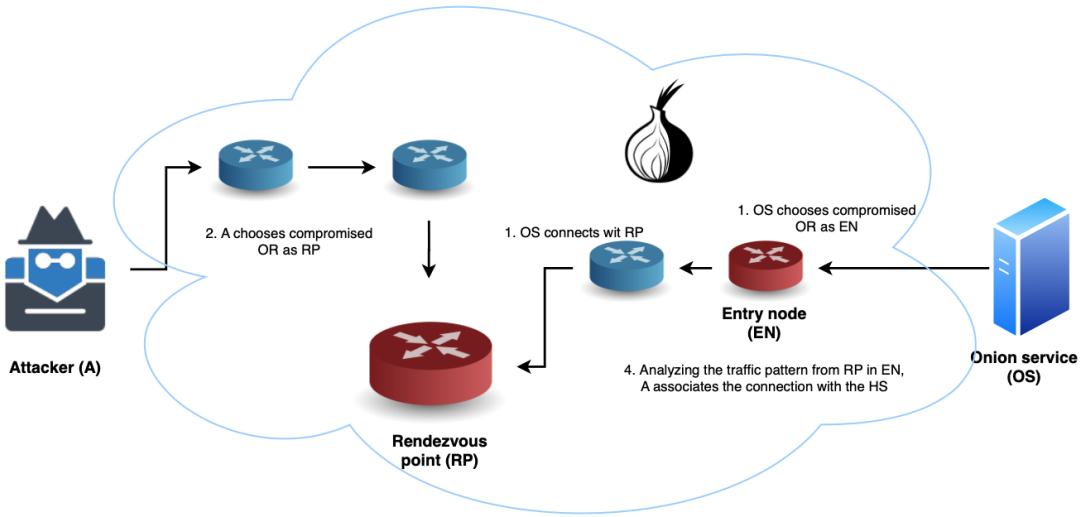

Figure 2.7: Traffic correlation attacks against an onion service

Defense against traffic correlation attacks

In general, there are two different approaches in the defense against traffic correlation attacks, either make it difficult for the adversary to correlate the traffic at the entry node with the traffic at the exit node or prevent the attacker from being able to perform such attack at all. Tor takes this second approach by introducing the concept of entry guards [22]. In Tor, users always select their entry node from a small set that never changes. As a result, it is more difficult for the attacker to control the first node. When guard nodes are selected, only two outcomes are possible: either one of the guard nodes is compromised, which makes all circuits vulnerable, or none of them are. In this second situation, the attacker cannot compromise any of the circuits. Since the number of corrupted nodes is small compared with the total number of nodes in the network, the likelihood of choosing a compromised node as a guard is also low [17, 23]. Nevertheless, this does not solve the problem. It only makes attacks more expensive, since a higher number of nodes needs to be controlled by the attacker. Besides, entry guards do not protect against attackers observing links, like a GPA.

On the other hand, some networks try to address the problem taking the first approach, and a wide range of techniques try to cover or break any identifiable patterns. This can either be done by adding significant delays in the communications to destroy timing patterns, as is the case of mix networks such as Mixmaster [24] or Mixminion [25] or by adding a large amount of cover traffic to hide real communications, as is the case with projects such as Loopix [26] or Tarzan [27]. These solutions, however, typically impose a high overhead on the users and the network and make scalability almost impossible. Other proposals such as Dissent [28] or especially Riposte [29] and Vuvuzela [30] manage to have linear costs and thus can have a higher number of users, but at the cost of a higher latency or the deployment of new infrastructure.

2.2 Traffic correlation techniques and classification

In the previous section traffic correlation attacks have been introduced, and some examples have been given regarding how the attacker gets to a situation that enables gathering information from the observed traffic. Once this situation is achieved however, the attack is not yet complete. The second step, arguably the most important, generally involves analyzing and correlating the obtained data to unveil, totally or partially, the identity of the victim. While a wide variety of techniques exist for an attacker to set up a situation where it is possible to observe anonymous traffic, most of the studies rely on the same techniques to correlate the obtained data. The first part of this section introduces the most common correlation techniques used in end to end traffic correlation attacks.

In the second part, machine learning classification is introduced. Classifiers play a fundamental role in this thesis because they are used to determine whether the implemented solution is capable of completely hiding patterns in the data sent to the Tor network. Emphasis will be made on the *random forest classifier*, since it is the one that will be used in the following chapters.

2.2.1 Traffic correlation techniques

The most simple approach, known as *packet counting*, consists in counting the number of packets in a traffic flow. Comparing the number of packets for a given stream with the number of packets present into each stream from a set allows an adversary to find out which stream from the set is potentially the same as the observed one. Nevertheless, external facTors like latency or throughput can alter the number of packets observed for the same stream entering and exiting the network. Therefore, simply checking the equality between the number of packets of two streams could be inaccurate. To address this, the similarity between the packet count x from stream X and the packet count y from stream Y is typically measured by the distance (Equation 2.3). The smaller the distance between x and y, the greater the similarity between the streams X and Y [31, 32].

$$d(x, y) = \sqrt{(x - y)^2} \tag{2.3}$$

A more complex approach is to use the *cross-correlation coefficient*, which measures the average slope obtained when plotting three or more (x, y) points on a graph. This technique is used to measure the similarity between two series as a function of the delay of one relative to the other. This technique can be applied to streams of network traffic by extracting a sequence of values from the streams and comparing it with the sequences obtained from other streams in the network. The sequence of values for a given stream is extracted by slicing the stream into small windows of size W and counting the number of packets received during that window size. This process is repeated for the duration of the stream. The correlation between the two streams, x and x'. is calculated with Equation 2.4.

$$r(d) = \frac{\sum((x_i - \mu)\left(x'_{i+d} - \mu'\right))}{\sqrt{\sum_i (x_i - \mu)^2}\sqrt{\sum_i (x'_{i+d} - \mu')^2}} \tag{2.4}$$

The delay value d is the time that it takes to the stream to transit the network. x_i is the ith packet count of stream x and x'_i is the ith packet count of stream x'. μ is the average of packet counts in stream x and μ' s the average of packet counts in stream x' [31].

For each pair, if both the x and y components have the same value, the slope is 1, meaning that a linear equation perfectly describes the relationship between the two streams being compared. As the values become random, the correlation disappears and the slope approaches 0. A slope of -1 indicates the pairwise values are the inverse of each other, meaning that the increase of the value of one variable results in a decrease of the value for the other variable. The measured rarely reach extreme values (1, 0, -1) but lay somewhere in between. [32] classifies them as shown in table 2.3.

Table 2.3: Categories of cross-correlation coefficient

Correlation strength	$\|r\|$
Strong	0.5 to 1
Medium	0.3 to 0.5
Low	0 to 1.3

Using the same windowing technique for obtaining sequences of data, another option to measure the dependence between the two sequences is through their *Mutual information*. Being X and Y two sequences, the mutual information (Equation 2.5) is a measure of the similarity between the two sequences based on the difference between the actual joint distribution of X and Y and the joint distribution of X and Y if X and Y were independent. $p(x, y)$ is the value from the joint probability distribution for x and y. $p(x)$ and $p(y)$ are the marginal probability distributions for x and y. *log* measures in bits the difference. This bit measure is then weighted. The weighting variable is the joint distribution value [31, 33].

$$I(X;Y) = \sum_{x \epsilon X} \sum_{y \epsilon Y} p(x,y) log(\frac{p(x,y)}{p(x)p(y)}) \tag{2.5}$$

Once again, the bigger the mutual information between the two sequences is, the more correlated they are.

Finally, a different approach can be of relevance in the case of an active attack. If the adversary manages to inject a pattern in the traffic flowing to the victim and is capable of checking the load of a given relay at a given time, it can be determined whether or not the load is correlated with the injected traffic pattern. In this case, correlation is measured as the sum of the multiplications of the template formed by the modulated traffic and the observed latency for every sampled time t. The template formed by the modulated traffic, $S(t)$, takes 1 as value when the corrupted server is sending at sample number t and 0 otherwise. $L(t)$ measures the latency of a given relay and normalized by the mean of all samples can be expressed as $L'(t)$. Given that, the correlation between the injected pattern and the latency of a given relay is expressed by Equation 2.6.

$$c = \frac{\sum S(t) * L'(t)))}{\sum S(t)} \tag{2.6}$$

The correlation index c reflects the strength of the correlation between the two data streams. As in the previous cases, a high correlation index indicates a strong relationship between the two streams whereas a low correlation index reveals that the two streams are independent of one another. A weakness of this technique is that despite being based on time patterns, it doesn't take into account the effect of network latency. Since the modulated traffic travels through the network and will likely be reshaped due to transmission delays or packet loss, time patterns can be altered. Therefore, the attacker will not be able to correlate the observed traffic with the original pattern [32, 34].

2.2.2 Machine learning classification

Machine learning is a branch of artificial intelligence that uses algorithms that improve automatically through experience for purposes such as finding patterns in data to make predictions. In machine learning, a dataset of observations called instances is comprised of several variables called attributes. Unlike *Unsupervised learning*, where the dataset does not include a target attribute and the algorithm tries to find similarities among the groups or some intrinsic clusters within the data, *Supervised learning* is the modeling of said datasets containing *labeled* instances. In supervised learning, instances are represented as x, the set of independent attributes, and y,

the dependent target attribute. If y is a continuous target, the category of modeling is called *regression* whereas if it is discrete, it is called *classification* [35, 36].

Classification

As stated above, classification is an instance of supervised learning in which the program learns from a training set of data containing observations correctly classified into several categories. Once trained, it can identify to which of said categories a new, unlabeled, observation belongs using pattern recognition. Since no algorithm is optimal for all datasets, there is a wide range of classifiers. Each of the uses a different algorithm to map the input data to a specific class. Classifiers are divided into two groups, *Lazy learners* and *Eager learners*.

- **Lazy learners** store the training data and wait until test data is introduced. Classification is done using the most related data in the stored dataset. Some examples of lazy learner classifiers are *nearest neighbor* and *case-based reasoning*.

- **Eager learners** use the training data to build a classification model before getting data for predictions. The algorithm must commit to a single hypothesis that works for the entire space. As a result, the training of an eager learner requires a lot of time, but they are faster in making predictions. Some examples of eager learner classifiers are *Naive Bayes*, *Artificial Neural Network*, *Decision Trees* and *Random forests*.

If the dependent target attribute only has two possible outcomes, the model is called *Binary classification*. Binary classifiers are typically used to evaluate whether the input attributes have some characteristic properties or not, and the categories are called Positive and Negative [36].

Random forest classifiers

Random forest is an ensemble, tree-based, learning algorithm. The forest consists of a set of decision trees trained with randomly selected subsets of training data. To classify a new object, each of the trees in the forest is fed with the input data and gives a classification that is regarded as a vote for the chosen class. Finally, all the votes are aggregated and the one with the most votes is chosen as the final classification.

The advantage of using ensemble algorithms, like the random forest, is that a large number of relatively uncorrelated models, in this case trees, operating as a committee will typically outperform any of the individual constituent models. By training different trees with different subsets of data, random forests correct the habit of overfitting to a training set characteristic of individual decision trees. Moreover, since random trees have methods for balancing error in unbalanced datasets, they are a valid option for binary classifiers [37, 38].

Figure 2.8 presents a representation of the functioning of a random forest classifier.

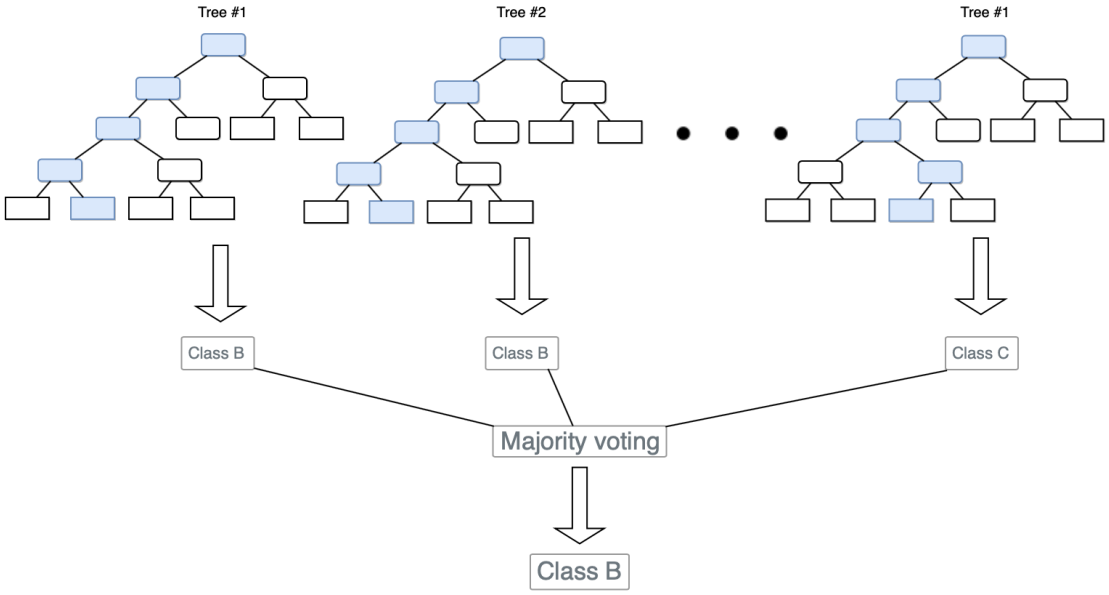

Figure 2.8: Random forest as an aggregation of random trees

To create a random forest classifier, each tree is grown as follows:

1. The N cases in the training dataset are divided at random with replacement, into several groups. Each group will be the training set to grow a tree.
2. If there are M input variables, a number $m << M$ is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
3. Each tree is grown to the largest extent possible without pruning.

The forest error rate depends on two factors:

- The *correlation* between the trees in the forest. The higher the correlation, the higher the forest error rate.
- The *strength* of each tree. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.

Reducing m reduces both the correlation and the strength while increasing it increases both. The ”optimal” range of values for m depends on the dataset [38].

Performance metrics

The performance of a classifier depends greatly on the characteristics of the data to be classified. Several metrics and graphical techniques are used to assess the performance in each situation.

A *confusion matrix* is a representation of the performance of a supervised learning method. A problem with n classes requires a $n * n$ confusion matrix with the rows representing the real class and the columns representing the class predicted by the

classifier. With binary classifiers, a 2x2 matrix is used. In a confusion matrix, TP (true positive) is the number of positives correctly identified, TN (true negative) is the number of negatives correctly identified, FP (false positive) is the number of negatives incorrectly identified as positive, and FN (false negative) is the number of positives incorrectly identified as negatives [35]. A graphical representations of said concepts can be seen in Figure 2.9 [39] and an example of confusion matrix in Table 2.4.

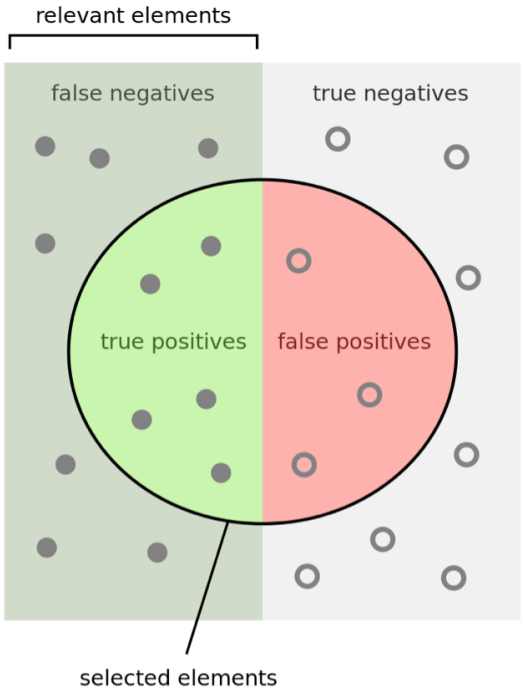

Figure 2.9: Venn diagram of possible classification outcomes

Table 2.4: Confusion matrix

		Prediction outcome		
		p	**n**	**total**
actual value	**p′**	True Positives	False Negatives	P′
	n′	False Positives	True Negatives	N′
	total	P	N	

From the confusion matrix, several metrics can be measured. The simplest is accuracy. Accuracy, as defined in Equation 2.7, is the total number of correct predictions made over the total number of predictions. Although accuracy is very easy

to interpret, it is not very descriptive when used with highly imbalanced datasets. A model may have very high accuracy, but may not obtain high levels of identification of the class that is relevant to predict. Besides, in an imbalanced dataset, a model may misidentify all positive classes and still have high levels of accuracy; besides, pure randomness is not taken into account with the accuracy metric. Accuracy's complementary metric is the error rate (1-accuracy), calculated in Equation 2.8 [35].

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.7}$$

$$\text{Error rate} = \frac{FP + FN}{TP + TN + FP + FN} \tag{2.8}$$

The most common metrics for evaluating the performance of binary classifiers with imbalanced datasets are precision and recall. Precision is the percentage that the model correctly predicts positive when making a decision (Equation 2.9). Recall, also known as sensitivity, is the percentage of positives correctly identified out of all the existing positives (Equation 2.10). An ideal model would have both high recall and high precision, but they are often achieved one at the expense of the other. The F1-score is the harmonic measure of precision and recall in a single measurement (Equation 2.12). The F1-score ranges from 0 to 1, with 1 indicating that both precision and recall are perfect. In some fields of research specificity (Equation 2.11) is also used together with sensitivity [35, 36].

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.9}$$

$$\text{Sensitivity (Recall)} = \frac{TP}{TP + FN} \tag{2.10}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{2.11}$$

$$\text{F1-score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.12}$$

A different approach to compare classifiers is to use the Receiver Operating Characteristic (ROC) curve. The ROC is a graphical representation of the true positive rate (TPR) (Equation 2.13), against the false positive rate (FPR) (Equation 2.14).

$$TPR = \frac{TP}{TP + FN} \tag{2.13}$$

$$FPR = \frac{FP}{TN + FP} \tag{2.14}$$

Figure 2.10 shows ROC curves for different classifiers. A curve close to the top left corner represents that the classifier is succeeding, and a curve along the diagonal reflects a random classification.

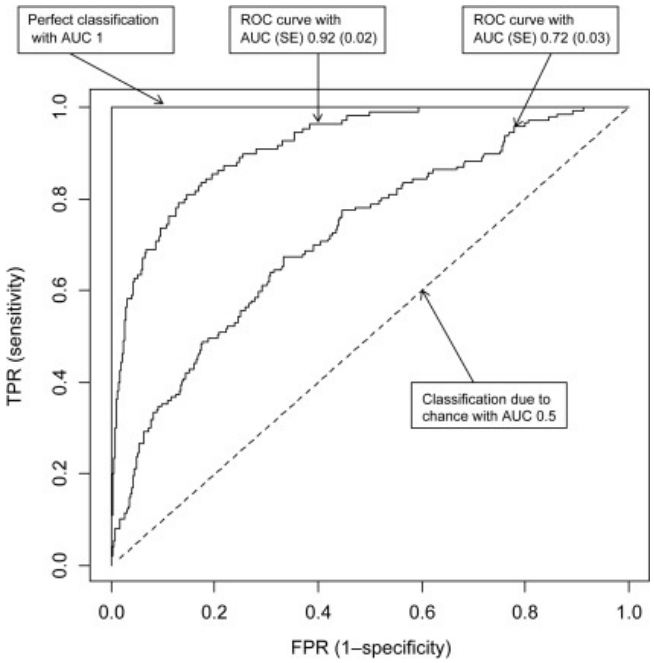

Figure 2.10: Example of ROC curves

Finally, the Area Under the ROC Curve (AUC) also provides a good indication of the performance of the classifier. It is calculated by integrating the ROC curve. An AUC of 1.0 indicates a perfect classification (both sensitivity and specificity are 1.0). In contrast, a test with an AUC of 0.0 is perfectly inaccurate. The line segment from (0,0) to (1,1) has an area of 0.5 and is called the chance diagonal. Tests with an AUC value larger than 0.5 have at least some discrimination ability. This technique is mainly used with binary classifiers, but with multi-class examples, the AUC can be weighted according to the class distribution [40].

2.3 Related work

This section reviews the literature for previous work on the field in order to identify research relevant to this thesis. In the first part, works on generic end-to-end traffic correlation attacks are presented. The second part focuses on applications of said attacks against the Tor network. Finally, a brief summary of the relevant findings closes the section and the chapter.

2.3.1 End-to-end traffic correlation attacks

End-to-end traffic correlation attacks are one of the most commonly analyzed attacks against anonymity systems because of their theoretical capacity to completely unveil the identity of a great number of users. Said attacks have been studied both in high-latency and low-latency networks. In this subsection, some of the main works in this area are presented.

The first approach to passive attacks against low-latency systems is presented at *Passive Attack Analysis for Connection-Based Anonymity Systems* [41]. The attack consists in observing all the active connections of a node in a network. From this position, the researchers identify two possible attacks that could be performed. In

the first, the adversary counts the total amount of messages in each link and compares them. If two different links present a similar count of messages, the attacker assumes that they are connected. The second attack tries to track the establishment of a connection. When the attacker detects a lot of traffic on an incoming connection followed by an increase in traffic on an outgoing connection, it is deduced that both connections belong to the same traffic flow. This work focuses on a single node, but the same attack could be performed by a global adversary to link traffic between two different nodes. Moreover, ignoring the specific details of the security protocol and limiting the observation to both endpoints of the communication, the application of these techniques is an example of an end-to-end traffic correlation attack as described in the previous sections.

A different group of attacks, specific for low-latency networks, are timing attacks. Timing Attacks are presented in *Timing Attacks in Low-Latency Mix-Based Systems* [42] and *Timing Analysis in Low-Latency Mix Networks: Attacks and Defenses* [43]. Both works try to correlate traffic using timing patterns. One of the used techniques consists in dividing the traffic in fixed-size windows and counting the number of messages in each window. One defense against these attacks sends cover traffic to hide traffic patterns. A more advanced approach, called defensive dropping [42], consists in sending cover traffic and dropping it at intermediary nodes. This confuses the adversary, since the amount of traffic that enters and leaves the Tor network is no longer the same. Another option is adaptive padding [43], where the same intermediary nodes add cover traffic to the real data. This randomizes the traffic pattern, making it more difficult to find any correlation.

These techniques treat traffic from each user independently. In contrast, *Dependent Link Padding Algorithms for Low Latency Anonymity Systems* [44] proposes dependent link padding. Dependent link padding joins all incoming traffic flows from different users and generates outgoing traffic flows that transmit with the same time patterns. If a user does not have data to send, a dummy packet is transmitted. As a result, all flows transmit packets, either real or dummy, at the same time with the same timing characteristics, making it impossible for an attacker to match incoming flows to outgoing flows. In some contexts, a maximum delay can be set, but this results in sending more dummy packets.

Another theoretical attack, the *statistical disclosure attack*, is introduced in *Statistical Disclosure Attacks: Traffic Confirmation in Open Environments* [45]. The work assumes that every user communicates with a fixed number of users and that all background traffic is also distributed uniformly. In this situation, an attacker tries to identify the users with whom a specific user is communicating in a mix network. The attacker observes every message entering and leaving the mix and remembers the recipients of messages in every round as candidates for the user's recipients. By observing many rounds, the most likely candidates are revealed, because they receive significantly more messages compared to the uniformly distributed background traffic. Since the assumptions for this attack can seem unrealistic, in *Practical Traffic Analysis: Extending and Resisting Statistical Disclosure* [46] the author includes the possibility to conduct this attack with more complex traffic patterns and without requiring a global adversary. Moreover, the attack can also be used against anonymity systems that do not operate in rounds with mixes. The attacker simulates the rounds with fixed-size time windows and using variable-size batches.

While statistical disclosure attacks consider mixes that forward messages in batches, *The Traffic Analysis of Continuous-Time Mixes* [47] analyses mixes forwarding each message individually. These messages can be randomly delayed. The attacker observes the traffic pattern of an incoming link and determines with a statistical hypothesis if this pattern is repeated on an outgoing link. Comparing the pattern with all links in the network, the whole route of the traffic can be traced. Besides, this system can also be used to correlate traffic at the edges of the network in an end-to-end traffic correlation attack.

The previous attacks require the partial or complete observation of the network. In contrast, *Sampled Traffic Analysis by Internet-Exchange-Level Adversaries* [48] determines that an Internet exchange point (IXP) can use the traffic statistics that it collects to correlate traffic. This attack is not influenced by the timing of the messages and, as a result, cannot be defended by randomly introducing delays.

Finally, a more complex option involving an active attacker is presented in *The Need for Flow Fingerprints to Link Correlated Network Flows* [49]. To perform this attack, the adversary delays packets to include a recognizable pattern. If this pattern is modulated onto traffic entering an anonymity network and recovered from traffic leaving the network, the attack can be used as an end-to-end traffic correlation attack. A defense against said attack is proposed in *Preventing Active Timing Attacks in Low-Latency Anonymous Communication* [50]. The idea is similar to some defenses against passive timing attacks and consists in sending packets over multiple paths through the anonymity network and forwarding them to the following nodes at a specific instant. As a result, if the attacker does not control the majority of the network, the real data will still reach the destination in time.

2.3.2 Attacks against Tor

The works presented so far describe general attacks and defenses against anonymity systems, but none is specific to the Tor network. As previously described, Tor is a low-latency anonymous network that provides user anonymity without significantly decreasing network performance. In order to provide good user experience, Tor keeps communication delays low while preserving the anonymity of its users. However, low Round Trip Time (RTT) comes with a price. Since Tor attempts to optimize the delays in the communications between users, the traffic patterns at one end of the anonymous path are reproduced at the other end. This makes Tor vulnerable to end-to-end traffic correlation attacks. This subsection describes some attacks and defenses specific to the Tor network that have been proposed in the recent literature.

Some basic attacks against the Tor network are presented in *Large Scale Simulation of Tor: Modeling a Global Passive Adversary* [51]. In said article, the authors present the implementation of some of the previously described passive attacks (packet counting, connection start tracking, timing attacks...) in a simulated Tor network to test their effectiveness when performed by a global passive adversary. The findings show that the attacks are very effective when there is a low volume of traffic in the network, not so much when the volume increases.

A more specific end-to-end timing attack against Tor is presented in *Low-Resource Routing Attacks Against Tor* [52]. Here, the adversary analyzes the traffic generated by Tor when creating a connection. By observing this traffic both in an entry node and an exit node, the attacker learns that a connection has been established between

these two nodes. In this situation, this attack breaks anonymity even when the users have not sent any data at all. The attack is tested and succeeds in a laboratory environment.

On the other hand, *Locating Hidden Servers* [22] presents traffic correlation attacks against location-hidden services in Tor. The researchers incorporate timing information to extend the packet counting attack and manage to discover the real IP address of an onion service. This attack is performed against an onion service set up in the Tor network and succeeds.

While [22, 52] prove that passive end-to-end traffic correlation attacks can be successful in the Tor network, *One Cell is Enough to Break Tor's Anonymity* [16] presents an active attack, the tagging attack. Here, a malicious entry node manipulates data sent through Tor and a colluding exit node detects the manipulation when it causes an error in the decryption. This confirms that traffic entering the network at the entry node leaves the network at the exit node, revealing both endpoints of the communication. *A New Cell Counter Based Attack Against Tor* [53] presents another active end-to-end traffic correlation attack against Tor. In this attack, a corrupt entry node inserts a specific signal in the traffic flow. The node generates bursts to encoded to encode the signal, and a colluding exit node recognizes it. A similar active attack is carried out against onion services in *Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization* [54]. All the presented attacks succeed in the performed experiments.

The theoretical concept presented in *Sampled Traffic Analysis by Internet-Exchange-Level Adversaries* [48] finds a practical implementation in *Traffic Analysis Attacks and Defenses in Low Latency Anonymous Communication* [55]. Here, the attacker implements an active end-to-end traffic correlation attack against Tor that consists in correlating network traffic statistics. The attack starts when a user connects to a corrupted server. Then, the server sends a response that is transmitted with a distinguishable traffic pattern. Using traffic management software at the intermediary nodes and the exit node, traffic statistics about connections are gathered and correlated to confirm that the user accessed the malicious server. The author also presents a defense against this particular attack. The defense consists in sending dummy packets with a very small time-to-live (TTL). This dummy traffic alters the statistics and breaks correlation. Since the TTL is very small, intermediary nodes drop the dummy packets and the user never receives them, ensuring that the dummy traffic does not reduce the throughput.

Finally, in *Defending End-to-End Confirmation Attacks against the Tor Network* [17] the author investigates end-to-end traffic correlation attacks against the Tor network and performs an evaluation as to whether they present a real threat to users. As opposed to many of the previously mentioned works, the experiments are made on the live Tor network. Besides, his work proposes a defense based on dummy traffic and examines the level of protection that it can provide. The proposed solution tries not to impose high computational costs on the Tor network so that it does not affect its usability. The experiments find that the proposed defense can protect against some of the proposed end-to-end correlation attacks, which are, according to the author, still a relevant threat.

2.3.3 Summary

After reviewing the mentioned works, it is observed that in the past years, researchers have designed, developed and implemented a wide range of attacks against low-latency anonymous networks such as Tor. Some of the attacks described in the literature are dedicated to low-latency anonymity systems in general while some others are specifically designed to be carried out against Tor. While the first are more theoretical than real and have not been tested on real networks, there are some interesting results from the implementation of attacks against Tor-like networks. The main findings indicate that such attacks can be very effective in specific situations, but remains unknown if they could work against a realistic adversary in the live Tor network. This last element is of great relevance, since it has been widely proved that a powerful adversary that could observe each link of the anonymous network is capable of identifying the sources of anonymous communications by performing end-to-end traffic correlation attacks. Nevertheless, such an adversary cannot exist in a real network.

As said omnipotent adversary does not exist, researchers have been working on reducing the network infrastructure that has to be observed to perform a successful traffic analysis attack. The most recent works are focused on attacks that could be carried out by a more realistic adversary.

As for the defenses, the most typical techniques are delaying traffic, sending cover traffic, defensive dropping or adaptive padding. The results of these implementations are once again mixed, and typically tend to ignore the impact that their use can have on the performance that Tor can provide to its users.

All in all, the literature presented in this section provides a strong theoretical background to the problem of end-to-end traffic correlation attacks. It also proves that some defenses have been presented in previous works, some in the same direction as this thesis, but the effectiveness of both the attacks and defenses against the current size Tor network needs to be further studied and a lot of questions remain unanswered.

Chapter 3

Methodology

> *The man of science has learned to believe in justification, not by faith, but by verification.*
>
> – Thomas Huxley

This chapter presents the methodology that has been followed in this thesis. It starts by providing a general overview of the objectives that LoopTor aims to provide and in which contexts it can be of use. In the next section, the threat against which LoopTor can be used is detailed. This is followed by an analysis of LoopTor's architecture, providing the reader with the most critical design aspects upon which LoopTor is based. The chapter closes with a final section in which all the possibilities that were considered for the implementation of LoopTor are reviewed and the chosen approach is justified.

3.1 Objective and high-level overview

LoopTor is a medium-latency communication scheme built on top of Tor that aims to protect users against end-to-end traffic correlation attacks without requiring the deployment of new infrastructure. The key objective of LoopTor is to ensure that the traffic between the user and its entry node remains constant regardless of the number, if any, of active connections. To avoid saturating the network, LoopTor accepts increasing latency and reducing throughput as the price to enhance privacy [56]. However, there are many applications in which anonymity is fundamental that do not have high bandwidth requirements. This is the case of messaging, e-mail, transactions of digital currencies, and electronic voting [3].

3.2 Threat model

LoopTor assumes a passive adversary aiming to link users to their communications and/or their communication partner(s). This adversary is capable of observing the TLS-encrypted traffic entering the Tor network and attempts to correlate the traffic pattern observed to that exiting from another endpoint. It cannot observe the traffic circulating within the Tor network. Figure 3.1 shows a representation of the presented adversary.

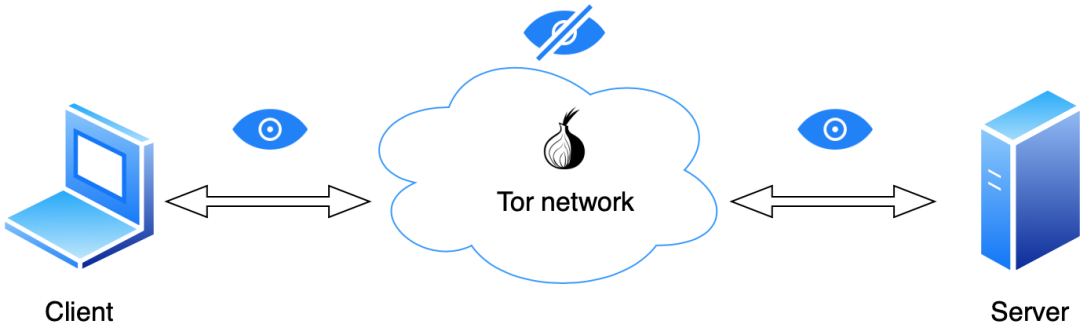

Figure 3.1: Passive observer performing end-to-end correlation attack

3.3 Architecture

The architecture of LoopTor is based on two fundamental pillars, a very low bandwidth link between the user and the Tor network and the generation of dummy traffic to be sent through a loopback connection. This loop traffic ensures that, without help from the network or other users, the link can be easily saturated in both directions. Figure 3.2 shows an example in which a Tor client uses LoopTor to hide a connection with another user. LoopTor throttles the link with the entry guard and fills this link using dummy loopback traffic so that the real communication is covered and undistinguishable [3].

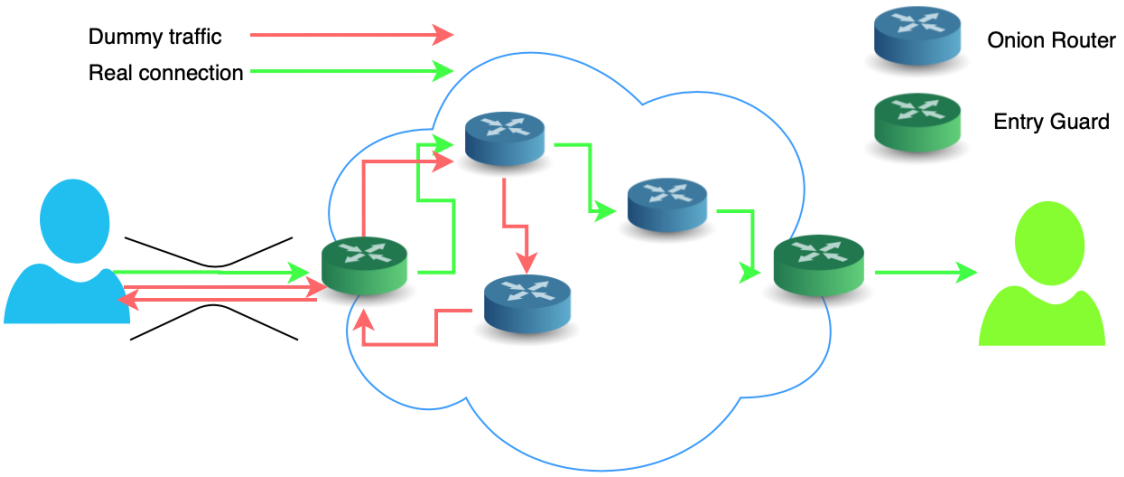

Figure 3.2: Schematic of the use of LoopTor

When a user connects to another user via the Tor network, the exit node creates a new TCP connection to that user. This means that a passive adversary can determine when there is communication by observing the number of connections. To prevent this, every LoopTor user sets up an onion service, and only accepts incoming connections on this OS. As a result, all incoming traffic is routed through the same TLS connection. LoopTor creates the loop connection by connecting to its own OS and makes sure to use the same TLS connection to the entry guard.

To simulate a slow connection between the user and Tor, LoopTor uses *Traffic Control (TC)* [57]. TC is a utility program used to configure the packet scheduler

of the kernel. One of the main applications of TC is to drop packets. LoopTor uses this feature to set a limit to the incoming bandwidth by triggering TCP's congestion control system. When the rate is too high, TC drops packets so that they have to be retransmitted, thus reducing the effective throughput and forcing the sender to transmit at a lower rate. This reduction in the throughput and the delay it introduces allows keeping a constant rate of packages sent through the loopback connection without having to saturate the Tor network with huge amounts of traffic. Since LoopTor sends an equal-sized reply for every real incoming message and all communication with the entry guard are encrypted, an attacker should always see an independent number of indistinguishable messages regardless of the number of real or dummy messages being sent, and will not be able to identify any patterns in the connection between the user and the entry node [3]. Figure 3.3 provides a detailed view of the TLS connection between a user running LoopTor and its entry guard. In the presented situation, the user has a real connection with another client. LoopTor uses the same TLS link to send both the traffic belonging to said connection and the loopback traffic. From the point of view of an adversary, observing this connection it is impossible to tell the difference between packets that belong to a real connection and the ones that do not.

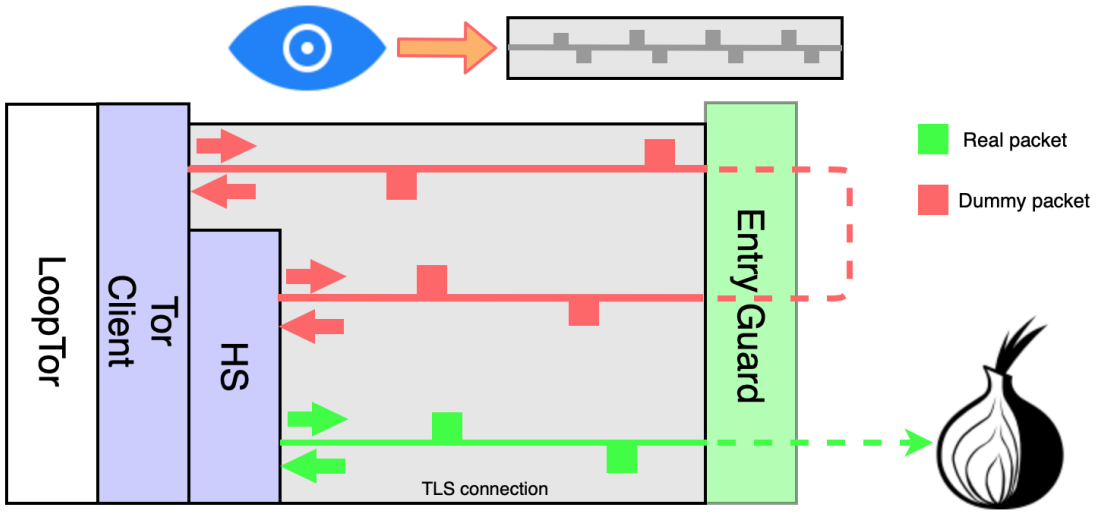

Figure 3.3: LoopTor traffic over TLS link

3.4 Implementation

In order to implement LoopTor and assess its performance, three different options are available: simulations, experiments with a test network and experiments on the live Tor network. This section presents the positive and negative aspects of each approach and justifies the choice of methods for the experiments that are carried out as described in the following chapter.

3.4.1 Experiments on the Tor network

Experiments can be done directly on the Tor network. All users can set up Tor nodes, integrate them into the live network and conduct their experiments in a real environment. The results from the experiments have a very high value because they

have been obtained in the same situation in which a real attack against a real user would be performed: in the live network with real traffic and real user activity. Nevertheless, the conditions in the Tor network are constantly changing, and this means that the same experiment can provide very different results when repeated.

Moreover, some experiments cannot be run on a live network for several reasons. For example, if the researchers have to modify a very big number of tor nodes, the infrastructure to set up all these nodes would be too expensive. In addition, experiments can have negative consequences on the network or its users. In some cases, these negative effects are hard to predict. This is especially critical in an anonymity system, like Tor, because as a result of some experiments, the identity of users could be unveiled.

3.4.2 Experiments with laboratory networks

An alternative solution to experiments with the live Tor network is a private laboratory network. Laboratory networks are low-scale replicas of the Tor network that are set up by the researchers to conduct their experiments. Because of the private nature of these networks, the experiments do not interfere with real users. Moreover, conditions in these simulated networks are typically stable, making the experiments more reliable and easy to replicate. Nevertheless, private Tor networks are much smaller than the real Tor network, and this means that it is almost impossible to properly replicate user activity or a realistic model of traffic. As a result, the conclusions from experiments conducted in these networks cannot be directly extrapolated to the real Tor network. Finally, creating private Tor networks can also be very expensive and require a lot of resources, such as computers.

3.4.3 Simulation

The third option is a simulation. A Tor network can easily be simulated on a computer, and this avoids requiring the deployment of a real network. Simulations are very reliable, meaning that they will typically produce very similar results when repeated with the same parameters. As a result, they are the best tool to identify cause-and-effect relationships, since they allow to repeat an experiment with the exact same conditions, only introducing a single modification, and observe how it affects the result. Moreover, simulations typically require less hardware than running a whole network, and this makes them cheaper. Simulations can also be automatized and scheduled, making them more time-efficient. On the downside, simulations have the same inconvenience as private research networks, which is that experiments are done in an ”artificial” environment and, as a result, the conclusions may not apply to the real Tor network.

3.4.4 Summary

After reviewing the three possible approaches, it becomes clear that each has its own advantages and all of them can be useful for this thesis. Nevertheless, given the special circumstances that have surrounded the development of this project because of the Covid-19 outbreak, described in Appendix A, the only viable possibility was to implement LoopTor on a simulated Tor network and use it to conduct all the experiments.

This has significant advantages, because it allows the conduction of said experiments with limited computational resources and ensures a stable environment. Nevertheless, the effectiveness of end-to-end correlation attacks against the current size Tor network cannot be properly validated, and the obtained results cannot be directly exported to a real environment.

LoopTor and all the scripts for each experiment have been implemented in Python 3. The code that has been used for this thesis can be found in the following GitHub repositories:

- https://github.com/spring-epfl/spring20-JaumePlanas
- https://github.com/spring-epfl/looptor/tree/jaume

Chapter 4

Results

Until you spread your wings, you'll have no idea how far you can fly.
– Napoleon Bonaparte

This chapter presents the results obtained from the experiments performed following the guidelines proposed in Chapter 3.

4.1 Experimental setup

- Experiment 1: Compare the performance of TC at different rates to determine at which range it operates properly. The goal is to reduce throughput without introducing too-high overhead. This experiment is divided into two parts:
  - Set different rate limits and compare them with the observed rate in order to determine the accuracy of TC at each rate.
  - Measure the overhead introduced by TC comparing different limits for both the rate and the burst.
- Experiment 2: Implement LoopTor on a simulated Tor network. Capture traffic both in situations in which there are real active connections and ones where there are not. Compare the observed traffic using both basic statistic metrics and machine learning classification to determine the success of LoopTor in making them indistinguishable.
- Experiment 3: Modify LoopTor so that the rate limit is established not only for the incoming traffic but also for the outgoing. In that situation, repeat the previous experiment and compare the results.

4.2 Metrics

For the first experiment, the following metrics will be used:

- **Throughput:** The rate at which data is successfully delivered over a communication channel, as defined in Equation 4.1. Equation 4.2 indicates the ratio between the throughput and the limit set with TC.

$$\text{Throughput} = \frac{\text{Received data (B)}}{\text{Time captured}} \tag{4.1}$$

$$\text{Ratio THR vs TC} = \frac{\text{Throughput}}{\text{TC Rate limit}} \tag{4.2}$$

- **Overhead:** Proportion of traffic that is sent in addition to user data. It is calculated according to Equation 4.3.

$$\text{Overhead} = 1 - \frac{\text{User data (B)}}{\text{Total data (B)}} \tag{4.3}$$

In the second and third experiments, the used metrics will be the ones that evaluate the performance of machine learning classifiers, described in Section 2.2.2. The accuracy (Equation 2.7), the precision (Equation 2.9), the sensitivity (Equation 2.10) and the F1-score (Equation 2.12) will be of relevance in this chapter.

4.3 Results

4.3.1 Determining rate of operation

TC takes as configuration a limit for rate the and a maximum size for the burst. In this experiment, TC's behavior is evaluated in order to determine at which range the system can operate better. To do that, a connection is established between a client and an echo server running on separate VMs inside the same private network. The client reads a .txt file and sends it to the server, which echoes it back to the client. TC reduces the rate at which the client receives data dropping TCP packets so that they are retransmitted until the desired throughput is achieved.

In the first part of the experiment, all incoming traffic is captured with *tcpdump* and analyzed with Wireshark to determine the real Throughput. The experiment has been conducted 10 times for each rate, ranging from 1kbps to 1Mbps. In comparison, without setting a limit with TC, the measured throughput in this scenario is around 27Mbps. In this experiment, the burst limit (B) is set at 20kB, but for the 1Mbps limit, another experiment is done setting the limit at 100kB to ensure that the desired rate can be reached. The results are presented in Table 4.1.

Table 4.1: Rate measurements (Average ± std. error of the mean)

TC Rate (kbps)	**THR(kbps)**	**Ratio THR vs TC**
1	1.23±0.22	1.23±2.15e−1
2	2.81±0.18	1.41±9.21e−2
4	4.99±0.29	1.25±7.49e−2
8	9.25±0.38	1.16±4.77e−2
16	17.94±0.39	1.12±2.46e−2
32	34.42±0.45	1.08±1.42e−2
64	65.58±0.31	1.02±4.84e−3
128	131.47±0.38	1.03±2.97e−3
256	261.15±0.33	1.02±1.29e−3
512	521.03±0.28	1.02±5.43e−4
1024 (B=20kB)	797.222±0.34	0.78±3.33e−4
1024 (B=100kB)	1039.17±0.29	1.02±2.83e−4

The results confirm that the accuracy of TC in limiting the rate is very high in the range between 62 and 512kbps. This is validated by the fact that the ratio is close to 1 at each measurement. Nevertheless, the measured throughput differs more from the limit when the rates become lower. To obtain rates higher than 1Mbps, the 20kB limit for the burst is not enough and produces a bottleneck effect.

The second part consists in running the experiment with different rate limits (1Mbps, 128kbps and 16kbps) and burst limits (2kB, 8kB, 16kB, 32kB, 128kB, 1MB) a total of ten times each combination. Extracting the measurements of the total overhead from all the runs of the experiment, the mean and the standard error of the mean are calculated. The obtained results, represented in Figure 4.1, indicate that running the experiment with a rate of 128kbps or 1Mbps provides a relatively low overhead as long as the burst limit is high enough.

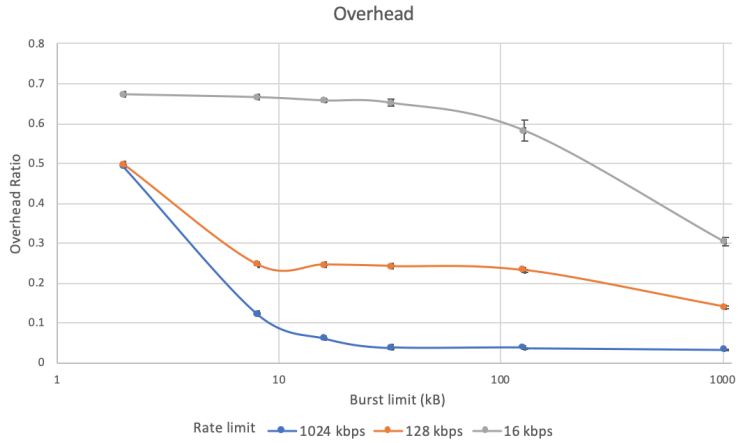

Figure 4.1: Overhead introduced by TC
(Average, with error bars indicating std. error of the mean)

All in all, this experiment shows that a rate limit in the range between 32 and 128 kbps is the best suited to reduce the throughput while keeping a reasonable overhead and without noise being introduced by TC.

4.3.2 LoopTor performance assessment

In order to evaluate the capacity of LoopTor to hide when real data is sent to an onion service, and thus provide sender unobservability, it has been tested on a simulated Tor network with users running LoopTor. In this scenario, traffic is captured both in situations in which the user has real active connections and others in which only loopback traffic is sent. For either situation, 100 captures of 100 seconds each have been made, setting a rate limit of 80kbps. In the captures with real connections, the data rate is 1 message per second (10.28% of all traffic sent). The two different situations are compared following two approaches: A simple comparison based on statistic metrics and a more powerful machine learning classifier.

Comparison of statistic metrics

In this situation, the comparison is done by analyzing statistic metrics. The examined parameters are the mean number of packages per capture, the standard error

and the probability distribution for the inter-arrival time. The results (Table 4.2) indicate that a higher number of packages and data packages are captured in the situations in which no real connection is established while the amount of Bytes is higher in the cases with real connections. This difference could be used by an adversary to determine whether real data is being sent/received, thus breaking unobservability.

Table 4.2: Captured packages using LoopTor (Average ± std. error of the mean)

	With connections	**Without connections**
Sent pkgs.	1100.05±2.71	1162.09±2.86
Sent kilobytes	689.23±0.97	686.22±0.85
Recv. pkgs.	1660.91±3.17	1726.59±3.35
Recv. kilobytes	1613.61±1.77	1608.95±1.89
Sent data pkgs.	773.10±2.12	848.97±2.43
Sent data kilobytes	663.14±0.93	661.27±0.84
Recv. data pkgs.	1081.53±1.31	1083.66±1.28
Recv. data kilobytes	1574.77±1.71	1565.55±1.85

The plots in Figure 4.2 represent the inter-arrival time distributions for the sent/received packages. The plots show minor differences in the distributions depending on the presence or absence of real connections.

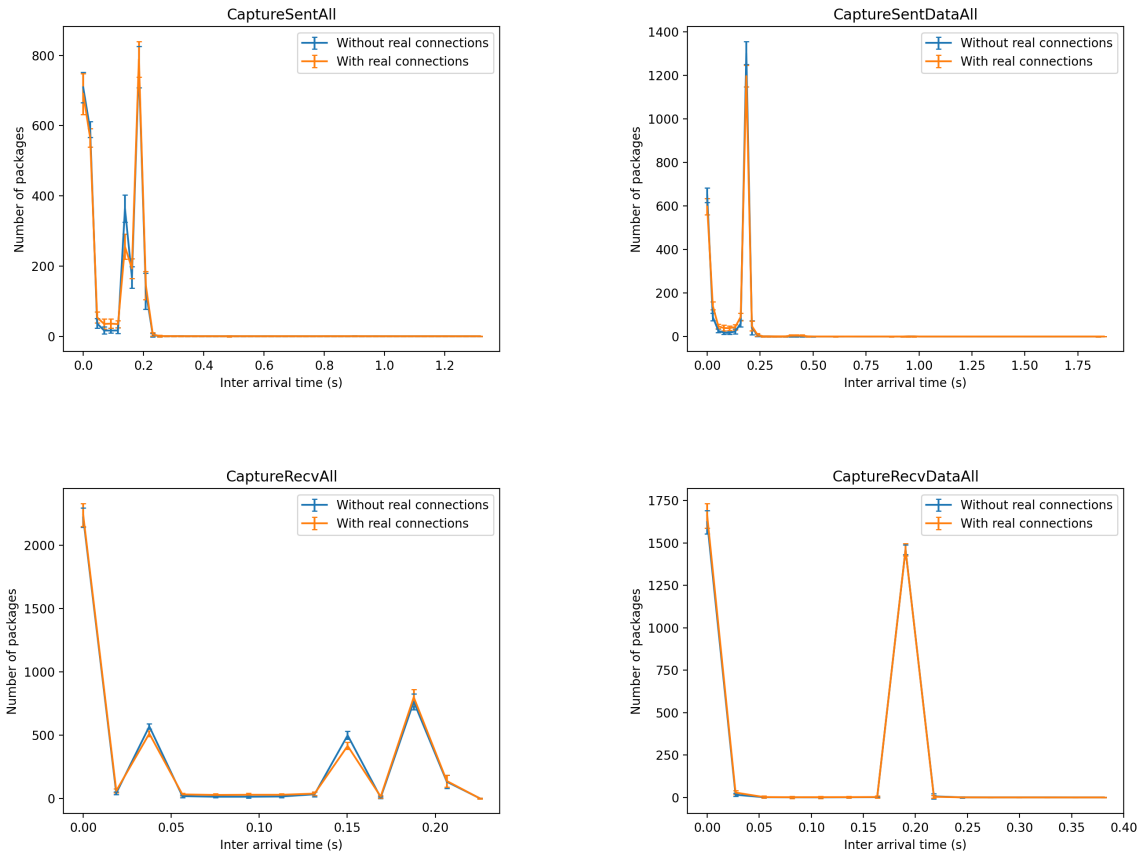

Figure 4.2: Inter-arrival time distributions

Machine learning classification

To test LoopTor against a more powerful tool that could be used to independently determine when the user is communicating through a real connection, a machine learning classifier is used. In concrete, a *Random Forest classifier* fetched with data extracted from captured traces. In this experiment, two different versions of this classifier are used: one trained only with statistical metrics from the captures (Mean, median, standard deviation, length, sum...), *TSFreshBasicExtractor*, and a more sophisticated one, *NgramsExtractor* that combines said features with the use of a *tf-idf vectorizer* that extracts packet counts and bursts. The purpose of both classifiers is to determine which of a given set of captures corresponded to real communication and which do not.

The metrics, calculated as the average of the results obtained performing 10-fold cross-classification, are presented in Table 4.3. The results indicate that the Ngrams classifier is always capable of segregating captures with real data from the ones where only loop traffic is included. The performance of the TSFreshBasic is lower but still very high.

Table 4.3: Classification metrics with LoopTor's basic implementation

Metric	**TSFreshBasic**	**Ngrams**
Accuracy	0.852857	1.0
Precision	0.857007	1.0
Sensitivity	0.852727	1.0
F1-Score	0.850984	1.0

4.3.3 Limiting outgoing rate

The last experiment tries to improve the capacity of covering real data. In this situation, the client runs a modified version of LoopTor in which TC is used not only to drop incoming TCP packets but also as an additional tool to limit the outgoing rate through an *egress* rule.

In order to determine which is the optimal limit that should be established for the sending rate (SR), different possibilities are evaluated; the same limit as the limit set for the receiving rate (RR), an increase of 20% and twice that value. Since the limit for the receiving rate is set at 80kbps, the limits for the sending rate are 80kbps, 96kbps and 160kbps. With the mentioned settings, the random forest classifier obtains the results shown at Table 4.4 using 10-fold cross-classification.

Table 4.4: Classification metrics with send rate limits

	SR=RR		**SR=1.2RR**		**SR=2RR**	
	TSFresh	Ngrams	TSFresh	Ngrams	TSFresh	Ngrams
Acc.	0.50000	0.95000	0.52500	0.95500	0.55000	0.95500
Prec.	0.50262	0.96130	0.52549	0.96182	0.55219	0.97284
Sens.	0.50000	0.95000	0.52500	0.95500	0.55000	0.96500
F1	0.49263	0.94871	0.52088	0.95455	0.54481	0.96405

Comparing these metrics with the ones at Table 4.3 shows that the use of TC to limit the outgoing rate succeeds in making it more difficult for the classifier to determine when there is a real connection. This is particularly relevant when only basic statistic metrics are taken into consideration by the classifier (TSFresh). When using the Ngrams classifier, the degree of success is still very high. This means that an adversary could still detect when the user is sending traffic. Regarding the different values tried for the rate limit, no significant differences are observed, being the success slightly lower when the limit for the send rate becomes closer to the limit for the receive rate.

Chapter 5

Discussion

For there is nothing either good or bad, but thinking makes it so.
– William Shakespeare, *Hamlet*

The results of the experiments present positive and negative findings regarding the achievement of the objectives set when designing LoopTor.

On the one hand, they prove that LoopTor has the potential to improve the security of a user against end-to-end traffic correlation attacks on the Tor network. This is achieved with LoopTor's final implementation, setting rate limits both for incoming and outgoing traffic, as showed by the difference between the metrics obtained in both situations (Tables 4.3 and 4.4). The last experiment demonstrates that machine learning classification based on simple statistical metrics only achieves an accuracy slightly higher than 50% in making a binary decision, the precision a random classifier would get.

On the other hand, LoopTor still allows machine learning classifiers trained with complex information about time and bursts patterns to determine with high precision (no lower than 95%) when there are active connections. This means that it is not yet capable of guaranteeing sender/receiver unobservability against powerful adversaries. In this situation, the improvement obtained by introducing TC for outgoing traffic is less significant. Besides, the different limits set for the send rate provide similar results, only with a slight improvement when setting a limit closer to the receive rate. This suggests that it is unlikely that this approach can provide significantly better results only by modifying the limits. In addition, the first experiment shows that the performance of TC decreases when getting outside of its optimal range, meaning that the margin for different values is not very wide.

Nevertheless, it is important to consider that all these experiments have been made in the best possible conditions for an attacker: a controlled environment where only traffic sent by the users is present, no computational restrictions and the capacity to capture all traffic in a link. As a result, they do not imply that an attacker would be able to do the same in a realistic situation on the Tor network. Moreover, although this thesis tried to achieve Sender-Receiver Unlinkability by providing both sender and receiver unobservability, only ensuring that an attacker cannot correlate the traffic sent by LoopTor observed simultaneously on two specific connections would be enough.

Further research is required to determine if the attacks conducted in this thesis would be as effective on the real Tor network. This will provide a more truthful indication of the effectiveness of LoopTor against its most powerful adversaries.

Chapter 6

Budget

I'm gonna make him an offer he can't refuse.

– Mario Puzo, *The Godfather*

In this chapter, the expenses incurred during the project are calculated. Since all used software is license-free and no raw materials have been required, the total cost of the project only includes computational costs and salaries of the researchers.

Regarding computational costs, it is impossible to determine the real economic impact that this project has had, since it has been developed using the resources provided by the SPRING lab of EPFL and the personal resources of the researchers without precise measurements being made in this regard. Nevertheless, Amazon Web Services (AWS) pricing calculator [58] provides an estimation of said costs. To perform this estimation, it is assumed that the proper set up would be a Linux machine with 4 CPUs, 16GB of RAM and 50 GB of storage. This is provided by a *t3a.xlarge* EC2 instance at a monthly cost of 129,83 USD (118.85€).

On the other hand, the project has been developed by the writer of this thesis, with a full-time dedication of 40 hours per week. Besides, he has had the support of SPRING lab co-worker Wouter Lueks, with a dedication of approximately 5 hours per week. Both researchers are considered Junior Engineers with a salary of 10€ per hour. Besides, professors Oscar Esparza and Carmela Troncoso have acted as supervisors of the project with a dedication of approximately 1 hour per week and are considered Senior Engineers with a salary of 30€ per hour. Since the project has been conducted between Switzerland and Spain, taxes and social security contributions are difficult to calculate. A fixed tax of 33.33% on all salaries is used as an estimation.

The total budget for the whole duration of the project (16 weeks) as defined in the work plan (Appendix A) is calculated in Table 6.1.

Table 6.1: Budget of the project

Concept	**Amount**	**Cost**	**Frequency**	**Total**
Full-time junior engineer	1	10.00€/h	40 hours/week	6,400€
Part-time junior engineer	1	10.00€/h	5 hours/week	800€
Part-time senior engineer	2	30.00€/h	1 hours/week	960€
Taxes and S.S. contr.	1	33.33%	170€/week	2,720€
Computational costs	1	118.85€/mth	4 months	475.40€
			Total Cost	11,355.40€

Chapter 7

Ethical and environmental considerations

> *Before you can change the world you must realize that you, yourself, are part of it.*
>
> – Gilbert Adair, *The Dreamers*

This chapter analyzes the social implications of this thesis. In the first section, the ethical considerations surrounding the concept of anonymity on the internet are discussed, as well as the ethical guidelines that the researcher has been entitled to follow for the duration of the project and in writing this dissertation. In the second part, the environmental impact of the project will be briefly analyzed.

7.1 Ethical considerations

This section evaluates the ethical implications of this project. It starts by tackling the ethical controversy that surrounds the concept of anonymity and continues by referring to the specific guidelines that have been followed in the research.

The main principles that guide research projects are to *do good* and *do no harm*. This thesis has been carried out with that in mind. It is well known that there are positive and negative implications of online anonymity. Behind the cover of anonymity is how attacks are made against companies or individuals using different kinds of malware. It is also the anonymity that allows access to parts of the Internet where illegal activities are conducted or hired. These behaviors have negative effects both socially and economically. Nevertheless, there are also situations in which to remain anonymous is a basic requirement, such as online voting or secure communications systems. The actual crisis of the Covid-19 provides a very significant example of how important it is to be cable of identifying people anonymously. There is a general consensus that the capacity to control all contacts that an infected person has had, using, for example, its mobile phone, can be a very helpful tool to stop the spread of the pandemic, but it cannot be done at the price of making public personal information about someone's health.

Since there already are a lot of techniques that make anonymity possible, as shown by the literature review in Chapter 2, the writer considers that the introduction of a new possibility does not additional harm, since whoever wants to use it for maleficent intentions, already has plenty of options. On the other hand, by creating

an accessible system with low computational requirements, the many advantages of anonymity are being made available to the general public. This is something that is expected to become fundamental in the near future.

Considering formal aspects, since this project has not required the participation of external individuals or organizations, the regulations for informed consent and privacy, confidentiality and anonymity of the participants are of no relevance. Moreover, no data from real users of the Tor network, or internet in general, has been collected. All experiments have been conducted on simulated environments. The author declares no conflicts of interest. The only external sources of funding to support this project have been the concession of a Movetia scholarship for a value of 2.200 CHF and a Mobint grant for a value of 1.200€. The concession of both scholarships is independent of the topic and scope of the research. The work presented is original and all information extracted from other author's work has been referenced. The results of the research are provided with honesty and transparency and all effort has been made to try to avoid any bias throughout all project.

This thesis will be made public in the format established by the university.

7.2 Environmental impact

This section includes an evaluation of the environmental impact of the project.

The researcher notes that, since environmental aspects were not determinant when designing the project and all the experiments and solutions that have been included in this thesis are software-based, there is no need for a formal Environmental Impact Assessment (EIA) as established by the European Commission [59] in *Environmental assessments of plans, programs and projects* [60]. Nevertheless, there are still some aspects that can be relevant and thus need proper analysis.

Since the project does not include the construction of any physical prototype and no raw materials are used, no direct environmental cost is identified regarding waste production, management and disposal. For the same reason, no pollutant emissions to air, water, or land are occasioned.

The only identified environmental cost is in terms of energy efficiency. Since LoopTor relies on constantly sending encrypted padding data to the network and the constant execution of several CPU requiring procedures such as TC, the consumption of energy increases significantly. Due to the scope limitations of this thesis, this reduction in efficiency has not been measured in a professional environment. Only an estimation of said impact has been done by the researcher, observing that by running LoopTor on a MacBook Pro 13 the duration of a fully charged battery is reduced by an average factor of 34% when compared to ordinary browsing.

This impact is insignificant if only the experiments conducted for this thesis are considered, but provided that the total number of Tor users as of 2020 is over 2 millions [61], further study should be made if the results of this research were to be implemented in a significant share of said users.

Independently of this project, the issue of energy efficiency of computational systems is a topic of growing relevance, and is now regarded as one of the main limitations to increase computing power. This issue is of extreme concern in supercomputers. Experts believe that without fresh approaches to system design, cooling and power delivery, it will soon be impossible to meet the demand for compute capacity without unaffordable operating costs and damaging environmental effects [62].

Chapter 8

Conclusions and future work

This is the way the world ends, not with a bang, but a whimper.

– T. S. Eliot

As this thesis comes to an end, it is time to remember the original purpose of the project. This research aimed to study end-to-end traffic correlation attacks and determine whether LoopTor was a useful defense against them for users of the Tor network.

The literature review and the experiments conducted have shown that traffic correlation attacks are a serious problem for most anonymity networks. In fact, it is assumed by most researchers that an attack will typically be successful if the attacker comes into the position to observe both endpoints of the communication. This assumption is also true for the Tor network.

Since Tor does not provide a solution for this vulnerability, some ideas have been proposed over time with mixed results. Despite over fifteen years of research on the Tor network, the problem of end-to-end traffic correlation attacks is not solved. For this reason, LoopTor was proposed as an innovative and simple defense against end-to-end traffic correlation attacks.

Based on the results from the experiments, it can be concluded that LoopTor was not capable of providing anonymity with its original design. Nevertheless, the introduction of modifications, specifically the restriction in the send rate, have provided optimistic results that present LoopTor as a useful tool for online anonymous communications in non-critical environments. Moreover, this thesis sets the baseline for future work that could be done in order to make LoopTor a completely secure communication system. The approaches that present more likelihood of success seem to be modifications in the rate regulation system. Precisely, the introduction of dynamic rate limits seems particularly promising based on the collected data and the findings gathered during this thesis.

Hopefully, this thesis will inspire researchers to tackle the issue of end-to-end traffic correlation attacks, improving the proposed defenses and ultimately finding a solution that manages to provide complete anonymity to Tor users. Although most attempts will probably only achieve partial success, as LoopTor does, it is important to remember that, in Marston Bates words:

"*Research is the process of going up alleys to see if they are blind.*"

Bibliography

[1] *Cambridge Dictionary*. Cambridge University Press, 2008, Anonymity.

[2] G. Le Bon, *The Crowd, A Study of the Popular Mind*. McMaster University Archive for the History of Economic Thought, 1896. [Online]. Available: https://EconPapers.repec.org/RePEc:hay:hetboo:lebon1896

[3] W. Lueks, A. Feal, and C. Troncoso, *LoopTor: resisting traffic-analysis without changing Tor*, SPRING Lab, EPFL.

[4] M. Backes, A. Kate, P. Manoharan, S. Meiser, and E. Mohammadi, "Anoa: A framework for analyzing anonymous communication protocols," in *2013 IEEE 26th Computer Security Foundations Symposium, New Orleans, LA, USA, June 26-28, 2013*. IEEE Computer Society, 2013, pp. 163–178. [Online]. Available: https://doi.org/10.1109/CSF.2013.18

[5] P. F. Syverson, D. M. Goldschlag, and M. G. Reed, "Anonymous connections and onion routing," in *1997 IEEE Symposium on Security and Privacy, May 4-7, 1997, Oakland, CA, USA*. IEEE Computer Society, 1997, pp. 44–54. [Online]. Available: https://doi.org/10.1109/SECPRI.1997.601314

[6] L. Barolli, M. Zhang, and X. A. Wang, Eds., *Advances in Internetworking, Data & Web Technologies, The 5th International Conference on Emerging Internetworking, Data & Web Technologies, EIDWT-2017, Wuhan, China, June 10-11, 2017*, ser. Lecture Notes on Data Engineering and Communications Technologies, vol. 6. Springer, 2018. [Online]. Available: https://doi.org/10.1007/978-3-319-59463-7

[7] E. Rescorla, "The transport layer security (TLS) protocol version 1.3," *RFC*, vol. 8446, pp. 1–160, 2018. [Online]. Available: https://doi.org/10.17487/RFC8446

[8] R. Dingledine and N. Mathewson, "Tor protocol specification," 2008. [Online]. Available: https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt

[9] Onion diagram. [Online]. Available: https://en.wikipedia.org/

[10] The Tor Project. [Online]. Available: https://www.torproject.org

[11] R. Dingledine, N. Mathewson, and P. F. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, M. Blaze, Ed. USENIX, 2004, pp. 303–320. [Online]. Available: http://www.usenix.org/publications/library/proceedings/sec04/tech/dingledine.html

[12] "Advanced encryption standard (aes)," National Institute of Standards and Technology (NIST), Standard FIPS PUB 197, 2001.

[13] E. Rescorla, "Diffie-Hellman Key Agreement Method," *RFC*, vol. 2631, pp. 1–13, 1999. [Online]. Available: https://doi.org/10.17487/RFC2631

[14] What is a ddos attack? [Online]. Available: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/

[15] Tor: Onion Service Protocol. [Online]. Available: https://2019.www.torproject.org/docs/onion-services

[16] X. Fu and Z. Ling, "One Cell is Enough to Break Tor's Anonymity," *Black Hat DC*, 2009. [Online]. Available: https://blog.torproject.org/one-cell-enough-break-tors-anonymity

[17] K. Muller, "Defending End-to-End Confirmation Attacks against the Tor Network," Master's thesis, Gjovik University College, 2015.

[18] R. Attarian, L. Abdi, and S. Hashemi, "Adawfpa: Adaptive online website fingerprinting attack for tor anonymous network: A stream-wise paradigm," *Comput. Commun.*, vol. 148, pp. 74–85, 2019. [Online]. Available: https://doi.org/10.1016/j.comcom.2019.09.008

[19] H. Jahani and S. Jalili, "Online tor privacy breach through website fingerprinting attack," *J. Network Syst. Manage.*, vol. 27, no. 2, pp. 289–326, 2019. [Online]. Available: https://doi.org/10.1007/s10922-018-9466-z

[20] A. Biryukov, I. Pustogarov, and R. Weinmann, "Trawling for tor hidden services: Detection, measurement, deanonymization," in *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*. IEEE Computer Society, 2013, pp. 80–94. [Online]. Available: https://doi.org/10.1109/SP.2013.15

[21] J. Díaz, "Tor, hidden services and de-anonymization," *INCIBE*, July 2015.

[22] L. Øverlier and P. F. Syverson, "Locating hidden servers," in *2006 IEEE Symposium on Security and Privacy (S&P 2006), 21-24 May 2006, Berkeley, California, USA*. IEEE Computer Society, 2006, pp. 100–114. [Online]. Available: https://doi.org/10.1109/SP.2006.24

[23] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. F. Syverson, "Users get routed: traffic correlation on tor by realistic adversaries," in *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, A. Sadeghi, V. D. Gligor, and M. Yung, Eds. ACM, 2013, pp. 337–348. [Online]. Available: https://doi.org/10.1145/2508859.2516651

[24] U. Muller, L. Cottrell, P. Palfrader, and L. Sassaman, "Mixmaster protocol - version 2." [Online]. Available: https://gnunet.org/mixmaster-spec.

[25] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a type III anonymous remailer protocol," in *2003 IEEE Symposium on Security and Privacy (S&P 2003), 11-14 May 2003, Berkeley, CA, USA*. IEEE Computer Society, 2003, pp. 2–15. [Online]. Available: https://doi.org/10.1109/SECPRI.2003.1199323

[26] A. M. Piotrowska, J. Hayes, T. Elahi, S. Meiser, and G. Danezis, "The loopix anonymity system," in *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*, E. Kirda and T. Ristenpart, Eds. USENIX Association, 2017, pp. 1199–1216. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/piotrowska

[27] M. J. Freedman, E. Sit, J. Cates, and R. T. Morris, "Introducing tarzan, a peer-to-peer anonymizing network layer," in *Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers*, ser. Lecture Notes in Computer Science, P. Druschel, M. F. Kaashoek, and A. I. T. Rowstron, Eds., vol. 2429. Springer, 2002, pp. 121–129. [Online]. Available: https://doi.org/10.1007/3-540-45748-8_12

[28] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Dissent in numbers: Making strong anonymity scale," in *10th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2012, Hollywood, CA, USA, October 8-10, 2012*, C. Thekkath and A. Vahdat, Eds. USENIX Association, 2012, pp. 179–182. [Online]. Available: https://www.usenix.org/conference/osdi12/technical-sessions/presentation/wolinsky

[29] H. Corrigan-Gibbs, D. Boneh, and D. Mazières, "Riposte: An anonymous messaging system handling millions of users," in *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*. IEEE Computer Society, 2015, pp. 321–338. [Online]. Available: https://doi.org/10.1109/SP.2015.27

[30] J. van den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich, "Vuvuzela: scalable private messaging resistant to traffic analysis," in *Proceedings of the 25th Symposium on Operating Systems Principles, SOSP 2015, Monterey, CA, USA, October 4-7, 2015*, E. L. Miller and S. Hand, Eds. ACM, 2015, pp. 137–152. [Online]. Available: https://doi.org/10.1145/2815400.2815417

[31] G. O'Gorman and S. Blott, "Improving stream correlation attacks on anonymous networks," in *Proceedings of the 2009 ACM Symposium on Applied Computing (SAC), Honolulu, Hawaii, USA, March 9-12, 2009*, S. Y. Shin and S. Ossowski, Eds. ACM, 2009, pp. 2024–2028. [Online]. Available: https://doi.org/10.1145/1529282.1529732

[32] H. Crombé and M. Declerq, "Correlation attacks on the tor network," Master's thesis, École polytechnique de Louvain (EPL), 2016.

[33] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "On flow correlation attacks and countermeasures in mix networks," in *Privacy Enhancing Technologies, 4th International Workshop, PET 2004, Toronto, Canada, May*

26-28, 2004, Revised Selected Papers, ser. Lecture Notes in Computer Science, D. M. M. Jr. and A. Serjantov, Eds., vol. 3424. Springer, 2004, pp. 207–225. [Online]. Available: https://doi.org/10.1007/11423409_13

[34] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of tor," in *2005 IEEE Symposium on Security and Privacy (S&P 2005), 8-11 May 2005, Oakland, CA, USA*. IEEE Computer Society, 2005, pp. 183–195. [Online]. Available: https://doi.org/10.1109/SP.2005.12

[35] M. D. Rechenthin, "Machine-learning classification techniques for the analysis and prediction of high-frequency stock direction," Ph.D. dissertation, University of Iowa, 2014.

[36] [Online]. Available: https://www.edureka.co/blog/classification-in-machine-learning/#classification

[37] [Online]. Available: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[38] L. Breiman and A. Cutler, "Random forests," Salford Systems, Tech. Rep., 2004.

[39] [Online]. Available: https://en.wikipedia.org/wiki/F1_score

[40] C. Parikh and H. T. Philbrook, *Biomarkers of Kidney Disease*. Academic press, 2017, ch. Chapter Two - Statistical Considerations in Analysis and Interpretation of Biomarker Studies.

[41] A. Serjantov and P. Sewell, "Passive attack analysis for connection-based anonymity systems," in *Computer Security - ESORICS 2003, 8th European Symposium on Research in Computer Security, Gjøvik, Norway, October 13-15, 2003, Proceedings*, ser. Lecture Notes in Computer Science, E. Snekkenes and D. Gollmann, Eds., vol. 2808. Springer, 2003, pp. 116–131. [Online]. Available: https://doi.org/10.1007/978-3-540-39650-5_7

[42] B. N. Levine, M. K. Reiter, C. Wang, and M. K. Wright, "Timing attacks in low-latency mix systems (extended abstract)," in *Financial Cryptography, 8th International Conference, FC 2004, Key West, FL, USA, February 9-12, 2004. Revised Papers*, ser. Lecture Notes in Computer Science, A. Juels, Ed., vol. 3110. Springer, 2004, pp. 251–265. [Online]. Available: https://doi.org/10.1007/978-3-540-27809-2_25

[43] V. Shmatikov and M. Wang, "Timing analysis in low-latency mix networks: Attacks and defenses," in *Computer Security - ESORICS 2006, 11th European Symposium on Research in Computer Security, Hamburg, Germany, September 18-20, 2006, Proceedings*, ser. Lecture Notes in Computer Science, D. Gollmann, J. Meier, and A. Sabelfeld, Eds., vol. 4189. Springer, 2006, pp. 18–33. [Online]. Available: https://doi.org/10.1007/11863908_2

[44] W. Wang, M. Motani, and V. Srinivasan, "Dependent link padding algorithms for low latency anonymity systems," in *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008,*

Alexandria, Virginia, USA, October 27-31, 2008, P. Ning, P. F. Syverson, and S. Jha, Eds. ACM, 2008, pp. 323–332. [Online]. Available: https://doi.org/10.1145/1455770.1455812

[45] G. Danezis, "Statistical disclosure attacks," in *Security and Privacy in the Age of Uncertainty, IFIP TC11 18th International Conference on Information Security (SEC2003), May 26-28, 2003, Athens, Greece*, ser. IFIP Conference Proceedings, D. Gritzalis, S. D. C. di Vimercati, P. Samarati, and S. K. Katsikas, Eds., vol. 250. Kluwer, 2003, pp. 421–426.

[46] N. Mathewson and R. Dingledine, "Practical traffic analysis: Extending and resisting statistical disclosure," in *Privacy Enhancing Technologies, 4th International Workshop, PET 2004, Toronto, Canada, May 26-28, 2004, Revised Selected Papers*, ser. Lecture Notes in Computer Science, D. M. M. Jr. and A. Serjantov, Eds., vol. 3424. Springer, 2004, pp. 17–34. [Online]. Available: https://doi.org/10.1007/11423409_2

[47] G. Danezis, "The traffic analysis of continuous-time mixes," in *Privacy Enhancing Technologies, 4th International Workshop, PET 2004, Toronto, Canada, May 26-28, 2004, Revised Selected Papers*, ser. Lecture Notes in Computer Science, D. M. M. Jr. and A. Serjantov, Eds., vol. 3424. Springer, 2004, pp. 35–50. [Online]. Available: https://doi.org/10.1007/11423409_3

[48] S. J. Murdoch and P. Zielinski, "Sampled traffic analysis by internet-exchange-level adversaries," in *Privacy Enhancing Technologies, 7th International Symposium, PET 2007 Ottawa, Canada, June 20-22, 2007, Revised Selected Papers*, ser. Lecture Notes in Computer Science, N. Borisov and P. Golle, Eds., vol. 4776. Springer, 2007, pp. 167–183. [Online]. Available: https://doi.org/10.1007/978-3-540-75551-7_11

[49] A. Houmansadr and N. Borisov, "The need for flow fingerprints to link correlated network flows," in *Privacy Enhancing Technologies - 13th International Symposium, PETS 2013, Bloomington, IN, USA, July 10-12, 2013. Proceedings*, ser. Lecture Notes in Computer Science, E. D. Cristofaro and M. K. Wright, Eds., vol. 7981. Springer, 2013, pp. 205–224. [Online]. Available: https://doi.org/10.1007/978-3-642-39077-7_11

[50] J. Feigenbaum, A. Johnson, and P. F. Syverson, "Preventing active timing attacks in low-latency anonymous communication," in *Privacy Enhancing Technologies, 10th International Symposium, PETS 2010, Berlin, Germany, July 21-23, 2010. Proceedings*, ser. Lecture Notes in Computer Science, M. J. Atallah and N. J. Hopper, Eds., vol. 6205. Springer, 2010, pp. 166–183. [Online]. Available: https://doi.org/10.1007/978-3-642-14527-8_10

[51] G. O'Gorman and S. Blott, "Large scale simulation of tor:," in *Advances in Computer Science - ASIAN 2007. Computer and Network Security, 12th Asian Computing Science Conference, Doha, Qatar, December 9-11, 2007, Proceedings*, ser. Lecture Notes in Computer Science, I. Cervesato, Ed., vol. 4846. Springer, 2007, pp. 48–54. [Online]. Available: https://doi.org/10.1007/978-3-540-76929-3_5

[52] Z. Ling, J. Luo, W. Yu, X. Fu, D. Xuan, and W. Jia, "A new cell counter based attack against tor," in *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*, E. Al-Shaer, S. Jha, and A. D. Keromytis, Eds. ACM, 2009, pp. 578–589. [Online]. Available: https://doi.org/10.1145/1653662.1653732

[53] Z. Ling, J. Luo, W. Yu, X. Fu, D. Xuan, and W. W. Jia, "A new cell counter based attack against tor," *Proceedings of the 16th ACM Conference on Computer and Communications Security*, 2009.

[54] A. Biryukov, I. Pustogarov, and R. Weinmann, "Trawling for tor hidden services: Detection, measurement, deanonymization," in *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*. IEEE Computer Society, 2013, pp. 80–94. [Online]. Available: https://doi.org/10.1109/SP.2013.15

[55] S. Chakravarty, "Traffic analysis attacks and defenses in low latency anonymous communication," Ph.D. dissertation, Graduate School of Arts and Sciences, Columbia University, 2014.

[56] D. Das, S. Meiser, E. Mohammadi, and A. Kate, "Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency - choose two," in *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. IEEE Computer Society, 2018, pp. 108–126. [Online]. Available: https://doi.org/10.1109/SP.2018.00011

[57] A. N. Kuznetsov, *TC*, 2001. [Online]. Available: http://man7.org/linux/man-pages/man8/tc.8.html

[58] Amazon web services pricing calculator. [Online]. Available: https://calculator.aws/#/createCalculator

[59] European commission. [Online]. Available: https://ec.europa.eu

[60] *Environmental assessments of plans, programmes and projects*, European Commission, 2017.

[61] Tor metrics. [Online]. Available: https://metrics.torproject.org/userstats-relay-country.html

[62] *Improving the energy efficiency of modern supercomputers*, Hewlett Packard Enterprise, 2017.

Appendix A

Work plan

This appendix presents the complete work plan that has been followed during the completion of this project. In the first section, there is a memory of all the incidences that have occurred and how they have affected the original work plan that was proposed in the *Project proposal* and the *Critical review*. This is followed by the actual work plan, consisting of the Work Breakdown Structure, the Work Packages, Tasks and Milestones and the Time Plan.

A.1 Incidences and deviations

Since this project was done during the Spring semester of 2019-2020, it was completely affected by the coronavirus (Covid-19) outbreak. This situation had significant effects on the project. First of all, courses and projects at EPFL were suspended, and all meetings with the supervisors have been done remotely Skype on a weekly basis to discuss the progress made.

Having to complete the project from home required to make some adjustments to the original conception of this project. The main issues that this situation brought basically affected the practical part of the project, since it was impossible to work with the other members of the lab (who had developed most of the code) and have access to more powerful computers that were required to run some of the scripts.

After discussion with the supervisors, it was decided that the best way to move forward with the project was to increase the importance of the theoretical part, since it is the one that could be more easily continued from home. This did not require significant modifications of the workplan because a part of the memory was always meant to be a summary of bibliographic research, but increasing the importance of this part reduced pressure from the practical one (development of scripts).

Nevertheless, thanks to the support of the supervisors, it has been possible to continue working on the code, and although some of the experiments that were planned have not been possible to carry out or had to be transformed into simulations, this project has produced relevant improvements, and most of the original goals have been achieved.

A.2 Work breakdown structure

The project is divided into six work packages.

On the one hand, four packages are related to the production of a technical solution. In that case, the work is divided into the typical steps that are done in such projects:

- Get basic knowledge
- Ideate and design solutions
- Implement the proposed solutions
- Benchmarking

These 4 work packages will be completed in order, because the previous are needed to move forward. When all the packages are completed, the technical part of the project will be ready.

In parallel, the project will also include a strong theoretical part, which will consist in summarizing information from several sources about relevant topics to this project. All this work is included in a single package that lasts the whole duration of the project.

Finally, documentation will be elaborated throughout the project, being the final report the most important element. It will include the results from both workflows and will be the document upon which the presentation will be done.

In figure A.1, a general overview of the process is depicted:

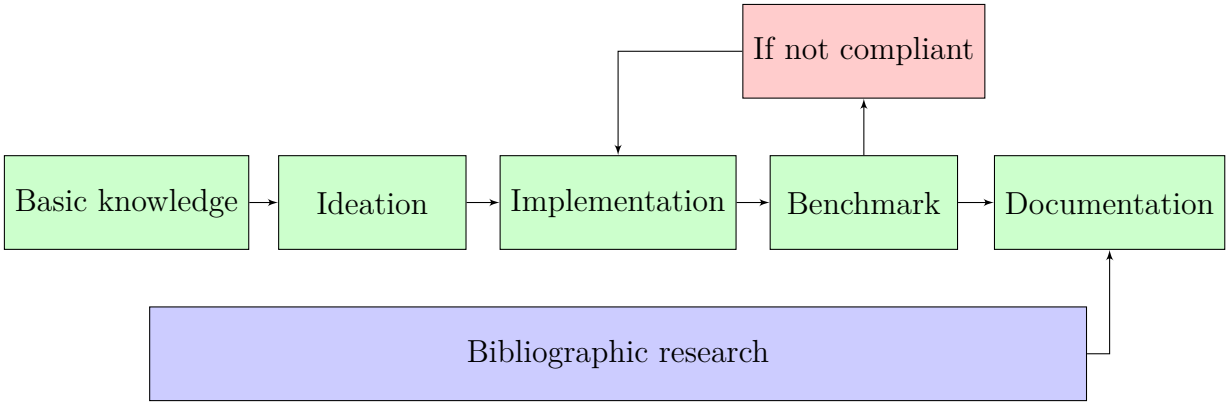

Figure A.1: Work breakdown structure schematic

A.3 Work packages, tasks and milestones

<table>
<tr><td colspan="2">Project: LoopTor</td><td>WP ref: 01</td></tr>
<tr><td colspan="2">Major constituent: Initial research</td><td>Sheet 1 of 3</td></tr>
<tr><td colspan="2">Short description: The purpose of this package is to gain the basic knowledge about Tor to start the project. This also includes learning about the current state of the art of the research conducted by the team. The supervisor also suggested making python scripts to improve understanding of TC.</td><td>Planned start date: 15/02/2020
Planned end date: 01/03/2020
Start event: 15/02/2020
End event: 05/03/2020</td></tr>
<tr><td>Internal tasks:
Read Bibliography.
Get familiar with previous work of the department.
Experiment with TC.</td><td>Deliverables: Report on the TC experiment results</td><td>Dates: None</td></tr>
</table>

Table A.1: Work package 1

<table>
<tr><td colspan="2">Project: LoopTor</td><td>WP ref: 02</td></tr>
<tr><td colspan="2">Major constituent: Ideation</td><td>Sheet 1 of 3</td></tr>
<tr><td colspan="2">Short description: After analyzing the situation, a proposal of how to improve the situation was be elaborated detailing the path to follow in order to make a solution that met all the criteria.</td><td>Planned start date: 01/03/2020
Planned end date: 31/03/2020
Start event: 06/03/2020
End event: 31/03/2020</td></tr>
<tr><td>Internal tasks:
Design a solution.
Present a proposal to the supervisor</td><td>Deliverables:
Formal proposal</td><td>Dates:
31/03/2020</td></tr>
</table>

Table A.2: Work package 2

<table>
<tr><td colspan="2">Project: LoopTor</td><td>WP ref: 03</td></tr>
<tr><td colspan="2">Major constituent: Implementation</td><td>Sheet 2 of 3</td></tr>
<tr><td colspan="2">Short description: Before starting with new code, the scripts required to run LoopTor were adapted into Python3, since most of them were written in the deprecated Python2. After that, the designed solutions were implemented.</td><td>Planned start date: 20/03/2020
Planned end date: 30/04/2020
Start event: 01/04/2020
End event: 30/04/2020</td></tr>
<tr><td>Internal tasks:
Adapt looptor into Python 3
Implement proposed solutions.</td><td>Deliverables:
Source code of the implementation.</td><td>Dates:
30/04/2020</td></tr>
</table>

Table A.3: Work package 3

<table>
<tr><td colspan="2">Project: LoopTor</td><td>WP ref: 04</td></tr>
<tr><td colspan="2">Major constituent: Benchmark</td><td>Sheet 2 of 3</td></tr>
<tr><td colspan="2">Short description: The implemented solutions were tested in order to assess whether they could fix the problem and how they measured in several relevant scores.</td><td>Planned start date: 01/05/2020
Planned end date: 15/05/2020
Start event: 01/05/2020
End event: 15/05/2020</td></tr>
<tr><td>Internal tasks:
Run benchmark.
Analyze results and extract conclusions.</td><td>Deliverables:
Benchmark results</td><td>Dates:
15/05/2020</td></tr>
</table>

Table A.4: Work package 4

<table>
<tr><td colspan="2">Project: LoopTor</td><td>WP ref: 05</td></tr>
<tr><td colspan="2">Major constituent: Bibliographic research</td><td>Sheet 2 of 3</td></tr>
<tr><td colspan="2">Short description: Throughout all the project bibliographic research from several sources of information was conducted to justify the purpose of this project, why it was relevant and how it improved the previous situation</td><td>Planned start date: 15/02/2020
Planned end date: 31/05/2020
Start event: 15/02/2020
End event: 15/05/2020</td></tr>
<tr><td>Internal tasks: None</td><td>Deliverables: Included in the final report.</td><td>Dates: None</td></tr>
</table>

Table A.5: Work package 5

<table>
<tr><td colspan="2">Project: LoopTor</td><td>WP ref: 06</td></tr>
<tr><td colspan="2">Major constituent: Documentation</td><td>Sheet 3 of 3</td></tr>
<tr><td colspan="2">Short description: Complete and submit all the required documents about the project in order to keep a record of all the work that has been done and set a baseline for future work.</td><td>Planned start date: 15/02/2020
Planned end date: 06/06/2020
Start event: 15/02/2020
End event: 06/06/2020</td></tr>
<tr><td>Internal tasks:
Project Proposal and Work Plan
Project Critical Review
Final Report
Continuous documentation of the progress</td><td>Deliverables:
Project Proposal and Work Plan
Project Critical Review
Final Report</td><td>Dates:
01/03/2020
14/04/2020
06/06/2020</td></tr>
</table>

Table A.6: Work package 6

A.4 Time plan (Gantt chart)

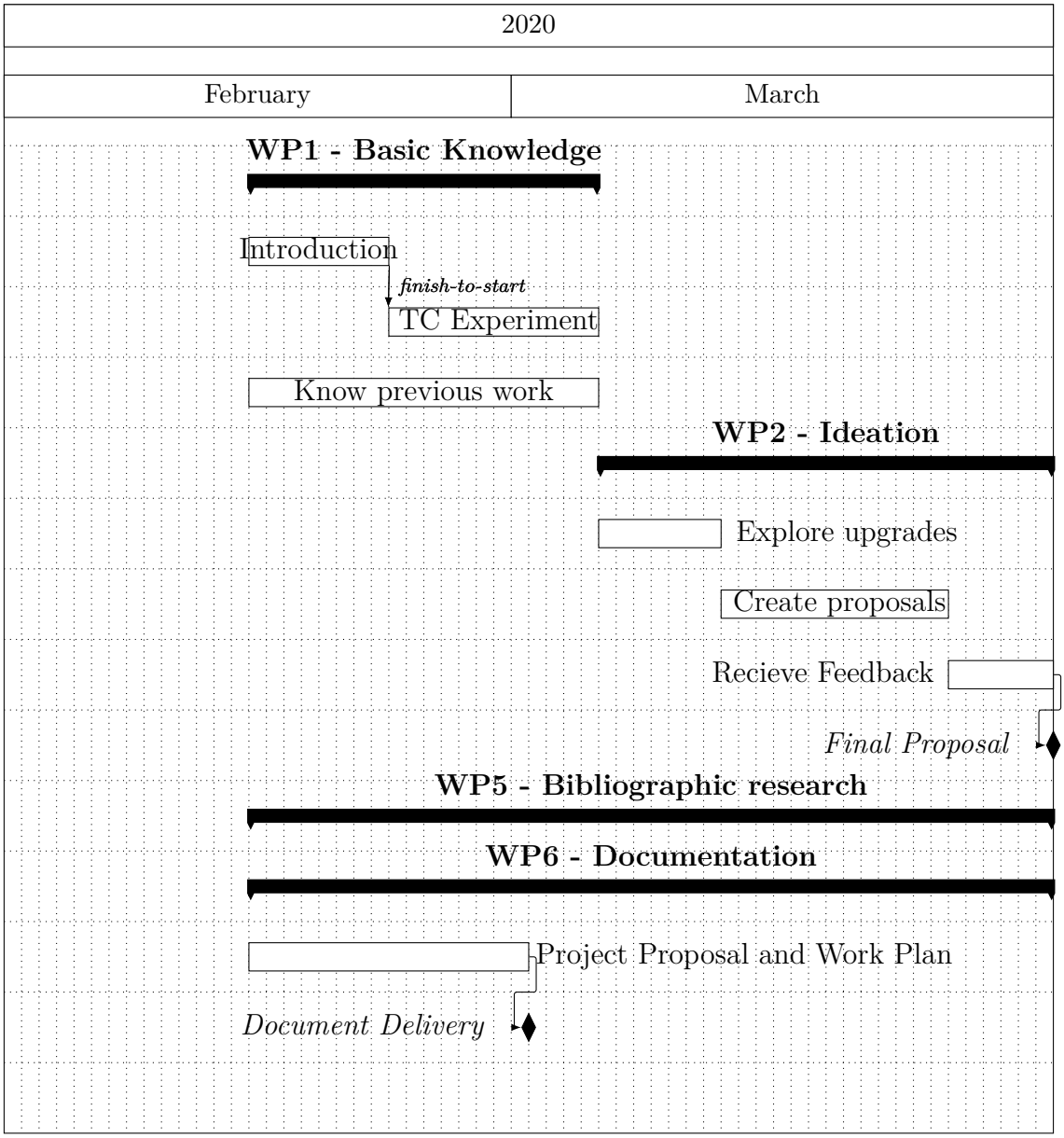

Figure A.2: Gantt chart p.1

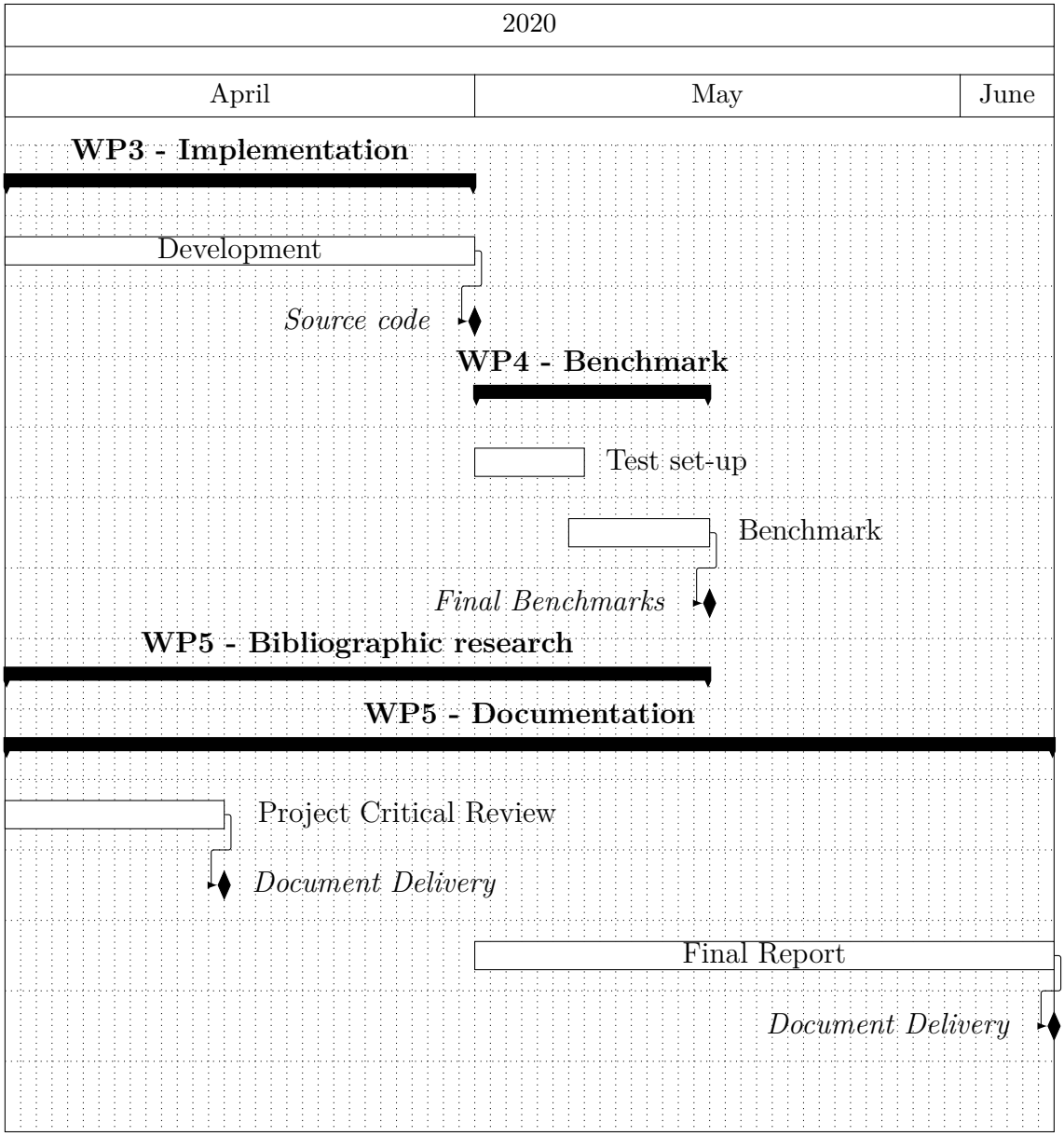

Figure A.3: Gantt chart p.2

Appendix B

Onion services

This appendix presents the procedure to access to a Tor onion service[15].

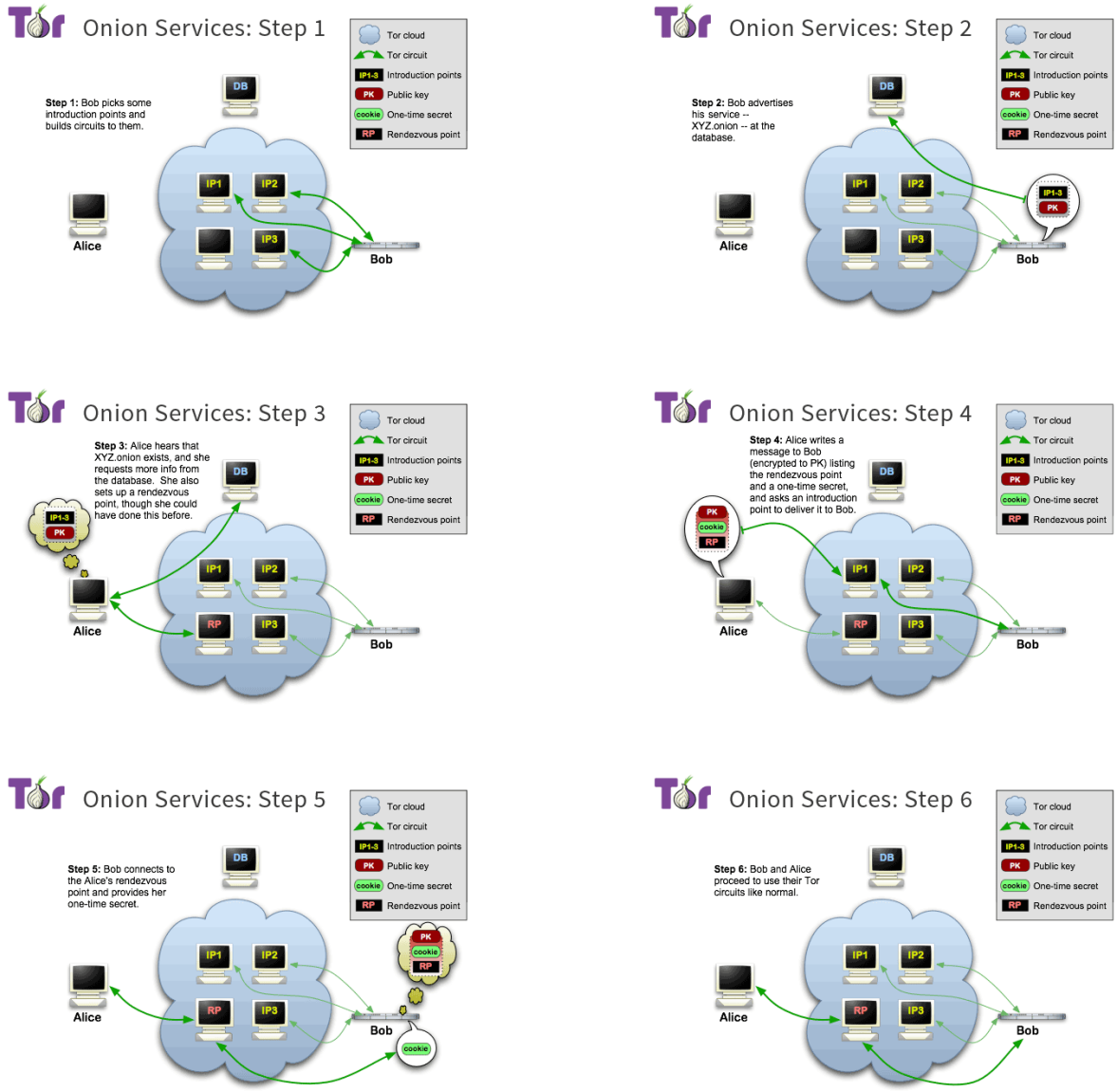

Figure B.1: Connection to an onion service