João André Furtado Teixeira

Aluno Nº 48047 (MIEI)

Strengthening of Tor Against Traffic Correlation with K-Anonymity Input Circuits

Dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Computer Science and Engineering

Adviser: Henrique João Lopes Domingos,
Professor Associado,
Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa

Examination Committee

Chair: Prof. Doutor António Maria Lobo César Alarcão Ravara, FCT-NOVA
Rapporteur: Prof. Doutor Paul Andrew Crocker

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

December, 2022

João André Furtado Teixeira

Aluno Nº 48047 (MIEI)

Strengthening of Tor Against Traffic Correlation with K-Anonymity Input Circuits

Dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Computer Science and Engineering

Adviser: Henrique João Lopes Domingos,
Professor Associado,
Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa

Examination Committee

Chair: Prof. Doutor António Maria Lobo César Alarcão Ravara, FCT-NOVA
Rapporteur: Prof. Doutor Paul Andrew Crocker

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

December, 2022

Strengthening of Tor Against Traffic Correlation with K-Anonymity Input Circuits

This document was created using the (pdf)LaTeX processor, based on the "novathesis" template[1], developed at the Dep. Informática of FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

To my parents and girlfriend.

Acknowledgements

I would like to express my gratitude to my adviser, Henrique Domingos, who guided me throughout the elaboration of this thesis. He always assisted me whenever I ran into a problem or a question about my research or writing, steered me in the right direction when he thought I needed it.

I would also like to thank my family for providing me with unfailing support and continuous encouragement in all my years of study and through the process of researching and writing this thesis. This accomplishment wouldn't have been possible without them. Thank you.

Abstract

Tor is a popular volunteer overlay network comprising thousands of relays in order to conceal a user's location, allowing them to surf the Internet anonymously. Despite its popularity, Tor, like other alternative anonymous systems, is vulnerable to traffic correlation attacks, which enables an attacker to match a source IP with a destination IP. This can be accomplished by a usually called "state-level adversary", capable of observing both endpoints of a Tor circuit and correlating the corresponding outgoing and incoming flows using traffic analysis techniques. Although correlation attacks are a known weakness in Tor's design, there aren't practical solutions to mitigate this attack without degrading the performance of the system. To tackle this problem, we present the Trusted Input Relay proposal (TIR) — a censorship-resistant Tor pluggable transport that leverages the notion of indistinguishable and tunnelled k-anonymity circuits. Used to decouple the input traffic from the origin user, segmenting it in multiple flows sent to multiple sources of Tor input nodes, by collaborative proxied connections. Thus, we aim at improving the existent Tor system to make it more robust against traffic correlation attacks launched by a global adversary acting as a censor, that observes all flows transiting to a circuit. In this way, we can mitigate the impact of correlation attacks between Tor input and output circuits, without significantly damaging the system and internetworking performance.

Keywords: Censorship, Tor network, traffic correlation, state-level adversary, k-anonymity, pluggable transport.

Resumo

Para lidar com essas limitações, diferente técnicas de contornar a censura foram sendo propostas pela comunidade de investigação. Algumas dessas propostas foram adotadas e implementadas pelos programadores como mecanismos de robustecimento da rede Tor a rede de anonimização mais popular na Internet, tendo em vista garantir maior anonimato e robustecer a evasão de censura e evitamento de bloqueios de tráfego. Algumas das técnicas propostas mais relevantes foram estudadas no contexto da dissertação. Estas variam desde o uso de *proxies* de anonimização do endereçamento IP, formas de ofuscação da origem e destino de tráfego com base em múltiplas rotas de encaminhamento baseado em redes sobrepostas ou rotas de engodo[1]. Também a utilização de métodos de ofuscação de tráfego com suporte de canais secretos encobertos suportados na transformação, camuflagem ou imitação de tráfego aparentemente regular, ou técnicas de transporte de tráfego proibido escondido como canais encobertos codificados e encapsulados em tráfego publico permitido. Estas soluções podem também envolver, por exemplo, soluções de processamento e reencaminhamento através de soluções em nuvem. Mais recentemente foram também propostas novas técnicas para suporte de canais encobertos baseados na transformação do tráfego escondido e a sua codificação camuflada em canais de transmissão de conteúdos multimédia, tráfego de jogos ou tráfego usual de ferramentas de videoconferência. Uma ideia fulcral das diferentes soluções e não ser possível, mesmo a autoridades em países de regimes totalitários, bloquear todo o tráfego ou impedir o uso das ferramentas, aplicações que envolvem esse tráfego, dados os prejuízos económicos ou danos políticos que dai podem advir.

Nesta tese propomos um mecanismo de fortalecimento da utilização da rede Tor para evasão de censura na Internet, de modo a defender os utilizadores de atividades de correlação de tráfego entre circuitos de entrada e de saída, por parte de oponentes do nível-estado. A solução pode ser vista como complementar em relação a muitas das técnicas acima indicadas e mitigará a possibilidade de análises de correlação de tráfego. Isto permite a um censor combinar pacotes IP de origem enviados por utilizadores para nós de entrada vigiados na rede Tor e pacotes analisados nos circuitos de saída da rede Tor e que se destinam a sistemas finais.

[1] sendo estas técnicas habitualmente associadas na literatura as noções de *network overlaying*, *tunneling*, *decoy routing ou onion-routing*.

As atividades de adversários habitualmente designados por oponentes de nível de estado podem envolver colaborações entre diferentes entidades, podendo envolver autoridades, empresas ou provedores de acesso Internet envolvidos em iniciativas de reconhecimento, ou identificação tráfego enviado para vários nós da rede Tor distribuídos na Internet. As técnicas mais eficazes de correlação para deteção e censura baseiam-se em métodos e algoritmos de correlação com aprendizagem automática que são suficientemente poderosos para detetar correlações que podem ser observadas em tempo real, por análise de diferentes características de tráfego mesmo no caso de pacotes com cargas cifradas. Normalmente essas atividades de análise e de censura tem lugar em coordenação das diferentes entidades atuando com particular ênfase em certas regiões de encaminhamento na Internet.

A solução estudada e proposta na dissertação e focada numa abordagem de utilização robustecida da rede Tor baseada na adoção de rotas multi-caminho e com k-anonimação suportadas em túneis TLS, formando estas rotas redes de pré-anonimização comunitárias e auto-organizadas. Tendo em vista o pré-processamento e pré-encaminhamento de pacotes que são depois enviados pelos diferentes nós para diferentes circuitos de entrada da rede Tor. A ideia e que o tráfego original enviado por um utilizador seja desagregado pelas múltiplas rotas e pelos nós operando em diferentes regiões de encaminhamento. Pressupõe-se que esses nós não são conhecidos publicamente, apenas do conhecimento das comunidades auto-organizadas e que os providenciam colaborativamente entre si. Assim, e possível disponibilizar esses nós em computadores voluntários de diferentes utilizadores em distintas regiões de encaminhamento na Internet, podendo também ser disponibilizados como nós operando em contentores virtualizados em vários nós de computação operando em várias nuvens de diversos provedores, espalhados em centros de dados pelo planeta.

A solução concebida foi implementada, sendo analisada a sua viabilidade e a correção do seu funcionamento. A implementação foi objeto de análises experimentais de modo a compreender-se o impacto da solução no que diz respeito a análises de condições de latência, capacidade de débito e não-observabilidade do tráfego, comparativamente a utilização da rede Tor, tal como a mesma e hoje utilizada e sem utilização da proteção proposta.

Palavras-chave: Censura na Internet, rede Tor, correlação de tráfego, circuitos de entrada e saída, k-anonimação, rotas multi-caminho.

Contents

List of Figures

List of Tables

Glossary

censorship	The control or suppression of what can be accessed, published, or viewed on the Internet enacted by regulators, or on their own initiative.
censorship circumvention	The use of various methods and tools to bypass internet censorship.
covert channels	Type of attack that creates a capability to transfer information objects between processes that are not supposed to be allowed to communicate by the computer security policy.
k-anonymity	Provides a measure of privacy protection by preventing re-identification of data to fewer than a group of k data items, in our case, K senders and the data is the IP source.
media-morphing	Protocols which enables the creation of morphed media data by modulating data into the input of popular multimedia applications (e.g. Skype, Zoom, Youtube).
pluggable transport	Pluggable transports (PT) transform the Tor traffic flow between the client and the bridge. This way, censors who monitor traffic between the client and the bridge will see innocent-looking transformed traffic instead of the actual tor traffic.
state-level adversary	An authority or organizational entity who controls the freedom of a system.
Tor bridges	A server in the Tor network whose existence is non-public and which can therefore provide access for blocked clients, often in combination with pluggable transports, which registers itself with the bridge authority.

traffic correlation	Traffic correlation analyzes the traffic on a set of links (observation points) inside the network and estimates the likelihood for each link to be on the path of the traffic under consideration.
traffic morphing	A special effect in traffic that changes (or morphs) the used protocol into another through a seamless transition.
tunnelling	A communications protocol that allows for the movement of data from one network to another. It involves allowing private network communications to be sent across a public network (such as the Internet) through a process called encapsulation.
unobservability	Unobservability ensures that a user may use a resource or service without others, especially third parties, being able to observe that the resource or service is being used, technically, it is the measure of how well a system can be inferred from knowledge of its external outputs, being avoided by internal states authorities.

Chapter 1

Introduction

The Internet has enabled excellent access to information and communication resources. Over the years, defenders of digital rights and activists for the use of the Internet as a free, neutral, and social good for a better world have raised growing concerns over how legal and regulatory trends might constrain online freedom of expression. Current Internet technology allows different players (including ISPs, content-distribution companies, state authorities and other players) to closely monitor the communications of users which may allow those entities to uncover data and private interests, such as political opinions, consumer preferences, health data, professional data or other personal information. Anonymity internetworking and privacy-enhanced communication environments have been designed to defeat network surveillance and traffic analysis activities. The goal is to allow for countermeasures to preserve anonymity and communication privacy for Internet users.

Internet censorship activities. Censorship activities against users, conducted by distinct entities types involved and working many times in collaborative actions, are particularly targeted to prevent access to information. Even though they lack physical or legal control over the resources or websites for which the traffic is also blocked. These entities use advanced technical censorship methods, such as: IP/Site blocking; content filtering (including text, images, or videos); identification of communication endpoints and their locations; use of cross-layering stateful traffic inspection tools for real-time analysis; and traffic deanonymization powerful machine learning methods. The censorship activities involving these techniques range from "fine-grain" inspection and specific blocking actions to block entire communication protocols. National and sometimes transnational control can be established through the combined use of the above censorship techniques. It is well known, and it was extensively published, that many countries and their authorities implement today tight censorship policies based in these mechanisms, deterring their

citizens from accessing free information with privacy preservation — a reality in political oppressive regimes. The value of online interactions for political rights and social reform is growing, as we notice in many situations: control of Hong Kong protests [13] and many situations of censorship techniques to enforce control over available online content, such as in China [20, 45], Russia [75], India [50] and Iran [10] are some of those regions.

Internet censorship circumvention. Towards the goal of bypassing censorship practices and mechanisms, a significant amount of research work proposed by the research community, targeted ways to implement Internet censorship circumvention solutions as countermeasures to protect Internet users. The development of available censorship circumvention tools from the recent research community has raised the challenge for censors, forcing them to adapt and to research more sophisticated countermeasures for their censorship intentions.

The Tor Network. Tor is today the most popular volunteer and global overlay network in the Internet environment, comprising thousands of relays using onion-routing techniques in order to conceal information such as network locations or used systems, allowing users to surf the Internet anonymously. Today, Tor allows the construction of special services to remain anonymous on the network, making client and targeted server interoperation anonymous. Unfortunately, Tor has been used also to support illegal activities, which raises the need for tools that can help in de-anonymizing illegal Tor flows hiding criminal activities. It is therefore not surprising that recent research work directions around Tor studies has been conducted with two different and antagonistic goals: solutions to improve anonymity conditions and proposals on techniques that allow to de-anonymize illegal flows between clients and servers. This way, Tor was extensively used as a "Lab" and testbench environment to inspire the development of different ideas. The goal and contribution of the present dissertation is in particular aligned with the first perspective above, with a close motivation on the study of possibilities for strengthening the Tor use with more robust techniques for privacy and anonymity preservation.

The "Arms-Race" around Tor. It is interesting to observe that the research on "censorship circumvention versus censorship effectiveness" around the Tor research communities is like an "arms race" in the research arena, motivating agendas for different research communities with possible antagonistic objectives: censorship avoidance and traffic deanonymization versus privacy preservation and endpoint's anonymization.

State-level and Omnipresent Censorship Adversaries. On the other hand, the strong commitment of many governments and political regimes in censorship activities, is today organized around censorship activities conducted by "state-level adversaries" or "state-sponsored adversaries" — definitions related to characterize the collaborative effort of different entities, such as state-authorities, ISPs and cybersecurity companies. These entities are capable of observing both endpoints and Tor established circuits to correlate corresponding outgoing and incoming flows, with advanced traffic analysis techniques including efficient machine learning solutions. In other situations, in the case of more oppressive regimes or in scenarios of cyberwar incidents and political crisis, a censor

can take the full control of Internet isolation. By using exceptional cybersurveillance programs and measures, a censor can limit or fully block freedom or privacy rights, and citizenship guarantees, acting as an omnipresent censorship adversaries, conducting their actions under special mandates based on legal acts. An example is the case of the Patriot Act, enacted in the sequence of the 11/September/2011 terroristic attacks in the USA [70]. Activities conducted by omnipresent adversaries in the context of cyberwarfare actions are other examples we saw in the past [19]. Omnipresent adversaries imposing strong or complete isolation on the Internet use are also the case of countries and oppressive regimes, where only specific authorized and privileged civilians or state entities can have free access to the internet, turning total censorship circumvention a difficult achievement.

Opportunities for censorship circumvention. Considering the more specific case of state-level adversaries and their current common practices in the Internet environment, the goal in the research on censorship circumvention has been in particular directed to exploit an interesting fundamental limitation: the practical impossibility of total blockage of Internet traffic due to political, social or economic damages. This means that the adversary can control and limit, more or less, the income and outcome traffic on a certain AS region, having means to conduct different censorship actions, as at first described. However, some Internet traffic classes and specific protocols exist in practice quite impossible to be blocked. Otherwise, this will cause widespread discontent among citizens, bringing social, political and commercial consequences that can be difficult to permanent counteract. This happens with traffic such as: HTTPS, TLS, SMTP, WebRTC/TLS channels, In/Out traffic involving Cloud-Services and E-Commerce Sites, or interchanged traffic of other popular media streaming application protocols (also including media traffic for gaming and Apps used in mobile devices). This reality opens the door to research forms of overlayed traffic, traffic tunnelling and proxy-based onion routing nodes. Operated by collaborative partners, spread among different AS (or Autonomous System) routing regions, in a global environment without complete control by the adversary — that only has authority on specific AS-regions under their control. In this dissertation, we studied solutions used to fight against those state-level censors, unless an omnipresent censor has total control over all internet-connections. We present different solutions in chapter 2 — Related Work. For the remaining or this introduction, we focus on the context for our work, the problem statement for the dissertation, the definition of our objectives and a summary of the main contributions.

1.1 Context and motivation

Tor [25] is a well-known and popular low-throughput network to support its users to access the Internet under anonymous suppositions. The main objective in Tor is to give an easy-to-use wide area network that allows for non-disclosure of its IP address in traffic sent by a source IP to a destination IP. At whatever point a user needs to connect for instance to a destination web server, Tor begins by building an ordinary encrypted circuit

based on three intermediary or so-called relay nodes: entry, middle, and exit relays. Communication is then sent alongside those relays which implement a variant of an onion routing protocol solution [74], using nested levels of encryption to tunnel the packets such that no relay can know both the source IP nor the destination IP. Tor can be a double-edged knife, used for the good and the evil, however, it is considered as a user-friendly tool for privacy and security communication, and it was used by numerous informants and journalists for ensuring their protection on the web, for directing journalism investigation in hostile environments [34]. It has likewise been in general used as an Internet censorship circumvention tool, permitting individuals to get to blocked sites, as a tool for censorship circumvention [14, 29, 60].

Despite its popularity and available end-user tool, (such as the Tor browser and different tools than improved overtime extra protections to the Tor ecosystem), like different solutions that can be described as low-throughput anonymity-preserving internetworking systems, there are proof that Tor can be helpless against traffic correlation attacks (or observing activities from censorship jurisdiction and existent traffic-analysis solutions)[57, 64]. Traffic correlations of Tor circuits empower the observation of source IP in inbound circuits with a destination IP in outbound circuits. Such attacks can be completed when an adversary (or censor) can observe the traffic at the circuit's entry and exit relay. By intercepting the traffic at these particular relay nodes, the adversary can use various features, for example, the volume of traffic, protocol typology, or inter-packet arrival times, to perform different results of correlation analysis. Two flows may be correlated to adequacy results, at that point, the adversary can decide with a severe potential extent of conviction the IP locations of the observed circuits and endpoints.

In the worldwide discussion of Internet censorship movement and particularly, in the research contributions from the scientific community, traffic correlation is accomplished today as activities from defined state-level adversaries [3, 64]. Those can join the intervention of different players involved, from state authorities to companies and internet service providers, making anonymity vulnerabilities more exploitable, by advanced machine-learning techniques applied to correspond outgoing and incoming flows [85, 100]. Although correlation attacks defence are a well-known weakness in Tor's design, and even considering that, differential privacy-preservation techniques can be used in complementary ways [4, 73] to enforce anonymity capabilities. However, there are still no practical solutions that can mitigate this attack without having a significant impact to the overall systems' performance, at least compared with the use of Tor as "it is". Using such combinations is yet an open research field [46, 53, 64, 73].

1.2 Problem statement

Traffic correlation attacks are a real danger for Tor users and their anonymity preservation. State-level adversaries can deploy at large-scale traffic correlation activities and related advanced machine learning techniques. A fundamental limitation of censorship

circumvention systems was previous proposed is that they rely on proxies operated by third parties. This hampers the scalability of the system, since the user must know and trust the proxy. By using a fixed input Tor proxy or bridge in permanent Tor established circuits (e.g., using onion-routing) crossing the Tor nodes from the same input node. It opens the possibility for the adversary to correlate input and output Tor traffic. This can happen even in the input application traffic is supported by encrypted protocols, because adversaries will conduct correlations using metadata and traffic characteristics and features such as: similarities in traffic shaping, packet lengths in a series of packets or temporal similarities inferred from inter-entering and inter-leaving packet distributions. This dissertation aims to be a defence mechanism to mitigate this problem.

1.3 Goals

In this dissertation, we focus on improving the current Tor system to make it progressively robust against correlation attacks by worldwide passive adversaries that examine flows on Tor established circuits. Our hypothesis is to leverage the notion of unobservable correlated flows, by creating the support for indistinguishability between income and outcome traffic in input and output Tor nodes. We based our approach on the use of k-anonymity input circuits that comprises a random managed multipath environment [27, 106] among multiple sources of Tor input connections. Also, we improve unobservability with the possible combination of server-side traffic through the use of Tor hidden service solutions, more recent available as a Tor facility [72].

To tackle the stated problem, we present the Trusted Input Relay proposal (TIR). TIR was designed as a censorship-resistant Tor pluggable transport candidate solution that leverages the notion of indistinguishable and tunnelled k-anonymity circuits used to decouple the input traffic from the origin user, segmenting it in multiple flows sent to multiple sources of Tor input nodes, by collaborative proxied connections. The idea is to use TIR nodes provided by communities of users distributed in different AS routing regions to organize an ad hoc pre-staging network for Tor input nodes, without modifying the Tor internal architecture and its operation principles. With the TIR solution, we aim at improving the use of the existent Tor network, to make it more robust against traffic correlation attacks in an enforced anonymization of the input user traffic. Our goal is to study the effectiveness of our designed and implemented solution, analysing its validity conditions for traffic unobservability purposes. For the validity, we must analyse if the solution is viable, not significantly damaging the system and internetworking performance, including latency and throughput conditions for practical use.

1.4 Main contributions

The main contributions of the thesis can be described as follows:

- Study of different techniques and the diversity of solutions proposed by the research community to fight against Internet censorship activities.
- Design and implementation of TIR — a censorship-resistant Tor solution. That leverages the notion of indistinguishable k-anonymity input circuits to decouple traffic from the origin user in multiple flows sent to collaborative proxied connections, that can use different Tor input nodes and input circuits.
- Promote dynamic adaptation capabilities in the TIR solution to manage the trade-off between unobservability, throughput, bandwidth, and resisting to possible network perturbations. These perturbations can derive from the normal behaviour of Tor latency issues, or can result from possible actions from censors.
- Implementation of the TIR solution and availability of the TIR prototype as a usable solution, in particular targeted as a base-contribution for our experimental assessment effort and for its use by researchers and practitioners for future research work improvements.
- Study and analysis of the validity of the TIR solution and its implementation, regarding unobservability metrics and performance metrics, including latency and throughput conditions.

1.5 Report organization

The remaining of the dissertation is organized as follows:

- **Chapter 2 — Related Work**: presents references and relevant background on solutions from the research on Internet censorship circumvention, providing a comprehensive description on such solutions, considering the dissertation goals.
- **Chapter 3 — System Model and Architecture**: is focused on the presentation and discussion of the system model and architecture assumptions in addressing our proposed objectives and in designing the TIR proposal.
- **Chapter 4 — Implementation and Prototype**: presents the implementation options, technology and practical details of the TIR prototype implementation.
- **Chapter 5 — Experimental Evaluation**: analyse the validity of the proposal, presenting our experimental evaluation effort and the obtained results for the validation of the TIR proposal and implementation.
- **Chapter 6 — Conclusions**: presents the main dissertation conclusions, also addressing findings and open issues for future work opportunities and research directions emerging from the dissertation work.

CHAPTER 2

RELATED WORK

This chapter addresses the presentation of relevant related work references for our dissertation objectives. The chapter covers the study of different Internet censorship circumvention techniques, Tor background and other interesting mechanisms found in the literature. The chapter is organized in the following way: first, we address different censorship circumvention techniques that have been proposed in the research (section 2.1); Then, we present Tor and a background on its operation and mechanisms (section 2.2); Following, we address concerns on Tor censorship circumvention and Tor traffic deanonymization techniques (section 2.3); In sequence, we address K-anonymity — a technique that has been used in different privacy-enhanced solutions and the hypothesis of using the notion in the context of Tor strengthening (section 2.4); To close the related work discussion we present a summary of the addressed related work and the analysis for the approach of the thesis objective (section 2.5).

2.1 Censorship circumvention

Internet censorship circumvention is the use of various methods, tools, systems, and related mechanisms, to evade internet censorship activities. The different solutions and methods available that emerged from the research community agenda are also different in terms of implementation requirements, difficulty of deployment, facility of use or operation, and resilience against censorship activities. From our study of the literature, we found a variety of solutions. We classified the different approaches in different types, ranging from the use of Internet proxies 2.1.1 or decoy routing solutions 2.1.2, mechanisms for covert-traffic obfuscation 2.1.3 and tunnel-based censorship circumvention 2.1.4. More recently, a novel approach of protocol tunnelling techniques to build covert channels encoded on multimedia-streaming protocols is perhaps the more recent and

effective approach against censorship. We discuss multimedia-streaming and traffic morphing solutions 2.1.5, the support for covert channels 2.1.6 and the materialization of media-morphed covert channels 2.1.7 and 2.1.8. In the following subsections, we present each one of the previous categories.

2.1.1 Internet proxies

Web proxying [44] is one of the simplest methods to bypass censorship, although there are a few problems related to this. It is essential not to expose the IP address of these circumvention proxies publicly, otherwise, a censor could detect and permanently block them, but to use these proxies, people must know about their existence, leading to several problems. A web proxy is a system that is configured to allow users to load external web pages through the proxy server, permitting the user to load the page as if it is coming from the proxy server and not the (blocked) source, however, depending on how the proxy is configured, a censor may be able to determine the pages loaded and determine that the user is using a proxy server. However, with the continuous updates of newer censorship methods, it is impossible to use these bridges. By traffic analysis, censoring entities can detect the protocol being used and block Tor bridges.

2.1.2 Decoy routing

This approach is different from the previous one, **decoy routing** [49] does not require a client to connect to a specific IP address (employing an end-to-end strategy which is easily blocked), to provide circumvention, the client communicates with a **decoy destination** [49]. The user selects a non-censored website that we call the overt destination, establishing a legitimate TLS connection using a friendly ISP (an entity that exists outside the censor's area of influence). The difference in this method is that it effectively uses steganography and public key cryptography to share a TLS master secret for this session between the client and the ISP, allowing the ISP (that owns a router somewhere on the path between the client and overt site) to man-in-the-middle this connection. After this happens, the censor as they are expecting sees just encrypted data that looks like it's heading to the overt site. However, this ISP is shuffling information back and forth between the client and the censored covert site. From the Point of View, censor see this legitimate TLS handshake happening, and then they observe encrypted data going back and forth between the client and the overt site.

Telex [102] is a decoy routing system, and its concept is to build end-to-middle proxying capabilities into the Internet's routing infrastructure, letting clients appeal proxying by establishing connections to regular, pre-existing servers. Telex has the potential to provide both greater resistance to blocking and higher performance than existing approaches. Avoids observable deviations from the behaviours of no proxied connections by applying this idea to a widely used encrypted transport (such as TLS), and we can construct a

robust service that allows users to bypass network-level censorship silently. This system promotes confidentiality, it's easy to deploy, transparent to users, and unblockable.

Cirripede [35] is a decoy routing system, can be used for hidden communication with Internet destinations, designed to be deployed by ISPs, and it scales to work with routers that handle large volumes of traffic while imposing minimal over-head on ISPs and not disrupting existing traffic. Allow proxies to be strategically deployed at central locations, requiring a lot of effort to block access to Cirripede. The client aims to communicate unobservable with a covert destination, to achieve this, Cirripede intercepts connections from clients to innocent-looking destinations and redirects them to the correct destinations, this communication is encoded in a way that is indistinguishable from regular communications to anyone without the master secret key. Meanwhile, public-key cryptography is used to abolish any need for secret information that must be shared with Cirripede users.

Slitheen [7] is another decoy routing system that has excellent resistance against replay attacks, latency analysis, and website fingerprinting attacks by maintaining packet size, timings, and directionality, which means, perfect mimicking the access to an allowed overt site. ISPs are needed to deploy its stations, to intercept specific traffic. The primary solution differences from the other approaches are that Slitheen as higher resistance against both passive and active attacks. The fundamental feature is to mimic regular requests and responses to/from uncensored destinations, which makes it harder for a censor to identify any possible censorship circumvention activities.

Waterfall [63] is the most recent solution for decoy routing and is partially different from the previous ones. Works using ISP as well and requires significant fewer decoy routers than the other ones. With this solution, only downstream are intercepted by decoy routers, which is why it's described as a downstream-only decoy routing system. While the remaining systems use upstream and downstream to send and receive blocked data, meaning there is a decoy router that interprets data from bidirectional streams.

The use of Internet proxies combined decoy-routing are two of the more initial solutions used to implement proxy-based censorship avoidance systems, which can piggyback sensitive information on available encrypted protocols that are allowed across a censor's borders in their AS routing regions of influence. A fundamental limitation of these systems, however, is that they rely on proxies operated by third parties (which must be also publicly available and discoverable by DNS querying and public IP addressing). These solutions can hamper the scalability conditions of use, since the user must know and trust proxies offered by those third parties and their provided connections, which are sometimes not necessarily powerful in terms of throughput and latency, as well as in the processing power and resources for internetworking processing. On the other hand, there is the danger for the user that the traffic interchanged with such proxies can be monitored by censors, being also possible for the censor to easily conduct denial of service attacks of those proxies are detected. This happens with particular emphasis on end-to-end traffic monitoring activities, when a user sends traffic to proxies in the same AS routing region where the censor has a powerful influence.

2.1.3 Covert traffic obfuscation

Censorship circumvention by covert traffic obfuscation uses specific techniques capable of morphing traffic [101]. The functionality of these methods is adding padding to packets, splitting data into smaller packets, and change ciphertext formats. There are two of many existent possibilities of traffic obfuscation we intended to study, one is a Skype protocol mimicked solution, and the other (more complex) is a resort to probabilistic automata to obfuscate traffic.

SkypeMorph [56] is a traffic obfuscation system, pluggable transport for Tor that disguises client-to-bridge connections as Skype video traffic, it is also possible to adapt the system for other UDP-based protocols [42].

Marionette [68] is the first programmable obfuscation system to offer users the ability to control traffic features ranging from the format of individual application-layer messages to statistical features of connections to dependencies among multiple connections. It is possible to control ciphertext formats and, so, create templates for ciphertext using template grammars, which are probabilistic context-free grammars (CFG).

2.1.4 Tunnel-based censorship circumvention

Tunnel-based censorship circumvention approaches use different mediums to tunnel covert traffic and rely on third-party servers, that are unaware of the system's procedure.

Meek [28] is a Tor pluggable transport that combines a technique called domain fronting, with a simple HTTP-based tunnelling proxy to send covert traffic to a Tor relay, which will then forward the traffic to the final endpoint (better described in section 2.2.2).

CDNBrowsing [39] is a reference censorship circumvention approach dependent on the use of content delivery networks (CDN) joined with tunnelling techniques. A later solution, **CDNReaper** [41], attempts to illuminate a portion of the at first recognized issues with the CDNBrowsing approach.

This kind of technique comprises getting to block web assets that are accessible in certain CDN. Given that a CDN edge server may have similar IP addresses for different contents, it implies that censor would not have the option to block that addresses, since it would make other uncensored web assets be similarly blocked. In addition, in this kind of approach, and censor could attempt to disturb the DNS, by checking and blocking certain censored content. However, that can still be avoided by getting to that censored content easily in other known edge servers, in this manner preventing the DNS resolution phase. Through the encryption-related with protocols like HTTPS, a censor is likewise unequipped for performing deep-packet inspection attacks and effectively distinguish the nearness of censorship circumvention mechanisms.

2.1.5 Multimedia streaming and traffic morphing

Multimedia streaming circumvention approaches apply to video and audio morphing techniques to encode web data into multimedia streams. **FreeWave** [36] is an audio-based system, designed to embed covert data in multimedia protocols through the modulation of audio signals sent through VoIP streams. However, this solution could be trivially detected by an adversary [81], so instead multimedia protocol tunnelling systems such as Facet [51], CovertCast [58], and DeltaShaper [8] resolve this problem, modulating data while striving to preserve the unobservability of the generated covert channels.**Facet** system's circumvents censorship through a video morphing technique, permitting users to watch censored videos over a Variable Bit Rate (VBR) video stream. Facet embeds covert video frames into a small region of the video frame area of a video conferencing Skype call, effectively allowing a client to watch blocked videos, this handled at the server-side.

CovertCast uses video streaming as a message carrier; it enables the content of blocked websites to be transmitted via modulated images of live-streaming feeds uploaded to sites like YouTube and Twitch. This approach is somewhat different from Facet, but both have some limitations, which that they do not support the transmission of covert TCP/IP streams because the format of secret messages is restricted to specific data types, namely videos (in Facet) or web content (in CovertCast).

DeltaShaper uses the Skype conferencing tool to trade covert data. The idea is to encode Internet packets into video streams, which are captured by a virtual camera and sent by Skype to the target host, at last, this host extracts the packets from the received video stream, acting as a proxy to the final endpoint.

Protozoa [21] is a censorship-resistant tunnelling tool that follows the system model of a general multimedia covert streaming (MCS) tool, the carrier application comprises a web application that uses the WebRTC framework for providing a point-to-point live-streaming service between its users (e.g., Whereby). Typically, such an application consists of a backend that handles the signalling and session establishment of video calls between participants, and client-side JavaScript and HTML code that starts video calls and manages the video transmission via the WebRTC API provided by the browser. Protozoa will use the resulting media stream as the carrier for a covert channels.

2.1.6 Covert channels

Data exfiltration is when an individual takes data off the system or network without permission, this is often accomplished with malware where and individual plants a malicious piece of code on the system. That code attempts to identify valuable information and then exfiltrate it out of the system using a covert-channel.

A **covert-channel** [58, 79] is a way of transmitting information using methods that were not at first intended for data transmission. Covert channels are in general unauthorized and hidden, and they are used to sending information in a way that violates your security policy, there are two different types of covert channels, covert timing channel

and covert storage channels. In a covert timing channel, a process will relay information by modulating the use of system resources based on different timing. With a covert storage channel, a process writes data to a storage location where another process of a lower clearance level can read it. For example, if an attacker is sending malicious traffic through port 80, so the firewall assumes that it is an HTTP web server traffic and allows the malicious traffic to flow through the firewall to the web server. A covert-channel attack is where we use a channel that was not intended for communication to transmit data, with a covert storage attack, we have a subject at a low-security level who can read data at a high-security level that they should not be able to access. With a covert timing attack, information is transmitted by altering a system resource's performance or timing, it's a way of hiding messages and mixing it in with legitimate traffic that is travelling over the network, and it would not be seen because it is mixed with legitimate traffic. One example is where an attacker can hide a small amount of data in a packet header for data that is already being transmitted across the network and would not look suspicious.

Data leakage protection or data loss prevention (DLP) is a tool to prevent sensitive data from leaving an enterprise's network. DLP content-aware policies can scan for proprietary information or protected data such as in person identifiable information like social security numbers or sensitive documents and prevent users from disseminating it outside the corporate network. DLP products can track violations of this policy back to individuals users and notify administrators, and this provides accountability if there is either malicious transferring or accidental transferring of the sensitive data.

The materialization of covert channels is possible by using different techniques. Many solutions for covert channels are based on some kind of overlaying. In such solutions, covert channels are piggybacked on top of overlayed protocols for traffic obfuscation, encoded in independent layers, supported by covert tunnels or mixed in the carrying protocols themselves. In the next sections, we discuss some examples of these approaches.

2.1.7 Media-morphed covert channels

Media-morphed covert channel [9] enables the creation of covert channels by modulating data into the input of popular multimedia applications (e.g., Skype, YouTube). Protocol tunnelling must have unobservability to make an adversary incapable of distinguishing the streams that carry a covert-channel from those that do not. In any case, existing multimedia protocol tunnelling systems have been evaluated using ad hoc methods, which throws questions on whether such systems are secure, for instance, for censorship-resistant communication. To be more precise, this technique consists of covert encoding data into the video (as well as audio) channel of popular encrypted streaming applications, without requiring any progressions to the carrier application. Systems, for example, Facet [51], Covert-Cast [58] and DeltaShaper [8] implement this technique, and present various methodologies for data modulation that aim at raising the difficulty of an adversary to recognize covert data transmissions. A significant property that all these systems

strive to accomplish is unobservability. In practice, a multimedia protocol tunnelling system that provides a high degree of unobservability keeps an adversary from flagging a large portion of covert flows (for instance, from attaining a high true positive rate) while flagging a low measure of regular traffic (i.e., while accomplishing low false positive rate).

In [9], the authors' examination uncovers that some state-of-the-art systems are flawed. Specifically, CovertCast streams can be distinguished with few false positives by an adversary, even when depending on existing similarity-based classifiers. While the other systems exhibit various levels of unobservability as per their definition, they show that none of the existing employed similarity-based classifiers can identify suck channels without acquiring in enormous quantities of false positives. In addition, they conclude that one of the current similarity-based classifiers reliably outperforms all others in the undertaking of covert detection channels. Second, they show that ML strategies based on the decision trees and a portion of their variants are incredible compelling at identifying covert traffic with decreased false positive rates. For instance, an adversary employing XGBoost [104] would have the option to flag 90% of all Facet traffic while wrong flagging only 2% of genuine connections. Also, the performance of such techniques is very high, implying that the adversary can classify traffic almost instantly, with a relatively low number of samples per training set, and taking a low memory footprint.

Furthermore, the use of decision tree-based techniques allows us to comprehend which traffic features are in general significant for detecting the functioning of specific multimedia protocol tunnelling systems. These discoveries recommend that, aside from their performance, decision tree-based techniques can give a considerable understanding of the inner working of these systems and that they should be used for assessing the unobservability of multimedia protocol tunnelling systems later on. Third, they investigated alternative ML approaches for the recognition of covert channels when the adversary is expected to be at most or denied of labelled data. Their discoveries recommend that unsupervised learning techniques give no advantage to the classification of multimedia protocol tunnelling covert channels. In contrast, the application of semi-supervised learning techniques yields a critical division of false positives.

2.1.8 Tools for Internet media-morphed covert channels

Facet allows users to observe random videos by replacing the audio and video feed of Skype video calls. To view a video, users contact a Facet server by sending it a message containing the ideal video URL. Next, the Facet server downloads the requested video and feeds its content to microphone and camera emulators. At that point, the server puts a video call to the user, transmitting the chose video and audio. Accordingly, users are not required to install any software to use the system. For approximating the traffic patterns of regular video calls, Facet re-samples the audio frequency. It overlays the ideal video in a small amount of each frame while the rest of the frame area fills up by a video resembling a regular video call. Decreasing the area involved by the disguised video

translates into expanded resistance against traffic analysis.

CovertCast scratches and balances the content of web pages into pictures that are dispersed through live-streaming platforms (for example, YouTube). Numerous users can consume the data being transmitted in a specific live stream at the same time. CovertCast regulates web content by encoding it into coloured matrix pictures. A coloured matrix defines by cell size (neighbouring pixels with a given colour), the number of bits encoded in every cell (represented with a colour), and the rate at which a matrix containing new data loads. Users scratch and demodulate the pictures served through the live stream, extracting the ideal web content.

DeltaShaper separates itself from the previous systems in that it takes into account tunnelling arbitrary TCP/IP traffic. This is accomplished by modulating covert data into pictures, which are transmitted through a bidirectional Skype video call. DeltaShaper follows a comparative data encoding system to that of CovertCast. In any case, and corresponding to Facet, a coloured matrix is overlayed in a division of the cell screen, on top of a regular chat video running in the background. This overlay, named payload frame, can be cautious defined to give different levels of resistance against traffic analysis. On-call start, DeltaShaper experiences an alignment stage for changing its encoding boundaries as indicated by the current network conditions to safeguard unobservability. An adversary faces an innate exchange off between the capacity to accurately identify countless covert channels and to wrong flag real flows. Flagging actual flows as covert channels are something that the adversary needs to maintain a strategic distance from in most useful settings. For instances, a censor that targets blocking flows containing covert channels may not be eager to block huge parts of real calls, that are used by day by organizations and business, as these calls, might be key for the economy of the blue censor's regime [22]. Likewise, law-enforcement offices may not be happy to hazard to legitimate flag activities of residents as criminal actions dishonestly.

In [9], various strategies were most part used the following metrics: true positive rate, false positive rate, accuracy, and the area under the ROC curve. The True Positive Rate (TPR) measures the fraction of positive samples that are accurately recognized in such. In contrast, the False Positive Rate (FPR) mismeasures the extent of negative samples classified as positive. Accordingly, adversaries will endeavour to acquire a high TPR and a low FPR when performing covert traffic classification. Exactness captures the fraction of correct labels output by the classifier among all predictions. It can be used as a synopsis of the classification performance, since high precision suggests a high true positive rate and a low false positive rate.

By this study, the overall features of these techniques are that: Facet is more and more defenceless against analysis based on packet lengths and burst behaviour, DeltaShaper is increasingly helpless against analysis based on packet lengths. CoverCast fails to provide unobservability, CoverCast fails to provide unobservability, Facet covert channels can be spotted by searching for packets on a threshold of 115-195 bytes and DeltaShaper covert channels can be detected by searching for packets on a limit of 85-100 and 1105-1205

bytes.

At last, we have **Protozoa**, a censorship-resistant tunnelling tool, which has a high-performing covert channels and strong traffic analysis resistance. To create a covert-channel, a user only needs to make a video call with a trusted party located outside the censored region using a popular WebRTC (a free, open-source project that provides web browsers and mobile applications with real-time communication (RTC) via simple application programming interfaces (APIs)) streaming service, (e.g., Whereby). Protozoa can then covert tunnel all IP traffic from unmodified user applications (such as Google Chrome) via WebRTC video stream. It hooks into the WebRTC stack and replaces the encoded video frame data with the IP packet payload, but the WebRTC stream payload remains encrypted, and the stream statistics properties are all identical to those of any common video call. This technique enables popular Internet applications, such as web browsing and mass data transfer, and avoid detection by sophisticated traffic analysis attacks. The authors propose that Protozoa can evade state-level censorship in China, India, and Russia.

2.2 The Tor network

Tor is today the most popular and well-known anonymization network used in the internet environment. At a glance, Tor is a low-throughput overlay network allowing users to surf the Internet anonymously, through the possible use of Tor-based tools available developed by the Tor development community [71] and distributed as open-source software. The Tor web browser is the more popular of such tools.

Tor operates as a volunteer and global overlay network in the Internet environment, comprising thousands of relays using proxies and onion-routing techniques implemented by public available Tor bridges, in order to conceal information such as network locations, DNS names and related IP endpoints. Tor also allows for the anonymization of used resources on the Internet, such as web servers or web-enabled applications. However, at the same time, Tor is also an open project, continuous evolving its different mechanisms for the enhancement of the Tor circumvention and privacy-preservation solutions.

Not surprising, Tor has been used for the good and for the bad goals. It is extensively used by the research community as a "Lab" and testbench environment to inspire the development of different ideas, in an "arms race" between censorship circumvention and censorship effectiveness.

However, similar to other low-throughput anonymity internetworking systems using proxies or decoy-routing strategies to route traffic, Tor is also inherent vulnerable to traffic correlation attacks, which enable an attacker to match a source IP with a destination IP, allowing the traffic deanonymization of ongoing communications. This can be accomplished by a state-level adversary capable of observing both endpoints of a Tor established circuit (between an end-user and a targeted used resource), by correlating the corresponding outgoing and incoming flows through advanced traffic analysis techniques.

Although this vulnerability is a well-known weakness in Tor's design, this system has nevertheless been considered secure for many years now due to the practical difficulty of mounting said attacks in practice.

In the next sections, we resent a background on the Tor operation and the mechanism used for the anonymity purpose.

2.2.1 Tor operation and Onion-Routing

Tor usage makes the trace of internet activities difficult, and it's intended use is to protect the personal privacy of its users, as well as their freedom and ability to conduct confidential communication by keeping their internet activities unmonitored.

Onion routing is the core principle of Tor, it is the fundamental key to achieve all of this, implemented by encryption in the application layer of a communication protocol stack, nested like the layers of an onion, anonymizing TCP-based applications like web browsing, secure shell, and instant messaging. Clients choose a path through the network and build a circuit, in which each node (or "onion router") in the path only knows its predecessor and successor. Tor encrypts the data, including the following node destination IP address, as many times as the nodes in the comprising successive, random-selected (active) Tor relays and sends it thought this virtual circuit with fixed/size cells, which are unwrapped by a symmetric key at each node.

Relay cells contain end-to-end data exchanged between the relays. Control cells are interpreted by the OR that receives them (e.g., extend circuits). Relays can transport TCP segments and can multiplex multiple TCP streams [25] along each circuit. This multiplexing facility improves efficiency and contributes to anonymity. Relay cells also use end-to-end integrity checksums, allowing last relays to identify possible tampered or defective cells that will be discarded.

The final relay decrypts the deepest layer of encryption and sends the original data to its destination without the acknowledgement of the source IP address. Since the steering of the communication was incompletely concealed at each hop within the Tor circuit, this strategy disposes of any single point at which the communicating peers can be determined through network surveillance that depends upon knowing its source and destination. [26]

Three types of nodes exist in the Tor network: directory servers, relays, and bridge relays. **Directory servers** [25] is composed of autonomous groups and provide global information of Tor relays. **Relays** run by volunteers (all around the world running Onion Routers), and they are listed at directory servers and used to build paths for anonymous communications. **Bridge relays** are not recorded in directory servers and are used as dynamic entry nodes to prevent the blocking of public relays by a few ISPs. As mentioned above, Tor uses a small group of relays as directory authorities, located in different internet regions, and addressed in different routing domains (in Internet Autonomous Systems or AS). Each AS is a collection of connected IP routing prefixes, under the control of one or more network operators acting on behalf of a single administrative entity or

routing domain. AS have common and clear defined and agreed on routing policies to the Internet, and is accomplished by the operators involved [37].

Tor directory authorities maintain a digital signed document of the current Tor network status, including relays list and their certificates of public keys. Clients can obtain the information of the available authorities and then can fetch the status document from any directory authority, to obtain a list of available relays for possible selection. In the current operation, Tor also added directory caches, which can obtain a copy of the Tor status from directory authorities, to be provided for interested clients [95].

In general, authorities or clients can fetch directories from caches for bandwidth optimization. Periodically, onion routers sign and publish their router descriptors [95]. The OR descriptors contain their public keys, capabilities, and other optional information. Directory authorities gather the router descriptors to generate a signed current view of the network and send a summary of that view to other directory authorities. All summaries are then voted to build a final signed document, designated as the Consensus Document. Formally, this document describes the current agreed status of the existing Tor network [95].

To establish a circuit, the clients obtain a list of the current relays and then exchange symmetric session keys with each OR for the selected circuit, one at a time. This process is known as the telescoping path-build design [25]. The symmetric keys are valid during a Tor session. ORs discard keys whenever the circuit closes, to provide perfect forward secrecy guarantees in the re-establishment of new circuits. Tor is today compatible with the majority of TCP-applications that can be used without any modifications or kernel requirements of client machines. By providing a SOCKS interface [96], Tor allows, for example, the support for email client applications to be used on top of Tor, as used on the Internet, only requiring the proper configuration of a proxy relay.

In addition, to the use of regular TCP based applications, the Tor community has been developed privacy-enhanced applications. It is presented below as an example of Tor browser usage.

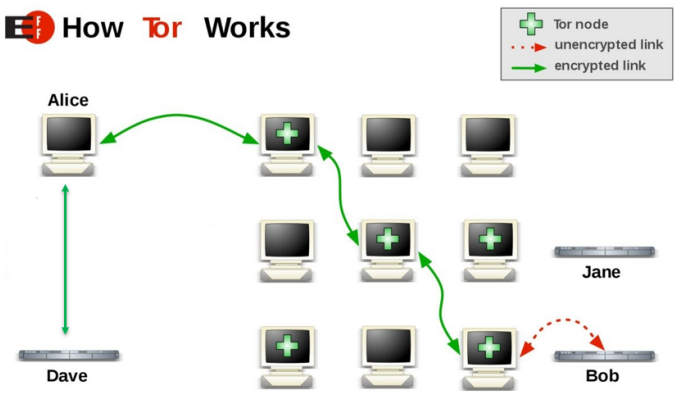


Figure 2.1: How Tor works (taken from [96]).

In figure 2.1 we have a client named Alice, that is using Tor Browser to access information on the target destination Jane and Bob servers. To do so, Alice request Dave (an entity that can contact Tor directory) for Tor nodes, giving Alice the acknowledgment of three nodes and an essential piece of metadata called circuit ID or CID (the number that identifies and associates Tor relays with the client). The client establishes a symmetric key between each of those nodes. Before a packet is sent through the relay nodes, the client must change the packet source IP address from Alice to the IP address of the last relay node (exit node), because this node will be the one who contacts the destination. We don't want to expose the real IP address of Alice on the internet. The last step is to consecutive encrypt the packet using the symmetric key of each node making three-layer encryption, start encrypting using a symmetric key of the exit node, next is the middle node and at last the entry node. This packet has been encrypted three times and is ready to send across the Tor relays. The Alice connects randomly to at least one of the publicly listed entry nodes, bounces that traffic through the selected middle relay, and eventually spits out your traffic through the third and final exit node. Because of this specific encryption order, each node can decrypt with his symmetric correspondent key and send the packet to their successor node until it reaches the final destination. The packet that is sent from the exit node to the target is the original one, which is not encrypted.

The response is similar to the request, and Bob sends a response packet to Alice using the same path on the Tor relays, sending the packet to the exit node, this node encrypts the packet with his symmetric key. The exit node sends to the middle node and so on until Alice gets the packet from the entry node and sequentially decrypts in using the Tor relays symmetric keys to get the original packet. If Alice communicates with another website, this process repeats and the Tor relay path will be different, with random-selected Tor relays and different CID.

2.2.2 Tor Bridges and Pluggable Transports

Given the secrecy improvements provided by Tor traffic flows exchanged on the Internet, many users adopt the system as an Internet circumvention tool used as a countermeasure against censored regimes and to access blocked websites or servers. Hence, Tor is a major target of this blocking problem, by the ISPs of such regimes. To block Tor traffic, Tor puts the affected Tor relays in blacklists. Since the relay's list is public, it's trivial to create blacklists of the respective IP addresses, preventing Tor clients from establishing the required circuits. To avoid this simple attack, Tor employs the notion of Tor bridges [97]. A Tor Bridge consists of unlisted proxies, whose goal is to forward, as an intermediary, a client's traffic to a certain entry relay. Instead of connecting to some potentially blacklisted entry relay, clients connect instead to some unlisted bridge (hopefully) not expectable, known by the adversary.

In the real operation of Tor, some relays are public, while others are private. Public

relays are usually deployed by volunteers to be used to by any Tor user (or client). However, private relays (bridges) are exclusive for who knows about their existence, under secrecy distribution conditions [59].

Unfortunately, bridges may be vulnerable to traffic fingerprinting attacks. As an example, if a user connects to a bridge using the vanilla Tor protocol, it becomes possible for an adversary to identify the Tor traffic, and them, the adversary will create a blacklist for that bridge. In [59], shows that 55% of public Tor bridges usable on the web are at risk of aggressive blocking strategies. This happens because the Tor traffic has some unique and observable properties that great facilitates the job of ISPs in identifying such traffic. Some evidences that are easy for detection are:

- Tor uses fixed-size 512-byte cells, which results in a characteristic packet size distribution easy analysed by an adversary, to identify, with high probability, that such flow carries Tor cells or not.
- Tor traffic may be detected through TLS byte patterns. As an example, many Tor communications between relays are encrypted using TLS, with Server Name Indication (SNI) used as TLS extension [78]. This extension adds the domain name to the TLS header to be used in the Client Hello handshake phase. We must notice that the SNI isn't encrypted within the handshake phase, allowing an eavesdropper to find the domain name that the client is trying to reach.
- Tor uses a fixed random string as a bogus domain name (e.g., www.kf3iammnyp.com).
- In TLS handshakes, Tor usually establishes well-known cipher suite sets [92].

The conjugation of the above properties uniquely identifies with high probability the encrypted traffic as Tor's, enabling censors to block new bridges in some minutes after they have been deployed. In the study presented in [92], the detection can be very effective even if only two of the above evidences are detected.

As a countermeasure against traffic identification, Tor bridges can also support pluggable transports (PTs) [78]. PTs are obfuscation wrappers that shield the Tor traffic between a client and a bridge to avoid traffic analysis. The most mainstream PTs are materialized by add-on tools for relays, such as meek [28] and obfs4 [67], an upgrade of the previous obfs2 [65] and obfs3 [66] tools.

Obfs4 provides a level of obfuscation to existing authenticated protocols, like SSH or TLS. The obfuscation protocol encrypts all traffic using a symmetric key shared between the bridge and the client, derived from both clients and bridge's initial key [92]. Later, the attacker would see just a randomly encrypted byte stream. In the case of meek, it uses a technique called Domain Fronting (or DF) [28].

Domain Fronting consists of using different domain names in the SNI and in the HTTP Host to encode a legitimate website domain name into the SNI (e.g., the address of a public cloud) and use the Host field in the encrypted HTTP request, to ask the access

to the forbidden one. To be used, meek requires a CDN (Content Delivery Network), like offered by public cloud providers, supporting domain fronting. To help many clients, the Tor Bridges can support multiple PTs. Bridges that offer more than one PT may decrease the security of the most secure PTs they offer. Also, bridges running non-Tor services (e.g., SSH) in addition decrease the level of security given to clients [59].

2.2.3 Tor output circuits and server-side anonymization

In the objectives of the dissertation, we primarily focus on the enforcement of anonymization, considering the Tor input circuits used by clients. One question arises in this objective, considering that censorship activities can be targeted on traffic correlation income and outcome network traffic. For a better understanding of our focus, we must explain how Tor protects today the outbound traffic and server-side anonymity (for so-called Hidden Servers), in the more recent implementations of Tor nodes and internetworking operation. In fact, although in the scenario presented above in the base-operation involving input, intermediary and output nodes, the receiver's IP address must be exposed so that the client can contact it. Once a Tor circuit has been established, the exit node needs to learn the IP address of the receiver so that it can proxy the client's connections accordingly. Hence, Tor circuits in the base operation, as at first explained, do not provide enforced receiver anonymity. To overcome the limitation, Tor introduced, more recently, new mechanisms based on extended onion services [25] and new rendezvous relay nodes, enabling the deployment of public servers that can be reachable through TCP/IP to provide standard services (e.g., over HTTPS or FTPS) while preserving the IP address of the server anonymous. To this end, the key idea is to leverage Tor circuits as the building block so that both the sender and the receiver create two independent circuits, which connect to those intermediate relay nodes called the rendezvous point (RP) and HS Entry Guard Nodes (HSEG).

By executing a set of specific initialization protocols, a client willing to connect to a receiver target (or final server) uses not the specific IP address of the receiver, but an onion service address, which is based on a public key owned by that receiver. Using this address (only seen as an asymmetric cryptographic public-key, perhaps using different cryptographic algorithms, such as RSA, DSA or ECDSA), the client can choose any relay in the Tor network to elect as the RP and then establish a circuit with such RP node. At this point, the RP does not know the IP address of the client. The receiver has then the ability to locate the RP and establish its own circuit to the client selected RP node. Because a Tor circuit provides sender anonymity, the RP itself does not know the real IP address of the receiver. Once the RP connects to both the sender and receiver through two independent circuits, all it does is to forward the traffic between both endpoints, therefore providing mutual anonymity guarantees.

This Tor enforced protocol for client-server mutual anonymity is in short illustrated in the following Figure2.2. Due to its complexity, we omit for summarization purposes

many details, focusing on the essential operation, e.g., not addressing complementary issues like name resolution of onion services and location of the RP by the receiver and direct mapping between public keys and real-DNS full qualified names or IP addresses. For more details, we refer the interested reader to these details explained in [25] or [48].

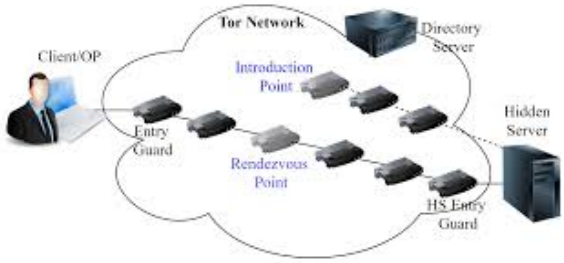


Figure 2.2: Components of server-side anonymity enforcement for hidden services (taken from [48]).

HSEG Nodes can be hosted in a node inside or outside the Tor network, with an increase of these nodes available and widespread on the Internet and in different AS-regions all around the world. These nodes usually have a DNS top-level domain name called onion.

2.2.4 Tor hidden services

Hidden servers (HS) are configured to publish a service descriptor of the HS on the Tor DS nodes. This service descriptor contains the HS's public key, expiration time, and the selected introduction points (called INTRO nodes). INTRO Nodes are random nodes selected by the HS at the start of the connection establishment process. Once the HS has selected an INTRO node point, it provides the HS's public key to the introduction point. To avoid any impact from Denial of Service (DOS) attacks against a single introduction point, the HS usually selects several of them. The HS then advertises the selected INTRO points in so-called Hidden Service Directories (HSDirs) and signs the advertisement with the HS's public key. HSDirs are in essence special types of DSs (directory servers), with some specific properties used to publish the service descriptors of HSs. We must notice that INTRO nodes do not know the IP address of the HS as they are connected to the HS via a Tor circuit, which also comprises different relays. Then, the HS owner can advertise the onion address over the public Internet, allowing potential clients to find out about this service from the web (including dark-web), secret informal means or other ad hoc similar ways. When the user searches a particular onion address in their browser, the service descriptor from the directory service is obtained and starts the connection establishment process.

To summarize, a hidden service works like this, taken from [12]:

1. A hidden service calculates its key pair (private and public key, asymmetric encryption).

2. Then the hidden service picks some relays as its introduction points.
3. It tells its public key to those introduction points over Tor circuits.
4. After that, the hidden-service creates a hidden service descriptor, containing its public key and what its introduction points are.
5. The hidden service assign the hidden service descriptor with its private key.
6. It then uploads the hidden service descriptor to a distributed hash table (DHT).
7. Clients learn the *.onion* address from a hidden service out-of-band. (e.g., public website) (A *$hash.onion* is a 16 character name derived from the service's public key.)
8. After retrieving the *.onion* address, the client connects to the DHT and asks for that $hash.
9. If it exists, the client learns about the hidden service's public key and its introduction points.
10. The client picks a relay at random to build a circuit to it, to tell it a one-time secret. The picked relay acts as a rendezvous point.
11. The client creates an introduction message, containing the address of the rendezvous point and the one-time secret, before encrypting the message with the hidden service's public key
12. The client sends its message over a Tor circuit to one of the introduction points, demanding it to be forwarded to the hidden service.
13. The hidden service decrypts the introduced message with its private key to learn about the rendezvous point and the one-time secret.
14. The hidden service creates a rendezvous message, containing the one-time secret, and sends it over a circuit to the rendezvous point.
15. The rendezvous point tells the client that a connection was established.
16. Client and hidden service talk to each other over this rendezvous point. All traffic is end-to-end encrypted, and the rendezvous point just relays it back and forth. Note that each of them, client and hidden service, build a circuit to the rendezvous point; at three hops per circuit this makes six hops in total.

With the strong anonymity enforcement, Tor has captured the attention of individuals engaged, more and more, in illicit activities in the provisioning of server-side resources, being today one relevant improvement of the foundations for the "dark web" [23]. In

particular, Tor has been used for running anonymous online marketplaces with, leveraged by such mutual client/server anonymization of illicit goods, behind onion services and complementary RP nodes, such as Silk Road [16], and allowing end-users to access them and purchase such goods in better anonymity conditions. Past studies [18, 40, 82, 103] have reported also many other illicit activities fostered by new web-based onion services and RP relay nodes, unfortunately, sometimes involving violence, markets of illegal weapons and drugs, child pornography, hacking shared tools and methods, and also illicit finance operations.

Unsurprisingly, in recent years (after 2015) this has prompted yet more governmental authorities to investigate illegal affairs involving Tor communications in its current diverse support nodes, by looking for ways to deanonymize suspicious Tor circuits. Using strengthened rendezvous relays distributed across different global AS regions, in a dynamic binding environment where such nodes can have a volatile existence, compared with the base operation, has shown that the server anonymization improved the difficulties for the censorship and authorities involved in surveillance activities [6].

2.3 Traffic correlation in Tor

Some of the most powerful classes of attacks for Tor traffic deanonymization are described in the literature as traffic correlation attacks. In general, a correlation attack is an end-to-end attack where an adversary searches for a correlation of traffic attributes between two flows on a given entry and an exit relay. Since an entry relay connects direct with a client and the exit relay connects direct with a targeted server, an attacker can conclude that the enumerated client is accessing the observed server if it finds that two observed flows are correlated by the observed traffic features. Censors acting in the context of state-level adversary models can be in particular focused on being able to launch such attacks due to their privileged control over a country's network or the authority control over a certain AS routing region. We in brief describe the more effective strategies that can be associated with correlation attacks to Tor known today, relaying on actions that can be fired in censorship activities, and the concerns and countermeasures involved.

2.3.1 Tor traffic analysis by circuit-fingerprinting

Circuit fingerprinting is the possibility to exploit Tor traffic fingerprinting on established circuits. This can be done with approaches based on asymmetric traffic analysis. This means that the adversary can correlate and deanonymize Tor circuit's endpoints accessing to different directions of the flow, e.g., from client to entry and server to exit. The rationale of this concern is based on the fact that the Internet routing strategy relies on possible asymmetric connections, i.e., the path from the client to the server may differ from the same server back to the client. An interesting conclusion for circuit fingerprinting is that an adversary is still able to perform traffic correlation by observing the different

directions of a given flow. One example is to conduct a BGP routing interception attack, which can be launched by a malicious AS region (where in order to divert traffic, enable traffic analysis, and forward traffic to the original destination.

2.3.2 Timing attacks

Timing attacks are end-to-end active attacks where an attacker attempts to determine the endpoints of a circuit by correlating the time it takes for a flow to travel from the entry to the exit relay. Some techniques for timing attacks conducted by censors employ single-end bandwidth estimation for deanonymizing the source of a given Tor connection. The censor places probe servers alongside the egress and ingress routers at the boundary of AS regions. Assuming that the client sends traffic through colluding servers controlled by the adversary, the server will then send traffic along with a pattern that can be detected by the attacker's probing nodes for their observation and correlation. In this way, the attacker can discover the AS of the client and may also escalate the attack to pinpoint the current user location.

A different timing attack is based on the disruption of the AES-CTR counter used to encrypt Tor cells. This attack requires the entry and exit node to be controlled by an attacker, offered as an apparently good Tor node. The attack works by duplicating a cell in the entry node, which causes a decryption error at the exit node. Then, the attacker can correlate the time of the cell transmission with the time of the decryption error to ensure that the error was caused by the cell duplication and at last identify the source of the originated connection.

In the literature, we can find some proposals of defensive countermeasures in order to mitigate timing attacks. One well-known strategy is the so-called perfect interference state. In this strategy, the output streams should all have the same shape, e.g., using threshold mix batching. Unfortunately, Tor does not operate with threshold mix batching, which relies on waiting for several streams and then flush them, all at once as a form of traffic-aggregations. Such a technique also increases the latency, defeating the low-throughput characteristic expected by the Tor users. To prevent timing attacks, Tor relays send cells internally through the nodes, from different streams, in a round-robin fashion.

2.3.3 Sybil-oriented attacks

The Sybil-oriented attack involves multiple malicious relays controlled remotely by the attacker, giving the appearance of belonging to different identities. The objective is to obtain a large disproportional network influence to divert user traffic to those malicious relays disseminating those identities. In fact, a wealth of attacks on Tor, such as correlation attacks or timing attacks, depending on the amount of network traffic the attacker can attract and control, together with the ability to trick users into using the promoted malicious relays.

Defences against Sybil attack types are difficult for effective countermeasures. It is always possible to perform the attacks unless a central authority is used to assure a trusted binding between a relay and a trusted identity. However, the use of central authorities is something not compatible with Tor's goal, giving the commitment to the elimination of any single point of control. The acceptance of central authorities in Tor would increase difficulties on new relays' deployment. Furthermore, it could be necessary to have some form of software attestation running on relays by the involved central authorities. Another approach against Sybil attacks is to limit the number of new accepted relays at directory authorities. This could be addressed by minimizing the number of relays by IP address subnet. In this case, an attacker would need to access multiple subnets to conduct the attack with success. Both solutions do not direct stop Sybil attacks, acting more as mitigations by increasing the attacker's startup and deployment cost

2.3.4 Correlation-based analysis

The first correlation attacks are based on timing analysis. Inter-packet timing information is usually not carefully protected, since it would require delaying packets to hide timing patterns. This is not good for performance and can induce inopportune jitter conditions. An attacker can exploit this timing property by correlating the inter-packet time on both endpoint links. The goal is to conclude that those links belong to the same circuit, which would tie a user (in the source) to the corresponding targeted resource (in the destination). The resilience of low-throughput anonymous systems regarding correlation attacks based on packet interval time is not particularly good. Some observations by conducting experiments using HTTP traces show that inter-packet times of input and output packets can easily correlate both flows. For example, the attacker starts by observing a set of entry and exit relays in order to link entries to exit nodes. With observations of 60 seconds, the attacker divides the time into a fixed size window. Then the attacker counts, during each time window, the number of observed packets. The correlation between any two sequences is then computed using a cross-correlation metric. If the metric extends some threshold, the attacker can infer that both samples belong to the same flow. Another source of correlations is based on the packet-length that are maintained in ingress and egress flows.

For example, state-of-the-artwork in performing flow correlation use machine learning algorithms, instead of statistical metrics, to conduct accurate flow correlation on Tor. Congestion may cause networks to suffer from bandwidth reductions, thus leading Tor packets to experience fragmentation, hence decreasing the effectiveness of existent statistical flow correlation techniques. However, DeepCorr learns a correlation function that is able to link flow samples regardless of their destination, while accounting for the possible unpredictability of the Tor network.

One interesting approach for countermeasures consists of using intermediate Tor relays injecting dummy packets with the goal of normalization of statistical information, a

technique called adaptive traffic padding. The injected padding would reduce the ability of an adversary to fingerprinting packets on a circuit, and to protect against traffic correlation attacks based on machine learning techniques. However, this requires the modification of Tor relays (modifying its current behaviour), enforcing the use of pluggable transports across all relays, instead of just on bridges as provided today in the vanilla Tor solution. The technique also introduces performance penalties. Other recent and promising countermeasures against traffic correlation attacks rely on employing AS-aware relay selection mechanisms

2.3.5 Correlation concerns and avoidance

By controlling the entry and exit nodes of existing Tor circuits, an adversary focused on breaking private and anonymous communication may be able to deanonymize the involved endpoints, for instance identifying the IP addresses of corresponding senders and receivers in traffic flow, and is achievable by using multiple techniques.

To mitigate adversary's attempts to launch such attacks, different solutions emerged in the literature in which, the focus is to keep users from using unsafe relay nodes (or nodes strongly monitored by such adversaries of possible "honeypot" nodes disseminated by those adversaries as "Tor friendly-nodes". We will summarize the proposed countermeasures in the following categories:

Use of co-located trust-controlled entry relay nodes, the user can run a trusted local relay node alongside the Tor client, using that relay as an entry node to the Tor circuits that are chosen and created by the user. The downstream relay nodes won't have the option to decide if the traffic sent by the trusted relay node was initially created by the local user or by another user that might be using additionally a similar relay node for building its circuits. The approach needs that every user must run a relay node that can't be broadly implemented, partly due to the difficulties in configuring such systems, but mostly because Tor clients situates behind NAT-boxes and restrictive firewalls and afterward they can't relay the required traffic [96].

Discovering and Labelling bad relay nodes, to decrease the chance of users in selecting malicious relays, there is a detection mechanism reporting bad relay nodes. A relay is considered bad if it's marked as malicious, misconfigured, or unreliable. To mark relays, Tor uses a set of labels, regularly assigned as flags [94].

As a major aspect of the Tor activity, nodes are acting as authorities, distributed alongside the internet, that can decide in favour of applying those flags. The pre-owned systems choose the vote dependent on the estimation of various features, for example, bandwidth, uptime, and other reliability or performance measurements, and marked geolocation of IP addresses. From the data gave in such directories, a user must choose relays as per their position on the circuit. For instance, an entry relay should be set as extensive. Otherwise, an attacker can set up a malicious relay and begin to efficiently find users' identities.

Tor maintainers also run a service focused on finding potentially unsafe relays [93] and attempt to suggest the Tor clients' community with refreshed information (that can be received automatically by some existent Tor-based client applications, for example, Tor-browser plugins). For this reason, they also seek to duplicate advised issues and conceivably attempt to connect with the relay operator when something is turning out badly. If the issue can't be settled, Tor maintainers will assign a flag — e.BadExit — to a reported relay, in this manner informing the users to not use it any further later on as an exit node.

In general, Tor maintainers and clients cooperate in scanning the network and seeking for "bad relays", especially "bad exit nodes". In this effort, the bandwidth conditions when using the observed relay nodes are continuously monitored by the Tor directory authorities. The idea is to avoid malicious relays from lying about their bandwidth (using various estimations taken from different locations and correlating such measurements). Simultaneously, for load balancing purposes, clients can use also the provided information to pick relays according to their measured bandwidth capacity limit. In general, the authorities compute weights related to each relay class (entry nodes, middle nodes, and exit nodes) given the current bandwidth capacity of observed relays. This keeps a malicious relay from advertising, for instance, a "fake" 100 Gbps bandwidth availability that would make it possibly eligible to be chosen by various clients, creating a sort of a sinkhole. Together, different sinkholes controlled by adversaries can shape a "wormhole" network, attracting many clients, allowing such adversaries to control a lot of Tor flows exchanged from clients and servers.

There are different auxiliary tools made and disseminated by the Tor user's community, implementing node scanning and flagging features, for instance, there are tools like exitmap [69], torscanner [1] and tortunnel [99] for scanning and flagging candidates for exit nodes, tools to decoy traffic to detect bad relays [80].

Client-based restriction control of usable sets of trusted nodes, Tor employs entry guard relays [24] to protect users from this danger. A guard relay is chosen from a restrictive list of relays when making the first hop of circuits. The idea is that the client will use just those relays as entry nodes (for a long period). If the adversary doesn't control (or can't observe) the guard, then the client can stay secure. Otherwise, the adversary can indeed see a larger fraction of the client's traffic, with the opportunity of avoiding profiling dropping significantly to a probability of about (N-C)/N. Assuming that an adversary controls, or observes, C relays and assuming that the total number of relays in the Tor network is N, if every time a client uses the network selects a new entry relay and exit relay, this implies the adversary can correlate the traffic sent by the client with a probability of about $(C/N)^2$. This implies that there is a C/N chance of connecting with the first relay and $(C-1)$ / $(N-1)$ chance for the last relay. Then by selecting many random entry and exit nodes, the user can incur in a situation where her/his communications could be profiled sooner or later.

Before, users picked an entry (guard) relay from a list of three relays, with each guard

commonly discarded of the following 60 days. The rotation of guards is an avoidance countermeasure against attacks and to disseminate traffic loads over numerous guards. Specifically, if a user was unfortunate and chosen a malicious guard, he got the opportunity of recovering anonymity while changing the current guard.

The issue is that those boundaries were not sufficiently able to be utilized today against robust AS-level or omniscient adversaries, beginning an issue known as the Guard Rotation Weakness [24]. Elahi et al. [89] Empirically exhibited that traditionally Tor's time-based guard rotation criteria prompted users to exchange guard relays more regularly than they should, expanding the chance of profiling attacks. From their perceptions, the creators in [89] notice two fundamental dangers: when a guard replaces another guard because of inaccessibility labelling, and when the guard rotation happens following 60 days. In the principal situation, choosing a malicious relay to supplant an inaccessible one doesn't present a prompt danger. The malicious relay sets toward the finish of a list containing the three selectable guards. On the off chance that the inaccessible guard becomes available up once more, at that point the malicious relay is disposed of and replaced with the past guard. Choosing a malicious relay in the second situation is a critical threat since the malicious guard relay will be used on different occasions by the user. For a superior exchange off between anonymity preservation and load-adjusting for proficiency, Tor clients are currently prescribed to keep a similar guard relay for a significant period. The suggestions point today to times of nine months as reference. To be chosen as a guard, a relay must meet an insignificant bandwidth limit, its uptime must be more prominent than the median overall possible relays, and the node probably been available in the network consensus for at least fourteen days.

Avoidance of unsafe autonomous systems using AS monitoring and tuning, all the previously presented techniques may not be effective against an adversary with the power of observation of vast fractions of the network. We must see that such adversaries don't have to control a particular relay node to deanonymize Tor traffic, since they can do it by eavesdropping on the inter-relay traffic that crosses the controlled network [3, 47]. The threat here is the capacity of the adversary to control (or to gain control) of the routing structure and Autonomous Systems (ASes) including the Tor circuits.

With this viewpoint, a few researchers have proposed another defensive approach focusing on mechanism acting against AS-level adversaries, i.e., those entities with the ability to access the network infrastructure of an entire Autonomous System (AS) in IP routing-management. Typically, the proposed approaches attempt to help Tor clients' to choose paths away from such possible malicious ASes, by using the analysis of the Internet topology limits and by observing inter-relay latencies [17].

The strategy **Avoidance by path Prediction of possible dangerous Autonomous Systems** is interesting for the mitigation of correlation attacks. As observed by a few authors, correlation attacks could be mitigated by an AS-awareness control provided by path selection algorithms for decreasing the chance of an AS-level attacker to observe traffic flowing between the two endpoints of a Tor circuit [54, 64, 105].

Large, routing-capable adversaries, for example, country states can censor and dispatch robust deanonymization attacks against Tor circuits that navigate their borders. Tor allows users to specify a set of countries to avoid circuit selection. However, this gives simply the illusion of control, as it doesn't block those countries from being on the way between nodes in a circuit. **Avoidance of Untrustable Geolocated Regions**, DeTor [52] is recommended, as a lot of strategies for proving when a Tor circuit has avoided user-specified geographic regions. DeTor extends ongoing work on using speed-of-light limitations to demonstrate that a round-trip of communication physically couldn't have crossed certain geographic regions. As such, DeTor doesn't expect changes to the Tor protocol, nor does it require a guide of the Internet's topology. DeTor can be used to maintain a strategic distance from censors (by never travelling the censor once) and to abstain from timing-based deanonymization attacks (by failing to transit a geographic area twice). DeTor effectively discovers avoidance circuits through recreation using real latencies from Tor.

2.4 Tor strengthening with K-anonymized circuits

We can expect a state-level omniscient adversary model that can observe the whole Tor network. To avoid this, our intuition is to guarantee that deanonymizing a given circuit through flow correlation won't give away the identity of the first sender unequivocally, yet just as one possible potential candidate among K plausible users that could be responsible for the communication. If it is difficult to identify precisely the starting point, and have a variable number (possibly high) of such K possibilities, we can improve the difficulty for the censor (or blocker) in pursuing a practical approach for the deanonymization goal. Besides, the K included endpoints in this membership can diverge. If we arrive at a solution wherein these variations can happen dynamically, as a reconfigurable multipath environment of entry circuits during the processing of the same traffic flows, we can plan to improve Tor circuits to give K-anonymity and arbitrarily chosen routes. To motivate this technique, it is interesting to review the use of K-anonymity techniques and how these mechanisms have been applied in various application domains.

2.4.1 K-Anonymity for privacy-enhanced databases

In Samarati and Sweeney [83], K-anonymity was introduced as a mechanism to solve the problem of disclosing privacy-sensitive database records that should have been anonymized. A practical issue addresses, in this case, is to avoid private database records (ex., person-specific field-organized data), can be accessed, creating a release of the data, ensuring that individuals who are the subjects of the data can't be re-identified, while the data remains useful for particular use. For this situation, K-anonymity is a property of released data such that the information about any given individual can't be recognized from at least $(K-1)$ other individuals. It follows that for a larger value of K, the anonymity

set will be bigger, giving that anonymity is more robust. Since it was first proposed in 1998, k-anonymity has been extensively studied, for example, to create strategies for k-anonymization in different contexts: datasets [11], craft new attacks to k-anonymized databases [5], and alternative or composable anonymity definitions [15].

2.4.2 K-Anonymity in location services

K-anonymity has also been used for securing the protection of clients' locations in the context of privacy-preserving Location Based Services (LBS) [76]. In this application, the idea is to accomplish k-anonymization of location queries by bypassing the LBS from recognizing a single sender on a specific location, based on received queries. This solution is interesting when users need to send their location query containing a user identifier, location, and time to the LBS. To provide a query containing a user identifier, a Central Anonymity Server (CAS) is used to remove the user identifiers, obfuscating the location-time pair, is given queries in a certain "window" (called a cloaking region). For the proposed reason, the "window" is considered as adequately large for containing several users.

2.4.3 K-Anonymity in social networks

Given the popularity of social networks and the concerns about the preservation of security, organizations were requested to have better solutions to secure users' personal data, accessible on or from these platforms. For this situation, the solution isn't so simple as removing individual identifiers before publishing, because other information can be more effectively used to infer the user identification. A possibility to identify individuals based on their neighbourhood (e.g., based on social friends list) is addressed in [2] using an anonymization approach based on a K-anonymity model not merely to ensure the users' attributes (personal information) but also users' structural information (which includes neighbourhood information).

2.4.4 K-Anonymity in processing big data

K-anonymity is currently used in the context of big data computation [62] because the data submitted to the related computation platforms usually includes data containing personal and private details about users and their private assets. Data must be safely secure, preventing confidentiality leaks, integrity violations, and it should remain available whenever required. One significant property of privacy-enhanced big data computations is to perform queries and hence use that data to improve user experience on a given application or service, for instance using no identifiable data, no-inferable correlations during operations, or permitting possible processing facilities well-executed over homomorphically-encrypted data using practical partial-homomorphic cryptographic methods and algorithms supporting specific types of operations. However, much of the

time and for some useful operations, those techniques would be impractical to query that information later on or processed with low-throughput, high-throughput for real-time and online requirements. To beat such limitations, K-anonymity can be used as a critical component and can be helpful to cluster personal information in anonymized and not-linkable groups, to avoid query-inferences and providing efficiency in the way the data can be queried.

2.5 Summary and discussion

The studied related work was a source of inspiration for the contribution of our dissertation. We learned that Tor, despite its interesting mechanisms or the possibility to improve Tor bridges with mechanisms inspired from the related work, is inherently vulnerable today to traffic correlation attacks which enable an attacker to match a source IP with a destination IP, allowing for traffic deanonymization. This can be accomplished by a state-level adversary capable of observing both endpoints of a Tor circuit and by correlating the corresponding outgoing and incoming flows through traffic analysis, considering different types of traffic features. This is a vulnerability particularly exploitable when the state-level adversary with authority over certain AS-routing regions on the Internet. Although this situation is a well-known weakness in Tor's design and current operation, this system has nevertheless been considered secure for many years now due to the practical difficulty of mounting such attacks in practical use and real-time operation. However, the complexity of attacks and difficulty of using advanced machine learning traffic analysis tools has been steadily decreasing as a relatively small number of large ISPs has dramatically expanded their network infrastructures. This expansion is now in a point that, today, they are able to intercept most of the Internet traffic, including Tor's, as well as, to apply forms of machine-learning correlations in real-time traffic flows. Unfortunately, no practical defences are currently known without requiring significant changes to Tor's protocols or infrastructure, which can be a difficult task. To overcome the problem, a possible idea is to work on pluggable transport solutions that can leverage the notion of indistinguishable flows to modulate the Tor input traffic on multiple segments used as a pre-staged environment where multiple cooperating K clients (distributed on different AS-routing regions) can provide this facility based on the k-anonymity notion. Put simply the hypothesis, a state-level adversary that can intercept the entirety of the Tor traffic (looking to ingress traffic on input Tor bridges in its controlled AS-region), will not be able to deanonymize the real source behind pre-staged segments of K-anonymized circuits from amongst a set of K equally plausible origins (in different AS regions). In the next chapter, we will describe our system model assumptions and architecture to address such a solution.

2.5.1 Related work analysis

In the different sections of the related work, we addressed dimensions of related work references studied for the objectives of the dissertation. From the introduced references, we can state that Tor (just as others low-throughput anonymized internetworked solutions) can have relevant vulnerabilities to traffic correlation attacks, whereby an adversary with the capacity to intercept entry and exit nodes can distinguish the initiator of the communication. As an outcome, those adversaries can observe features of traffic-flows infused in the network, deanonymize the circuits subjacent to those traffic flows, or block the identified circuits. To reduce the chance of such correlation attacks, our approach is to avoid unsafe relay nodes, for example, by deploying multiple trusted relays as clients endpoints, always scan and advertise bad relay nodes, and enforce reliable guards by limiting the rotation period in an apparent random-membership of insider nodes, when seen from outside entities. Even though avoiding unsafe relays can diminish the probability of flow correlation attacks, such defences can barely adapt to large AS-level or state-level omniscient adversaries. Such adversaries can perform today successful flow traffic correlation using advanced unsupervised machine-learning techniques, by intercepting the Tor traffic in the observation of entry and exit circuits, without the need to hijack the included nodes — a situation that remains undetectable by the owners of such relay nodes.

To manage the considered attacks, the related work addresses various ways for clients to keep away from possibly unsafe ASes. Techniques that are based on monitoring routing data [43] yet isn't trivial nor efficient for clients to effectively identify changes in the routing structure of the Internet, namely BGP changes over the routing paths. Alternative strategies to decrease the chance of falling prey to correlation attacks are to use geographical avoidance, which refers to avoiding paths belonging to specific regions [52] but this is also hard to be addressed with sufficient guarantees.

Finally, the introduced defences can't withstand powerful adversaries that can observe the traffic of the whole Tor network (or a big part of it, involved in specific traffic flows). This weakness results from a fundamental design choice whereby Tor offers unicast point-to-point channels: a single sender connects to one single receiver through a specific Tor circuit path.

2.5.2 Thesis goal and hypothesis

In the study of the presented related work, we analysed different censorship-resistance techniques to overcome Tor vulnerability to correlation attacks and preserve unobservability on entry circuits. Our aim is to improve Tor by overcoming the unicast limitation, mainly focusing on entry circuits. To provide some degree of anonymity, our idea is to adopt a multipath strategy combined with K-Anonymization of used circuits in the client-side and with the server-side anonymization Tor solution, using hidden service, where

other Tor users can connect to these onion services, each without knowing the other's network identity.

K-Anonymization is a technique that was proposed in the past to address privacy-preservation with regard to different applications [2, 27, 62, 83, 106]. However, the dissertation can be the first approach in addressing a K-anonymity approach into Tor, so that a single circuit can be connected with multiple anonymized K senders, each of them could be equally responsible for starting a circuit as a covert-channel. In our envisaged solution, these covert channels can be tunnelled with possible heterogeneous protocols, in addressing a solution for K-anonymized traffic morphing multi-channels for Tor entry circuits. The plan to achieve this is by designing and building up an add-on system to Tor that can be materialized as a plugin-facility for Tor clients (or client proxies), but not requiring the adjustment of existing Tor protocols or the already existent software components (such as Tor-based protected applications).

In the arms race involving censors and solutions for server-side anonymization [48], more recent solutions improved the Tor network with complementary exit-bridges on exit nodes, used as short-lived proxies, alternatively used as egress points [107] a solution that can also be used together in established circuits between the receivers (server targets) and rendezvous nodes. Some authors observe that the mechanisms in 2.2.3 for server-side anonymization achieved a better traffic anonymity protection against traffic correlation involving traffic-inspection of exit circuits, with an indirect improvement of client-side privacy. Given the current true positive rate (TPR) support existent today for the anonymization of receivers (targeted servers) and circuits between Tor exit nodes and those receivers, we decided also to improve the anonymization in the client-side generated traffic relevant protection for users that don't have the possibility to deploy and control the enforcement of anonymization guarantees in the remaining Tor overlay segments. The idea is to provide such protection in such a way that the use of receiver-side anonymization via Tor rendezvous nodes or other exit-node anonymization enforcement techniques [48, 107] can be also orthogonally used.

2.5.3 Principles for the thesis approach

From the aspects discussed above, we retained different mechanisms that have been proposed for internet censorship circumvention. Some of the described mechanisms have been impacted by the evolution of the Tor system as an anonymity network solution. However, Tor is inherently vulnerable to traffic correlation attacks, which enable an attacker to match a source IP with a destination IP of Tor communications. As it can be observed, different categories of attacks to the Tor network can be categorized into two distinct groups: using malicious Tor entities and using advanced traffic analysis techniques. We must notice that correlation-based attacks do not depend on the attackers' ability to compromise entities and some involved activities are much harder to detect and mitigate, even when the related traffic monitoring is made on encrypted traffic. To

overcome the problem, our hypothesis follows the idea behind the proposal of pluggable transport mechanisms that leverages the notion of indistinguishable flows to modulate the Tor input traffic decoupled in multiple segments, managed by cooperative K clients to provide k-anonymity. A primary concern for our solution is not to modify the internal structure of Tor relay nodes. The goal is to design a pre-staging input ecosystem, only requiring the volunteering involvement of communities of users, to implement those K-anonymity input circuits, to decouple the entire traffic from origin users, in multiple traffic on multipath routes supported by a set of K equally plausible Tor circuit sources, using different Tor input nodes.

Chapter 3

System Model and Architecture

In this chapter, we address a system model approach for the elaboration phase. Our proposal relies on the designed system capable of offering K-anonymity multipath routing and server-side anonymity. This is possible through the use of secure tunnelling anonymized techniques and Tor hidden services.

The current chapter is divided into six different main sections, starting with the objectives and main contributions presented in chapter 1, as the goals and requirements (section 3.1), followed by the system model definition for the implementation phase (section 3.2). Then, we introduce some guidelines related to censorship modelling (section 3.3). After understanding the system and threat model, we explain the system architecture and components (section 3.4). At last, we conclude mentioning the integrated solution (section 3.5) for the proposed software and concretizing our objectives summarized in (section 3.6).

3.1 Goals and requirements

As early presented, we focus the dissertation on enhancing the Tor ecosystem with countermeasures against traffic correlation attacks, with particular attention on protecting anonymity by avoiding correlations between input circuits and output targets. For this purpose, we will consider two primary integrated defences: **K-anonymization** of input circuits in observing traffic of output targets and improvement of unobservability capabilities by using Tor **hidden services** for the server-side.

For K-anonymization of input circuits, the idea is to give a user the possibility of coordination with a collaboration group of K endpoint-senders, acting together to create indistinguishable outgoing traffic flows, and thus protecting the real sender identity in each moment. The actual sender and the used $(K-1)$ helpers will shape traffic such that

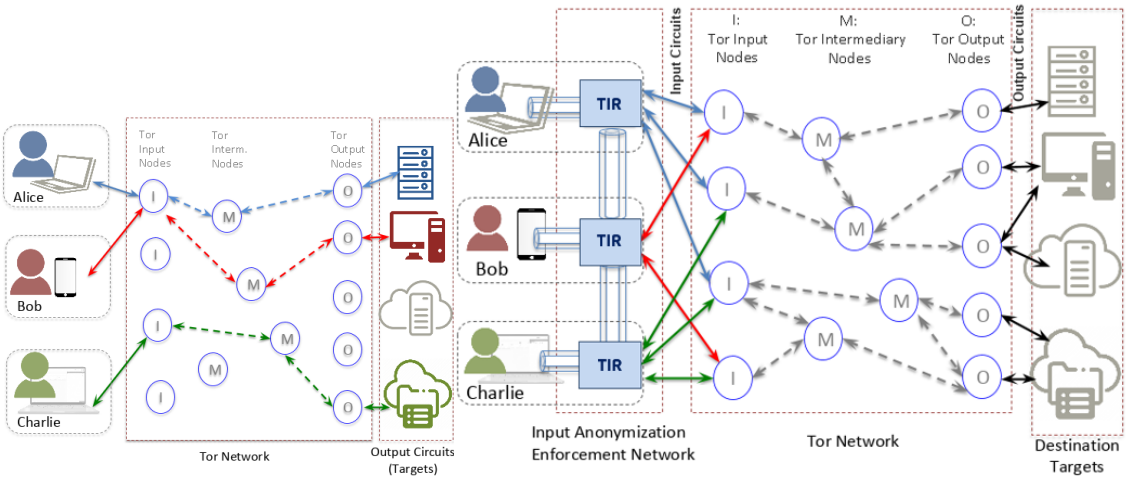
they achieve the indistinguishability of the senders. These K-Anonymous circuits can support the traffic in a random-walk way. From the viewpoint of an external censor (or adversary), it must be impossible to correlate traffic to the original sender. It must be impracticable to know about the "apparent" random schedule strategy, in forwarding the traffic by the multipath routes coordinated and controlled by the sender. In this direction, we consider the general idea behind the use of K-anonymity solutions [2, 62, 76, 83, 106], combined with a randomly chosen multipath routing strategy (using peer-collaborators as onion-routers).

The major challenges in the above's concretization objectives will be:

- The design and implementation of each one of the two combined facets that we will intend to accommodate in the proposed solution.
- The combination of the two facets itself, to have an integrated solution where we will use K-Anonymity input circuits used as a multipath controlled onion routing strategy and the Tor hidden service for the final web server.
- The compatibility of the proposed solution to be supported in the Tor network, without the modification of the pre-existent implementation and, in the limit, with a simple addition of an input-processing component in Tor nodes that can be used as input nodes. To achieve our goals without shattering the overall system's performance when compared with the typical performance achieved by the current use of Tor.
- The solution is generic enough and configurable to support the use of Tor hidden service and accessible for all group of collaborative k endpoint-senders, without revealing the web server IP address to its users, guaranteeing unobservability.
- Support for interactive TCP, UDP, TLS and DTLS traffic. We intended to enlarge the range of applications, so we can have various types of clients interacting with our system, and we can achieve it given the design choice of intercepting packets at a data link level. But it is also important to ensure an interactive communication, so that the user interaction at an application-level remains below a certain connection speed [33, 61].

3.2 System model definition

The following figures (3.1a and 3.1b) represent the considered system model approach for the proposed solution. figure 3.1a is a simple representation of the use of Tor vanilla circuits, where final users (senders) use destination targets (example, web servers), using the structure of onion-routing established by the Tor relay nodes (typically two or three relays, with one input-relay, one optional intermediary relay, and one output relay).



(a) Tor Vanilla. (b) Our solution, with TIR nodes.

Figure 3.1: Use of Tor as an Onion-Routing based Anonymization Network.

In the vanilla model represented in figure 3.1a, Alice constructed and uses a circuit (blue arrows) to access the final a.com target, processing HTTPS traffic supported by the onion-routing structure of the used Tor relays. Likewise, Bob constructed and used a circuit (red arrows) to access also to b.com. Charlie built and used a circuit (green arrows) to access the resources in d.com.

In figure 3.1b, we show how we will intend to enhance the previous Tor vanilla ecosystem by adding an input anonymization enforcement network. In this model, Alice, for example, sends data to her TIR through a socket. The TIR determines whether the data is to be sent to an input relay in Tor or to another TIR via a tunnelled channel, then to an input relay in Tor. After the data is sent from TIR to a relay, the TIR acts like a Tor client and starts the standard Tor procedure. The same process applies to Bob and Charlie.

As discussed in the related work, the use of Tor (in the current vanilla setting) is vulnerable to traffic correlation attacks whereby an adversary with the ability to intercept entry and exit nodes can identify (and then, the adversary can censor or block the traffic) from the initiator of the communication. Using trusted and collaborative input relays (at the level of the K-anonymization input circuits), we intend to mitigate the use of unsafe relay nodes. Used in a multipath strategy of using trusted input relays, which communicate with each other via secure tunnelling, supporting onion-routing designated as TIR. TIRs can be in or out of the AS region of the sender, and in this case, with the minimization of the effect of censorship activities conducted from powerful, omniscient adversaries controlling traffic in entire geolocations or AS routing regions.

In the intended solution, the TIR nodes can also continuous scan and advertise bad Tor relay nodes, or enforce reliable guards by limiting the rotation period (or onion-routing validity period) of their use. Alternating Tor relays from different AS regions, likewise, as today some Tor client software solutions do and as the research literature suggests. The purpose of TIR nodes is to avoid weaknesses arising from Tor's unicast point-to-point

input channel design: one sender connected in each moment to one input relay through an established Tor circuit, and by avoiding the possibility of traffic observability in such input circuits.

Figure 3.2 represents a simplified system architecture. In this diagram, we can see the software and environment support communication that is a TIR in the input Anonymization enforcement network (shown in figure 3.1b). Starting with the **App (1...n)** that represents the Tor enabled applications, which can be Tor applications or standard browser, this region represents the level of application of a TIR. Although Tor only supports the transport of TCP-based protocol, we wanted to widen the range of communication protocols, so TIR supports applications that use TCP/IP and UDP/IP base's transport layer protocol. The **TIR middleware layer** handles the requested operation from the clients, selecting the strategy to define the type of traffic tunnel channel. The **IP (outbound/inbound) layer** handles the requests and delivers of file data from Tor network or from another TIR inside the TIR Community. The TIR architecture is detailed in section 3.4.

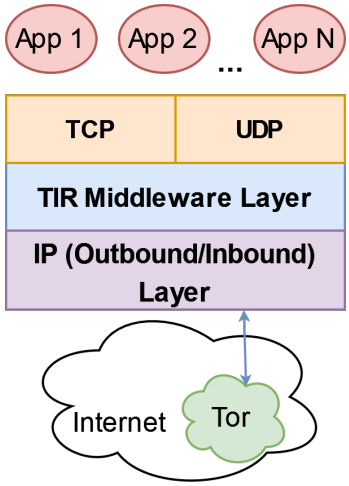


Figure 3.2: Simplified TIR diagram.

3.3 Threat modelling for censorship avoidance

In this section, we characterize the threat model (or adversary), which is a typical method of characterizing the security properties of a security system, however, for our situation, the adversaries are censors. These entities may act on the behalf of an oppressive government to block or monitoring access to certain content considered prohibited by the regime, following an adversary model typology and define a censor as being a state-level adversary.

3.3.1 Censorship modelling

We consider it realistic that a censor can observe, delay, alter, drop, or add communication in a variety of ways. However, a censor that only passively observes can be significant and illuminating. We limit our description and attentions to a passive end-to-end correlating adversary: one that learns source or destination of communication when in position to observe either or both, and that always links observations of the same communication flow anywhere in its path. This connection occurs regardless of how the flow's appearance may have changed en route. We consider a censor that may actively add network resources or corrupt existing resources. But we do not consider any addition, alteration, or disruption of network traffic directed over those added adversary resources. Nor do we consider adversarial removal or degradation of network resources in this thesis.

A censor is an entity that wants to break the non-observability of the injected traffic of the Tor network, intends to correlate outgoing and incoming traffic from the observation he may have from traffic samples at two points to correlate the traffic that enters from an end-user and the traffic that leaves from the node (outgoing relay) of the Tor network. To do this, the censor can be articulated in a censorship environment in which distinct entities can participate, these can be ISP or another type of entity that are collaborating in the attempt to make these correlations. This is described as a state-level adversary, this opponent, however, is not omnipresent in several AS regions on the internet, it has the possibility of having limited targets in the area in which it operates, it can probably only observe traffic within one or more service providers or in the aggregate traffic of a set of service providers that participate in this collaboration's censorship. But it cannot do the same thing on an AS region of the internet that includes other providers, so there is no notion here of an omnipresent adversary, an individual capable to indiscriminate do this at all circuits of the internet.

Even though our censor has access to a vast quantity of resources, in this thesis, his actions are limited and presented in the following issues are outside the scope of our defined censor:

- We expect the adversary cannot decipher the packets sent through the clients and the TIR's and Tor input relays, which implies except if the censor is acting as a user of our censorship circumvention solution, he cannot have access to the media.

- The adversary (censor) is only interested in producing samples, he will behave in a way that will only make passive attacks in the traffic analysis, he will not intrude on the nodes in presence, neither on the final targets nor on the users. In the system model presented above, we assume that the TIR are reliable and are not attackable by the adversary, in special, he will not try to corrupt the software, so it means, the communication endpoints where TIR run are deemed trusted. To be clear that the adversary does not have this chance, we will not have a defence against this adversary, it is outside our adversary model.

- We consider out of this censorship model, the chance of having software or hardware installed in client's machines that are deemed malicious, implying that we assume our censorship circumvention solution to run consistently on machines with trusted software stacks and confided hardware devices.

3.3.2 Capabilities and actions from censors

The adversary (or censor) can observe the traffic at the circuit's entry and exit relay and tries to find patterns in the communication channel to match outgoing and incoming data to deanonymize users (as presented in figure 3.3). By intercepting the traffic at these relay nodes, the adversary can use various features, for example, correlating the volume of traffic data, protocol typology, or inter-packet arrival times, to perform different results of correlation analysis. It may correlate two flows to adequacy results, at that point, the adversary can decide with a severe potential extent of conviction the IP locations of the observed circuits and endpoints.

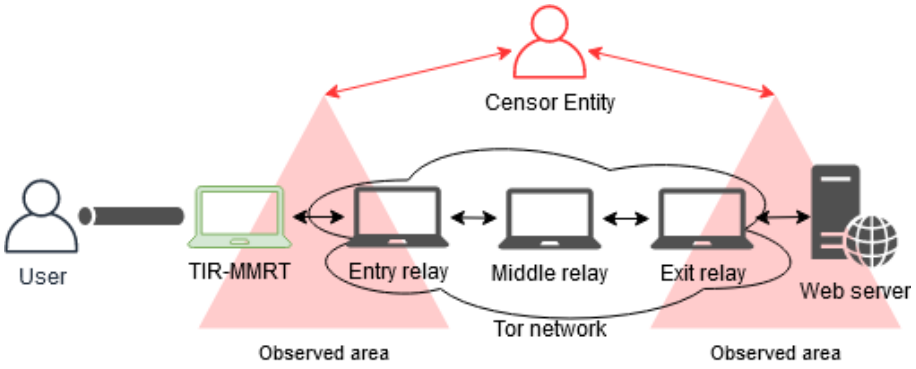


Figure 3.3: Censor model approach

3.3.3 Censorship vectors

Censors are treated as a black box, drawing inferences about their internal workings from their externally visible characteristics. The easiest thing to learn is the censors *what* are the destinations and contents that are blocked. Somewhat harder is the investigation into *where* and *how* the specific technical mechanisms used to effect censorship and where they are deployed in the network. Most difficult to infer is the *why*, the motivations, and goals that underlie an apparatus of censorship. Censors change over time, and not always toward more restrictions. Censorship differs across countries, not only in subject, but in mechanism and motivation. However, we are reasonable to assume that a basic set of capabilities that our threat model censors have:

- **Active attacks** are out of our threat model, as we previously mentioned.
- **Passive attacks** are based on monitoring and analyse network traffic, trying to detect any discrepancies or signs that uncover the access to prohibited content or

use of Internet censorship circumvention tools. We consider our adversary capable of recurring to a few techniques that can perform these attacks:

- **Correlation attacks** — this attack empower the observation of source IPs in inbound circuits with a destination IP in outbound circuits, explained in subsection 3.3.4.
- **Deep packet inspection (DPI)** or packet sniffing — is a type of data processing that inspects in detail the data being sent over a computer network, and usually takes action by blocking, re-routing, or logging it accordingly.

3.3.4 Correlation attacks

As described in Chapter 1 and 2, the Tor network is a low-throughput anonymous network, providing both user anonymity and network performance. To provide a pleasant user experience, Tor focuses on providing low communication delays while preserving the anonymity of its users. However, low Round Trip Time (R.T.T.) comes with a price. Since Tor attempts to optimize the communication delays between peers, the traffic characteristics such as the R.T.T and throughput at one end of the anonymous path gets reproduced at the other end of the path. This makes the Tor network vulnerable to correlation attacks for a censor who can eavesdrop the communication channels. Correlation attacks are used to derive a wide range of information. It could extract information about the identity of the communicating peers and also be used to profile the status or the habits of a specific target. And yet, it could extract the information transferred in a specific communication. Therefore, traffic analysis is a serious threat to the primary aim of Tor which is preserving users and services anonymity.

Correlation attacks are well-known de-anonymization attacks. In this category of attacks, it's assumed that the attacker controls both the entry node and the exit node of the circuit between the client and the server. The attacker is looking for a correlation in traffic between the entry node and the exit node because then he can conclude that the entry node and the exit node participate in the circuit. The entry node knows the client, the exit node knows the server, so the attacker can confirm that the client and the server are communicating.

For the attack to be executed, the adversary requires a means to intercept packets flowing through the output connections. Malicious ISPs or governments may have the capabilities to intercept packets when they control (part of) a network. The attack works as follows. First, the adversary records the packet interarrival times on all output connections of the targeted relay nodes. Second, the interarrival times are transferred into so-called pattern vectors that in effect contain the number of packets per batching interval as their elements. This is also done for the known input stream. The transformation process that is used depends on the batching strategy that is used. For example, if packets are sent per specified time-out interval, the number of packets per unit of time for each

time-out interval are the elements of the vector. Third, the distance between the input stream and all the output connections is calculated using the pattern vectors. There are propose two measures for doing this. The first one is mutual information. The second one is frequency analysis. Fourth and last, the output connection which has the minimum distance to the known input stream is selected as the output connection that corresponds to the known input stream. One last thing worth noting about this attack is that the amount of available data (i.e., the amount of recorded packet interarrival times) is important. Large amounts of data can lead to detection rates near 100%. This is still true when a lot of cross-traffic is present.

To prevent this attacks, we leveraged (presented in 3.2) the notion of unobservable correlated flows, by creating the necessary support for indistinguishability on income and outcome Tor circuits, in such a way that input and output circuits will be managed with an enforced anonymity solution. The solution (TIR network) is based on the use of the k-anonymity channel among multiple sources of Tor input relay connections, that is also improved for unobservability with the use of the randomly-managed multipath input circuits and traffic-morphing solution.

3.3.5 Countermeasures for censorship circumvention

Table 3.1: System countermeasures.

Attacks	Solution
Correlation attacks	To mitigate this threat, we developed a system capable of combining **K-anonymization multipath input** and **Tor hidden service**. By using K-anonymization, we avoid the surveillance and identification of each input circuits between senders and TIRs, in the entire context of Tor onion-route structure and its correlation with the outbound, observed traffic from the whole Tor circuits. By using Tor hidden service, we create unobservable traffic route data to and from this onion service, even those hosted behind firewalls or network address translators (NAT), while preserving the anonymity of both parties (the TIR and web server).
Deep packet inspection	The countermeasure is to use multiple routes through several endpoints (in the multipoint circuit between TIRs) since it's considered that the censor (attacker) does not have the possibility to have access and produce traffic analysis on all routes and thus cannot aggregate (as these circuits can always pass through AS routing regions not controlled by the attacker), which would only be available to a censor characterized as "omnipresent", with the ability to produce traffic analysis across the internet (and thus, all potential circuits that may be established by the network of the TIRs) or in the initial segment of access in the network of each TIR that is sending traffic through the multiple routes, but, such an attacker is outside the definition of our threat model. In addition, the Tor hidden service solution further reinforces this possibility, because in this case, the connection to an onion service is encrypted end-to-end, so the traffic packets are encrypted if the user access onion sites or clearnet https ones.

3.4 System architecture and components

Based on the above-described guiding principles of the system model (in Section 3.2), in this section we elaborate and explain the main modules in the system architecture and its components, and probe our understanding of the role they play in the overall system.

3.4.1 System model and architecture overview

In the following figure 3.1b, we sketch the generic model of our proposed solution. We can divide this figure into three parts:

- The first part contains the group of clients and their respective TIRs, which are in a network called **input anonymization enforcement network**, distributed world-wide and set up in a voluntary community base. The communication with clients and their TIRs are defined using sockets and the supported transport layer are TCP and UDP protocols and application layer are TLS and DTLS security protocols, this means we have four types of clients that can use our solution. The communication amongst TIRs is as equal defined using sockets with the same mentioned protocols, although, in this communication representing the TIR network, in addition, we encapsulated the communication channels using the following strategies: **Stunnel** service as a secure tunnelled circuit (explained in subsection 3.4.4), this strategy guarantee security enforcement between TIRs. A TIR is a Java proxy that establishes communication between its client and a Tor input relay, represents the core component of the architecture (explained in subsection 3.4.2). Its goal is to handle client requests to a specific web target, however, the request must pass first through the Tor network to reach the destination targets. The TIR as the ability to handle parallel requests from the client and other TIRs, those requests are delivered direct to a Tor input relay, or bypass to another TIR and in time ending up requesting to a Tor input node, because of this feature, for each request to the Tor network, one TIR can connect to different Tor input relays, this creates the arbitrarily chosen routes. Since Tor only supports TCP packets, a TIR manages and converts different protocols into TCP protocol, then the responses arrived from Tor are converted to the original protocol and sent back to the respective client.

- The second part is the vanilla **Tor network**, responsible for delivering requests from TIRs to destination targets. We configure the Tor network as default settings with three nodes, according to the figure (input/entry node, middle node, output/exit node).

- The third part is the **destination targets**, represents any website on the internet that a client would want to access, in our case, to simplify, we built a simple Java HTTP file server that handles client file requests thought Tor and retrieves local files.

3.4.2 Components for TOR Strengthening

Here we detail the components of the TIR architecture presented in 3.4, which is the mechanism we leveraged for strengthening the Tor network.

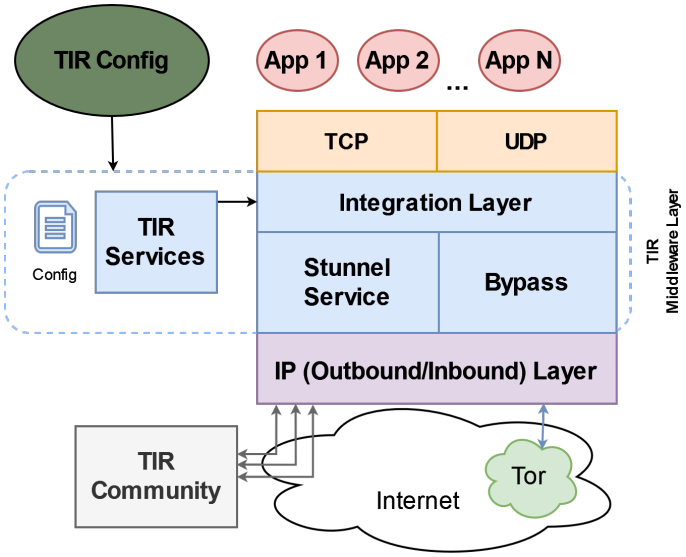


Figure 3.4: TIR architecture

- Application requests are handled by the **Integration layer**, once TIR is up and running, this component will start accepting connections on TCP and UDP ports (as well as TLS and DTLS) defined in external configuration files in TIR Config. This layer uses the Stunnel service for communication with the TIR network.

- **TIR Config** holds all files with configurations and parameters needed for the TIR setup, those include:

  - TIR services file (list of all available TIR).
  - Stunnel configuration file (required configuration file to execute Stunnel service), here we define the address to accept connections and the addresses of the other TIRs to connect to, and since Stunnel is a TLS encryption proxy, we need to define the path location of private key, certificate file and certificates of trusted certificate authorities file).
  - Keystore file, that as private key and certificates for the TLS and DTLS connections, as well as the Stunnel authentication parameters.
  - TIR service file, contains the dynamic multipath proxy parameter and the randomized time parameter for TIR selection.

- Both configurable files **Config** and **TIR Services** extract the parameterization from the external TIR Config file. Config is an internal file that defines how the TIR behaves in the TIR network, by defining a timer value to randomly select one TIR in the network to bypass client requests or to send straight to a Tor input relay. TIR Services contains the IP address of the TIRs in the network in order to establish a connection.

- **Stunnel Service** component uses the K-anonymization strategy which secures tunnelled circuits for communication with other TIRs, as explained in subsection 3.4.4, adding TLS encryption functionality between TIR nodes.

- **Bypass** component forwards or redirects TIR requests to the other TIRs without altering the packet's protocol. This component is behind the random multipath chosen routes mechanism.

- **IP (Outbound/inbound) layer** handles the requests/responses to/from the applications straight to the Tor network or to the TIR Community. The connection to the Tor network is created by a SOCKS5 proxy [55] at the same address and port as Tor service while its running, this is doable because Tor provides a socks proxy by default, by creating a socks proxy and matching socks address will allow us to use Tor service, any outgoing/incoming HTTP connection that uses a URL goes straight to Tor. This component is also responsible for packet protocol conversion if case the requests goes to Tor network, in order to correspond to the Tor acceptance protocol (in case the application packets are already in TCP protocol, the conversion is unnecessary). At last, responses from the Tor network are converted back to the original protocol and sent to the respective applications.

- **TIR Community** component is the voluntary community of TIR nodes. Each of the TIR inside this community will have the same architecture and behaviours as currently described, so in essence, any request from a TIR to another can result into a Tor request or another forward request to another TIR node.

- **Tor** component represents the vanilla Tor itself, configured with default settings. Handles incoming requests from the IP(Outbound/inbound) layer of the TIRs and delivers responses in the same communication channel.

3.4.3 Enforcement with K-Anonymized input channels

Here we describe the first strategy of our solution that is the K-anonymity input channels. K-anonymity improves Tor unicast limitations and enforces the unobservability of traffic, creating a multipath chosen routes in the input anonymization enforcement network between TIR nodes. Because of the external configuration files, every TIR knows the address of each TIR and previous connects to them before starting any requests. Once they are up and running, the different timer configured in each one will determine whether it creates a connection to a Tor input relay or pass on data to another TIR (for example, using a timer for randomizing selection of a TIR to forward data or send to a Tor input relay, when the timer ends it will start this process again). This mechanism of forwarding requests through multiple K number of TIR is the so-called K-anonymity strategy, and makes it difficult for a censor to observe and match a source IP in inbound circuits with a

destination IP in outbound circuits. K-anonymity is built on top of tunnelled channels, which is the strategy we are using to secure tunnelled TIRs circuits 3.4.4.

In a future work, we could have another strategy, by combining with the previous one, which is using media traffic-morphing (explained in 3.4.5). The solution must be generic enough to support the use of all group of collaborative K endpoint-senders, and distinctive media streaming tools as carriers for the hidden data packets passing through the covert-channels. To do, the system can manage request messages between clients and k endpoint-senders in order to create the multipath routing strategy. For the media morphing covert-channel, we could design independently of both the carrier media streaming services and the communication protocols used in the covert-channels. This allows supporting diverse media streaming services such as (Skype, Zoom, YouTube), and particular application-level covert traffic (e.g., HTTP, HTTPS, FTP, SSH), without having to change the tools used or focusing on encrypting the hidden packets.

3.4.4 K-anonymization using secure tunnelled circuits

We use secure tunnelled circuits to enhance unobservability, using Stunnel service component, which uses the Stunnel proxy to add TLS encryption in the communication between each connected TIR node. The Stunnel component builds a secure tunnelled communication between a client and a server. Since we are interested in connecting TIR nodes, they will represent both client and server, for two TIR to connect, one will act as a client and the other as a server. To enable this strategy, we start the Stunnel service in both client and server machines, which runs a configuration file name Stunnel.conf that contains all the settings needed to establish an SSL/TLS encrypted tunnel: client or server mode (differentiates who is the client and the server for SSL/TLS handshake purposes), accept address to accept incoming connections, connect the address to request connection to the server TIR, certificate file path (for the TLS encryption, every TIR certificate is imported into the TIR certificates of trusted certificates authorities) and lastly, private key file path (we may omit this option if certificate file also contains the private key). The Stunnel service architecture is detailed in section 4.2.

3.4.5 K-anonymization using media traffic-morphing

DeltaShaper could be a good candidate tool for enhancing unobservability. We could use the data encoding and decoding to morph the packet's payload in the tunnelled traffic between TIRs. With this mechanism, we guarantee privacy-preserving covert-channels enhancing better unobservability compared with the Stunnel solution, with the additional protection against protocol fingerprinting attack. The design is: On the sending side; the transmitter receives the payload and encodes it into a video stream that feeds to Skype using a virtual camera interface. Skype transmits this video to the remote Skype instance, and the received stream is captured from the Skype video buffer. The decoder then extracts the payload from the video stream and delivers it to the application. The same

approach applies to both endpoints of a Skype call, in effect, it supports a two-way channel. To make the system as simple as possible, the architecture displays data-link-level protocols at the upper layers, such as IP packets that can be received, encoded, decoded and delivered remotely using this technique. As a result, the system supports any TCP/IP application that can handle low throughput / high latency links. The problem with DeltaShaper is that nowadays it doesn't exactly implement encapsulation on top of the current Skype protocol, but on an outdated version of the Skype protocol, since Microsoft bought Skype, the protocol has become proprietary, producing an encapsulation on that protocol immediately matches an initial problem which is that one does not have access to the protocol as it exists exactly today.

To get around this incontinent, Protozoa[21] comes in place, which, as a role of a media traffic-morphing used to enhance unobservability, related to the future work improvements, being the substitute of the Stunnel service component. To encode media traffic through tunnelling, the Protozoa uses the video streams generated by client and proxy as a medium for carrying covert IP packet data in both directions. The authors use an alternative approach called *encoded media tunnelling*. Similar to existing raw media tunnelling technique [8, 51], their method replaces the carrier video information with a covert message. However, instead of replacing the pixels of the raw input video, it replaces the bits of the encoded video signal, i.e., after compressing the input video via the WebRTC video codec. This technology makes it possible to increase not only the channel's capacity but also its resistance to traffic analysis, and it's implemented by instrumenting the WebRTC stack of the Protozoa gateway, which are:

- Upstream and downstream — The WebRTC stack contains a built-in codec (VP8) that processes the video signals for local web applications that use the WebRTC API. To access the video frames generated by the WebRTC application and to implement encoded media tunnelling, the WebRTC stack has two hooks that can intercept media stream processing in different directions (for example, upstream or downstream). The upstream hook prevents frame data from going out of the local camera device into the network. This video by the video engine just before processing the raw video signal and then delivered to the frame data transport layer where the SRTP packets are generated and sent to the network. Downstream hooks intercept incoming frame data, for example, from the network to the local screen. The transport layer is placed in multiple network packets immediately after the completion of the reconstruction of the encoded frame and just before going into the video engine for decoding and rendering on the screen.

- Using EFBP (encoded frame bit stream partitions) as a covert data mule — They hooked up to the WebRTC stack to manipulate a particular data structure (EFBP), where they can embed Protozoa messages. The covert data is embedded in the carrier frame data, depicts the format of SRTP packets, which is the means through which video data is exchanged. Most of the SRTP packet space is reserved for

streaming media payloads in an area called *encoded frame bitstream* (EFB). This field contains the bits of an encoded (compressed) video frame as it is generated by VP8, the default WebRTC codec. An encoded frame comprises a small uncompressed header (3-10 bytes) and two partitions that carry a compressed bitstream that contains the actual carrier video data.

- Protozoa message transmission — Protozoa use the message format to add covert IP packets to EFBP. The same message can contain multiple layers followed by a Terminator (for example, a zero-length segment) that limits the EFBP area occupied by covert data. Each segment contains a complete IP packet or a packet fragment. The need to fragment the IP packet may be at the endpoint of the sender. The next available IP packet is larger than the available EFBP space in the frame; This process helps to use the covert channel to its maximum potential. As a result, a covert IP packet can spread to one or more segments. Each segment has a short header that allows the receiving endpoint to reassemble the IP packet fragments. The following are the functions of sending a message. Each new frame produced by the video engine has the option to send a new message, that implements an upstream hook and gives access to the encoded frame. Hook accesses the EFBP data structure, checks its size, and tells the encoder service how much available storage there is in the frame to send the covert data. Hook Encoder is waiting to send a new message containing IP packets in a queue locally. After that, the hook copies and returns it to the EFBP, so that the WebRTC pipeline will continue to operate as usual until the SRTP packets are transferred. At the receiving endpoint, the reverse operation is performed. Whenever the encoded frame rearranges, the downstream hook executes and extracts the Protozoa message that is sent from the EFBP field. This message is sent to the local decoder service, which rearranges incoming IP packets and forwards them to their proper destination.

Although there are other ways to encode and decode data, it will be interesting to compare the DeltaShaper solution with the one presented in Rui Sanches thesis [84], which has a different approach. They used three parameters to maximize the amount of data to be transmitted in a video frame, the parameters are as follows:

- In the **image parameterization** there are eight values that can be customized to achieve the best balance between the ability to embed data and the unobservability of covert-channels both at the traffic analysis level and at the visual level. An image is made up of pixels and each pixel can be represented by an amount of x bits based on several aspects, but a common approach is to use 24 bits per selected pixel. It's possible to divide a 24 bit pixel into three components according to patterns like RGB and Y'CoCg. That is, 8 bits can represent each component. Assuming a little conservative image parameterization, one could mask the data using each pixel of the frame and the maximum number of bits per pixel to maximize image

embedding capacity. an image with a width of 640 and a height of 480 becomes three smaller usable images with a width of 320 and a height of 240, for a total of 76,800 pixels. The 24 bits represent each pixel, but they only use 8 bits because they only place the data in the Y'component. Therefore, the maximum amount of data that a decomposed image can transmit is 614400 bits (76800 × 8), but because of the three decomposed images, the maximum viable integrated capacity of a video frame is 1,843,200 bits (614,400 × 3), or 230,400 bytes.

- In the **segment parameterization** only one value can be set, which is the size of segments composing a packet. This shows how a packet should be divided in case its size exceeds a segment size. So, for example, if a packet has 100 bytes and the segment size is 40 bytes, then those 100 bytes are divide between 3 segments, where the two first have both 40 bytes, but the last only has 20 bytes of actual data. In these cases, where the last segment does not match exactly the segment size present in the segment parameterization, they add padding to fill the remaining bytes.

- In the **error-correcting codes parameterization**, that is, it is only possible to define one value, which is the number of maximum errors to be corrected in a data block. This value affects the amount of actual data is embedded in an image.

3.5 Integrated solution

We conclude that the overall TIR is a censorship circumvention solution based on correlation attacks avoidance, integrated as an add-on in the Tor network, providing unobservable correlated flows, by creating the support for indistinguishability on income and outcome Tor circuits. We describe the complete flow of a request in our solution as:

1. The client request for a file targeting an HTTP file server.

2. The client's request is handled by this respective TIR, with a configured multipath routing secure tunnelled circuits to other TIR in the network (in 3.4.4).

3. A timer is running from the beginning on the background to select a new candidate TIR from the TIR network list, containing the TIR's IP address to route the data to. If the IP address matches the current TIR's IP, then the request goes directly through the Tor, if it's a different IP address the request is sent to that TIR location and the process repeats on that TIR.

4. Upon Tor receives the request, it starts the onion routing process using the 3 default relay nodes.

5. Once the process is finished, the Tor's exit relay node sends the request to the HTTP file server, and the server processes a response.

6. Finally, the response goes back again to Tor's exit relay node, passes thought to the same TIR chain and end up in the respective client.

7. In case that the HTTP file server is configured as an onion service, the last three steps will change, since we are using a hidden service, the Tor will behave differently, instead of using 3 normal relay nodes, it's used 3 introduction points nodes and a rendezvous point node to establish an end-to-end encrypted and anonymous connection. After this connection establishes, the client and the server communicate over the rendezvous point node.

8. Once the TIR got the response from Tor, the response redirects back to the original TIR that did the request (in the same circuit routing path), closing the circuit.

3.6 Summary

In this chapter we presented the system model and architecture of our solution that composed by a network of TIRs that will be add-on offering security to the Tor network, leveraging the unobservability of the traffic, hiding the identity of the source and destination IP, using the K-anonymity input circuits to create random multipath routing and the use of Tor hidden service, these two strategies guarantee client-side and server-side unobservability. For the considered adversary model, we have a mechanism for the countermeasures' correlation attacks and deep packet inspection or packet sniffing. The integration of the final solution allows the system to have a way to defend these types of passive attacks. In the next chapter, we will describe the prototype implementation of this design model, discussing the technological options and technology that we used to build the solution.

Chapter 4

Implementation and Prototype

This chapter addresses the implementation details in prototyping the TIR solution as described conceptually in chapter 3. We organized the following sections of the chapter in the following way:

- First, we introduce the main implementation issues, describing an overview of the implemented prototype (section 4.1) and then, we discuss the implementation components in the TIR architecture (section 4.2), including the technological options in this implementation.

- In section 4.3 we present the details of the system's installation and setup.

- In sequence, we explain the interconnection and adaptability support for TIR bridging with Tor relay and bridge nodes (section 4.4), focusing on the implementation option for Tor Pluggable Transport.

- Then, we explain the provided TIR proxy-facility for transparent support of end-user applications (in section 4.5).

- The next sections dedicate to other complementary implementation aspects promoted by the current TIR prototype, such as flexibility and openness for traffic shaping facilities and optimizations related to TIR and Tor bridging facilities (in section 4.6).

- At last, we described the implementation details of the TIR prototype developed in our dissertation work in (section 4.7).

4.1 Implementation of the proposed solution

We have implemented a K-anonymization multipath routing circuits built on top of secure tunnels (shown in Figure 4.1), we developed it on Linux and other operating systems. The prototype has many components that apply to the clients using TIR and HTTP file server pipelines (see Figure 3.1b). Most of the components (Clients, TIR and HTTP file Server) were created from scratch in Java; others based on existing tools. The prototype is available for research purposes in (https://github.com/joaoteixeira96/Thesis). It can be used as a base platform for the study of censorship circumvention techniques providing censorship resistance and guarantee unobservability of the input circuits, also, for possible expansion in future work research directions.

4.2 Design and implementation options and technology

4.2.1 Technology

We designed the entire system to be more general and efficient in order to overcome the limitations of identifying the relevant solutions described in chapter 3. Compared to other existing research on managing multipath routing circumvention strategies and the Tor solution for hidden services, the implemented prototype requires a more sophisticated development approach rather than using some already implemented modules. In the implemented prototype, there are three major components:

- Related to the core of our proposed TIR solution, there we have a reconfigurable multipath routing environment of entry circuits.
- Related to the most important external component, the Tor network, we have two:
  - The vanilla Tor solution using three relay nodes.
  - The Tor hidden service solution.

The figure 4.1 extends the architecture shown in figure 3.4 in chapter 3. The prototype required a set of technologies that have been selected, and they are:

- **Client endpoint**: Contains the client application, which is a small program that reads system input file path and the protocol type desired for communication, building a Java socket TCP or UDP connecting with IP address A.B.C.D with port 1234 or a Java secure socket TLS or DTLS connecting with IP address A.B.C.D with port 2000. We can have the same port for both TCP and UDP since each request is identified by a quintuple contained by source IP, destination IP, source port, destination port and protocol (as a protocol can be TCP or UDP protocol), the same condition for TLS and DTLS which derives from TCP and UDP. The client application reads two files: keystore (holds the client private key and TIR certificate for the TLS and DTLS protocol) and network config (holds the IP address, unsecure and secure

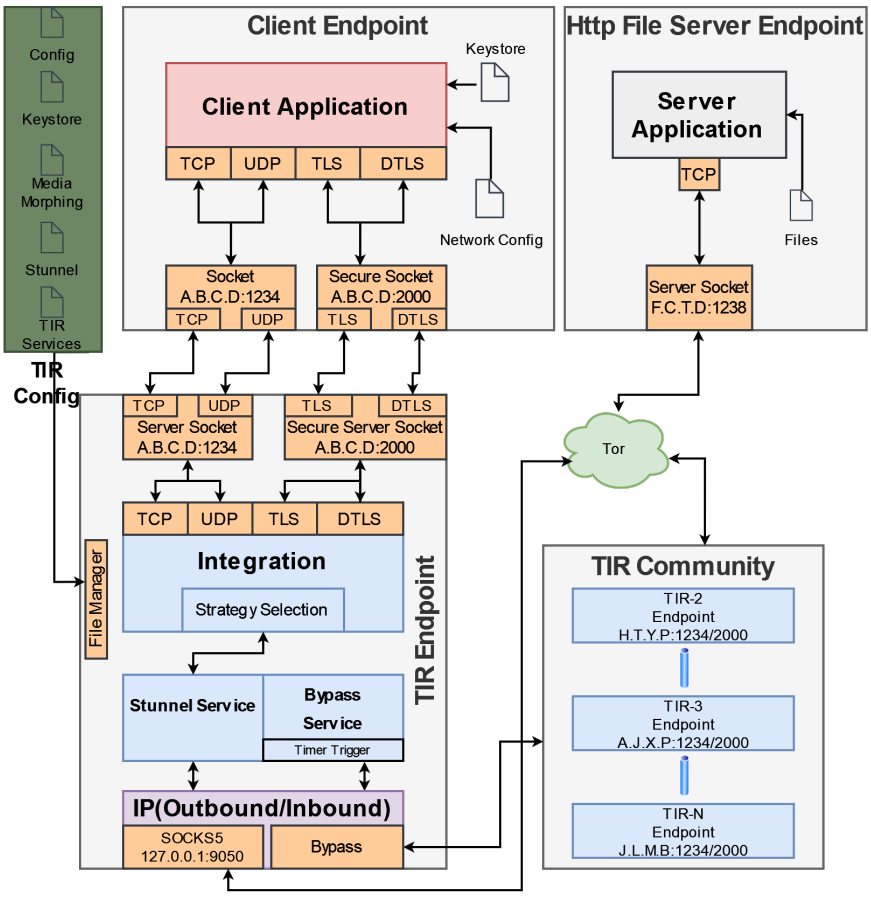


Figure 4.1: Design of prototype solution

port of the TIR). At last, it is important to know that every client endpoint only communicates with his respective TIR endpoint.

- **TIR Config**: Component that holds five configuration files for the TIR. Config file contains the internal configurations such as local host IP address, local unsecure and secure port, HTTP file server IP address and port, SOCKS5 proxy IP Address and port for Tor service, strategy type (for now it is only one strategy to select) and last, the bypass timer.

- **TIR endpoint**: The core TIR solution, responsible for using the multipath routing mechanism with K-anonymization of used circuits, which builds on top of tunnelled channels as secure tunnelled circuits. After running TIR starts by reading the TIR config using a file manager. After that, the **integration** component launches four threads, one for each type of protocol, using two Java server sockets (TCP and UDP) and two Java secure server socket (TLS and DTLS) to accept connections on the designated ports. In the figure 4.1 we can see that the IP address of the TIR sockets and the client sockets are equal, it is because they are running on the same machine and using the same Internet service provider. There is a possibility that we could deploy the TIR on a remote machine, but we are not interested in that case, because the

result would be the same with the penalty of higher latency, degrading the overall performance. The integration component selects the communication strategy, in our only case is the **Stunnel** Service, but in future work, there will be another strategy to select (such as the media-morphed covert channel), this strategy selection depends on the settings read from the config file. At last, the **IP(Outbound/Inbound)** component as the purpose of communicating to the outside network, establishing a connection for file requests with the final web target, the **HTTP File Server Endpoint**, redirecting the traffic through the Tor network or redirects requests to the other TIR nodes inside the **TIR Community**. For the first case, the IP(Outbound/Inbound) component runs a SOCKS5 proxy on IP Address 127.0.0.1 and port 9050, by default, Tor listens for SOCKS local connections on port 9050, meaning that Tor captures requests on this port and executes them. For the second case, this component uses the Bypass Service to select the chosen TIR node that will handle the file request. The **Bypass Service** loads the settings of TIR Services and Config files, extracting the TIR endpoint locations from the Community and the timer number, stores the nodes into a local data structure and the timer into a local constant. Whenever the timer reaches zero, a trigger is sent to this service to select a new TIR candidate to redirect file requests. There is an option that, if the selected TIR is itself, then it is responsible to create the request to the HTTP File Server, else another TIR node is selected to bypass the request. The **Stunnel** Services is explained in the figure 4.2 below.

- **TIR Community**: Represents the voluntary community of a set of TIR nodes distributed worldwide. Each TIR is configured according to their client settings, for example, TIR-2 endpoint is running on a remote machine using a public IP address H.T.Y.P.. The presented ports 1234 and 2000 corresponds to the unsecure (for TCP and UDP) and secure (for TLS and DTLS) ports that the client defined in the configuration files. As said before, each TIR-(2...N) has a unique client aggregated with a defined configuration files for the client and TIR.

- **Tor**: The vanilla Tor network, installed and configured with the default settings in each TIR machine. Its role is to accept SOCKS connections on listen port 9050 and executes the onion routing technique, passing the request through a virtual circuit consisting of three random-selected Tor relays and at the end, reaches to the HTTP File Server. Tor's application independence sets it apart from most other anonymity networks: it works at the Transmission Control Protocol (TCP) stream level. In case we set the HTTP File Server to be an onion service, Tor will behave different according to the hidden service Tor routine already explained in 2.2.3.

- **HTTP File Server Endpoint**: A file server running on a remote machine that operates a file system in the **Server Application**, reading from a directory named Files that contains a list of all type of file format. We built the server to simulate a typical

web browser where we can access any type of files. We only use TCP protocol, in our case, with the purpose of accepting Tor connections via Java server socket.

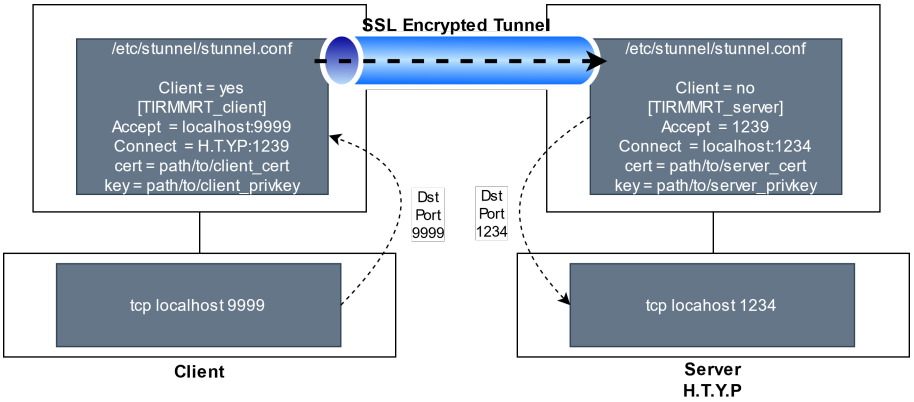


Figure 4.2: Stunnel service

The figure 4.2 represents the Stunnel service component in the figure 4.1, containing all the configuration needed to use the Stunnel to establish a secure tunnelled connection between two or more TIR. One TIR node must act as a client and the other as a server, both are running the Stunnel service on their machines. Starting by, the client application requests a TCP connection to his Stunnel service, accepting on port 9999. Stunnel loads the certificate and private key and requests the handshake with the Stunnel service of the server in IP address H.T.Y.P and port 1239. The server accepts the incoming request on port 1239 and validates the certificate. The handshake completes and creates an SSL encrypted tunnel. At last, the Stunnel service of the server side connects to the TIR application on TCP port 1234. To better understand the request flow, once the TIR node gets a bypass requests with Stunnel strategy enabled, the TIR (client) application requests a connection via Stunnel to the TIR in IP address H.T.Y.P port 1234 by sending a TCP request to local Stunnel, forwarding the requests through the SSL encrypted tunnel and reaching the TIR (server) Stunnel service, at last, the Stunnel service sends a TCP request to the local TIR (server) application.

4.2.2 System prototype

We developed the TIR prototype, a component software, by writing about 4,200 lines of Java and Python code, including the client application and HTTP file server. We built the system on Ubuntu 20.04 using: Java v11, Tor v0.4.4.6 [71], Stunnel v5.57 [90] and Python v2. Furthermore, we imported some extensions, using Maven v3.6.3, to help create the Java SOCKS proxy for Tor usability with Java program. We could use Java Secure Socket Extension (JSSE) to create the Java socket for DTLS communication protocols, since it was added in Java SE v9.

4.3 Prototype installation and setup

To install the prototype, there are some system requirements that must be fulfilled. The machine for this experiment must support: Java v9+, Stunnel v5+ and Tor v0.4.4.6+. First the executable Java JARs must be downloaded from a link at 4.1, which holds the code of the client, TIR and HTTP File Server. The installation of each software can be local or remote (noting that the client and is respective TIR should be installed on the same machine). In our experiment, we used one client and TIR installed in a local physical machine and two other TIRs installed remotely in different Virtual Private Servers (VPS) regions. At last, the HTTP File Server was installed in one of the VPS machines. The most important part is that we need to simulate a wide community of distributed TIR nodes. After the installation, we need to set up:

- **Client**: Change the IP address and port of its configuration file, according to the web target server we pretend to request files from.

- **TIR**: We need to set two files: TIR Services and Config. In the first one, we add per line, the location addresses of all the TIR community (TCP IP address and port) we wish to communicate, followed by the address on the local physical machine. In the second file, since we are running in localhost, we just need to change: the remote address and port for the web target location, Stunnel port according to the acceptance port in the configuration below, the strategy type we intend to select and at last, the bypass timer value.

- **Stunnel**: For the TIR which acts as a client (as we see in figure 4.2), we need to change the connect property, set the IP address and port of the TIR we wish to communicate to, similar to the TIR Services file. Also need to change to Client property to yes. In case of multiple connections, we to add the correspondent amount in connect properties. In case the TIR acts as a server (accepting other TIR connections), we set the Client property to no, the accept port must be the same as the connecting port of the TIR client in order to establish a connection and the connect property to the address that the TIR server is accepting TCP connections. Finally, for both client and server, the cert and key property must change according to the certificate and private key path in Keystore folder.

4.4 TIR bridging and Tor transparency

In this section, we will cover how the TIR nodes transmits bridging to the Tor transparently and how it uses the Tor transport pluggable protocol without any changes in the Tor nodes.

4.4.1 TIR Tor client bridge component

To understand how TIR dialogue with a Tor bridge, we know that the TIR nodes send traffic to Tor and become Tor networks clients, attempting to connect to an input circuit in a Tor input bridge. For this to happen, TIR needs to have knowledge of one Tor entry node, this discovery process is the same as a normal user who will discover and connect to an entry Tor node. It's important to know that, distinct TIR nodes can bind to different Tor input nodes, making them independent of each other. This opens up the possibility that, even if a censor had to produce intrinsic correlations within the Tor network itself, could no longer do because of this feature.

4.4.2 Supported Tor pluggable transport

In our prototype, we use the supported Tor pluggable transport (PT) obfs4 [67], it's current the most effective transport to bypass censorship compared to other currently deployed PT: doc/meek [28], Format-Transforming Encryption (FTE) [30] and ScrambleSuit [86], including the ones that aren't much used: SnowFlake, basket2, SkypeMorph, bananaphone, LODP, sshproxy, hexchat, Dust2, Code Talker Tunnel (previously SkypeMorph), git, Castle, marionette, lampshade and TapDance. There are deprecated PTs which are not used any more, most of the time because a better version has come out, these are Dust and StegoTorus. Also, there are undeployed PTs: obfs3, obfs2 and Flashproxy.

To enable the pluggable transport, we need to install obfsproxy package, and add the following lines to the Tor configuration file (torrc):

```
UseBridges 1
Bridge obfs4 IP:Port
ClientTransportPlugin obfs4 exec /usr/bin/obfsproxy
```

UseBridges defines Tor communication using bridges. The *Bridge* defines the bridge to use and the pluggable transport, and the *ClientTransportPlugin* defines Tor a name and a path to an executable to act as the pluggable transport.

4.5 Interaction between TIR and user application

A transparent mechanism establishes the link between TIR protection and user application. TCP, TLS, HTTP, HTTPS, UDP, and DTLS are among the protocols enabled by the prototype, and one node of any TIR is used as a proxy for them. We establish communication by instructing the client sending the traffic to use the TIR nodes as transparent proxies in the same way that they would any other HTTP/HTTPS proxy. We can see the

TIR protection as a "browser plugin" which adds extra security, and the user needs to install and manual configure to his personal options and selecting the voluntary community network of other TIR nodes.

4.6 Openness, flexibility, modularity and extensibility issues

It is important to notice that, machines in a private network have conditions of routing through the infrastructure of the internet service provider and the dynamic host configuration protocol (DHCP) assigns the home network address by the provider. In a scenario, where we use some TIR on private networks, to use a TIR with this private address, we needed to have to do network address translation (NAT) firewall locally in order to forward packets to the exterior (either to other TIR running on a VPS or any other location machine). It is not possible to have configurations in which the other TIRs deliver traffic to the TIR of the local machine in the private zone. For that, the TIR will have to run in a public domain, so the volunteers running the TIR would have to have a machine in the cloud with a TIR instance.

4.7 Summary

This chapter has described the implementation details of the TIR prototype developed in our dissertation work. The prototype is currently available (in an open-source project) and it is ready to be used and showed for research purposes. The TIR prototype can be well-used as a base platform for possible extensions that could be considered as future enhancements, by the interested research community.

We emphasize that the role of TIR is to provide a distributed peer-group K-anonymization environment targeted as a pre-stage anonymization ad hoc internetworking environment to improve anonymized, and indistinguishable channels leveraged by the Tor network. TIR wasn't implemented as a plugin or a new implementation of Tor nodes, to enhance the anonymization mechanisms internally used today by Tor bridges, onion-routing relays or rendezvous nodes.

We also mention that there are other requirements to withstand a more powerful attacker on our design considerations. TIR already employs setup tools for the use of the system to synchronize actions among clients and Tor bridges, considering the distribution of ad hoc collaborative TIR nodes running in different AS routing regions (what we can call anonymization-out approach). However, other concerns to deal with the dynamic ad hoc organization of TIR nodes or other mechanisms that can be designed as counter-measures against disruptions and performance adaptations against possible active attacks causing network perturbations (as a specific anonymization-in mechanism) can be addressed in the future. To address these challenges, TIR already implemented mechanisms that could be optimized in the current prototype to provide a more dynamic

and flexible configuration environment to deal with network perturbations that censors can conduct.

The chapter ends by addressing other complementary challenges, hardening other current TIR limitations, in addressing client-side and Tor bridge interoperability with better trustworthiness conditions. For this purpose, we discuss how our current implementation options offer a good baseline to address the challenge to support TIR in a trusted execution environment. In this direction, we could run TIR as a hardware-shielded and trusted solution, withstanding against a local attacker capable of inspecting or intruding TIR, when it runs in vulnerable memory space.

At last, using the prototype presented in this chapter, we conducted several experimental observations for validation that we present in the next chapter 5.

Chapter 5

Validation and Experimental Evaluation

In this chapter, we present the experimental evaluations and related observations conducted for the validation of our TIR proposed solution. We carried the experimental observations out with the prototype explained before in the previous chapter.

We start by explaining our methodology and the test bench environment used for the experimental tests conducted for the TIR evaluation purpose (section 5.1). Then, we study and analyse the operation and performance of TIR, when using the tunnelled K-Anonymity solution protecting Tor input circuits, under different settings, to analyse latency, throughput, and overall performance impact (section 5.2). We provide some reference indicators of Tor network usage and operation in (section 5.3) to compare the vanilla Tor performance against our solution (section 5.4). In sequence, we analyse the effectiveness of the proposed solution in establishing the desired indistinguishability criteria, as provided by the TIR K-anonymous circuits (section 5.5), even in the presence of robust against active network correlation attacks. Next, we present a comprehensive study respecting the TIR communication and computation resources, such as network, memory requirements and CPU usage (section 5.6). Last, we conclude the chapter by summarizing and validating our current available prototype (section 5.7).

5.1 Methodology and test bench environments

To present our evaluation methodology, we describe our assessment goals and approach to analyse two main metrics: performance (including latency and throughput observations) and traffic unobservability (including the analysis of the indistinguishability of traffic flowing through the K-anonymization structure provided by different TIR instances). The objectives are: (1) to assess the performance merit indications of TIR and (2) to measure the resistance when using different TIR instances against traffic analysis correlation

attacks. For each one of the intended analysis, we prepared the required test bench environments and an evaluation methodology.

5.1.1 Evaluation goals

We structure the evaluation goals into three major goals:

1. To assess the performance of the TIR prototype; For this case, we selected as primary observation targets the latency, reliability and throughput conditions, comparing to the direct use of Tor, as currently available for Tor clients.

2. To test TIR resistance against multiple traffic analysis attacks; For this case, we focused the analysis in assessing whether it is possible:

   - To identify flows between interoperable clients (input traffic and peer-to-peer circuits between clients) and in Tor input nodes, to test if those flows can be distinguished, comparing K-anonymized input data versus those that only carry chaff traffic.
   - If there is a possibility for correlation of useful data flows produced by protected clients with the outbound flows exiting Tor to final targeted systems.

3. To observe the impact in communication and computation resources supporting TIR; These observations are based on the use of typical instrumentation tools provided for Linux-OS monitoring tools, to capture the allocation of communication and computation resources, when we run TIR in Linux-based client nodes.

5.1.2 Methodology for performance evaluations

To conduct our performance evaluations, we focused our analysis on throughput and latency measurements using TIR. Our observations include a comparative analysis involving TIR and the direct use of Tor, regarding different traffic classes (ranging from IP processing performance and HTTP measurements) and different packet chunk sizes, including 512 bytes, 1024 bytes, 2048 bytes and 4096 bytes. These sizes are relevant in the sense of implications on Tor internal onion-routing chunks containing up to 1, 2, 4 and 8 Tor cells. These observations are also relevant for the analysis of possible traffic shaping conditions as a rationale to maintain minimum and maximum throughput compared to vanilla observations of the Tor internetwork performance (when not using TIR). For example, if we consider a rate function for traffic debit as rate $(d) = 10 \times t \times n$ µs, where $\boldsymbol{t}$ is the number of maximum Tor cells that fit one chunk and $\boldsymbol{n}$ is the number of connected clients, we will get a related observed theoretical throughput (about 8.5 Mbps) for 50 connected clients, regardless the frame chunk size for the given results in a test bench for a typical Tor end-user. However, the same effect observed from different TIR instances in different internet locations together with the variation of the number of used

TIR instances can vary the observation. These measurements are useful to understand the implications of possible optimal parameterizations of those TIR instances, in their interconnection with the Tor relay or bridge input points.

5.1.3 Methodology for unobservability assessment

Regarding the resistance of the TIR solution against traffic analysis attacks, the goal is to assess whether it is possible to identify which flows between protected end-clients and Tor bridges (income and outcome circuits) transport hidden useful data, versus those that only carry chaff (or placebo) traffic. In the last case, we must analyse the effect when we use a different number of distributed TIR instances interconnected to Tor network input bridges and how the K-anonymization effect relates to the unobservability protection.

In the recent literature aimed at distinguishing encrypted traffic flows (including the related work on traffic correlation factors and its mitigation in anonymization networks, such as the Tor), the main target of the approach was primarily focused on the analysis of packet length and inter-packet arrival times. Pre-existent correlation approaches employ classifiers based on machine-learning observations, such as those using decision-trees (e.g., Random Forests [38] or XGBoost [9]). Similarly, as we find in earlier traffic analysis over encrypted network streams, we leveraged the XGBoost classifier to study and assess indistinguishability properties. Then, for this classifier, we leverage features based on multiple statistics over the packet length and inter-arrival times, such as burst behaviours and high-order statistics, as well as, in observing the correlations when using different TIR instances. To manage the observations, we train and test the classifier through 10-fold stratified cross-validation. We conducted our observations by obtaining the following metrics with XGBoost tools: *true positive rate* (TPR), *false positive rate* (FPR), and the *area under the ROC curve* (known as AUC metrics).

To analyse which flows between end-clients, TIR instances and Tor income bridges and outcome relay nodes, we considered pairs of traffic sample sets, transporting useful data versus placebo or chaff traffic data. The evaluation of TPR metrics in analysing such pairs gives us the fraction of observed flows that are correctly identified as carrying useful data, while the FPR metrics correspond to the fraction of placebo traffic flows, erroneously classified with XGBoost as apparently useful traffic. To observe the correlation between useful data flows produced by clients with the outbound flows from TIR instances bridging with the Tor input bridges, the TPR metrics allow us to analyse the fraction of client-bridge and TIR bridge-Tor pairs that are correctly correlated, whereas the FPR metrics measures the fraction of uncorrelated pairs that are deemed as positively correlated. The Receiver Operating Characteristic (ROC) curve plots the TPR against the FPR for the different interception points, helping in the study of optimal thresholds that can be adjusted or optimized for the AUC trade-offs, starting from the rationale that an AUC optimal value of 0,5 corresponds to random guessing in classifying traffic as chaff or useful, from a censor perspective. The analysis of the above metrics when different

located TIR instances are used allows for the study and validation of the solution for the desired effect.

5.2 Performance evaluation test bench environments

To address the previously presented methodology, we created a set of test bench environments, as we explain in the following subsections.

5.2.1 Vanilla Tor test bench

The aim of the test bench for vanilla Tor involves having only the use of traffic that carry chaff data that flow from only one client and his TIR, through the Tor network and to the final HTTP file server. The measurements and unobservability results from this environment are used to compare with the test bench environment of our solution, described below, in order to see the differences from the normal usage of Tor operation.

5.2.2 TIR test bench environment and infrastructure

The test bench environment is represented in the following figure 5.1, describing the structure of the test bench environment of our planned solution, following a test bed environment inspired by the system model initially explained in Chapter 3 (section 3.2).

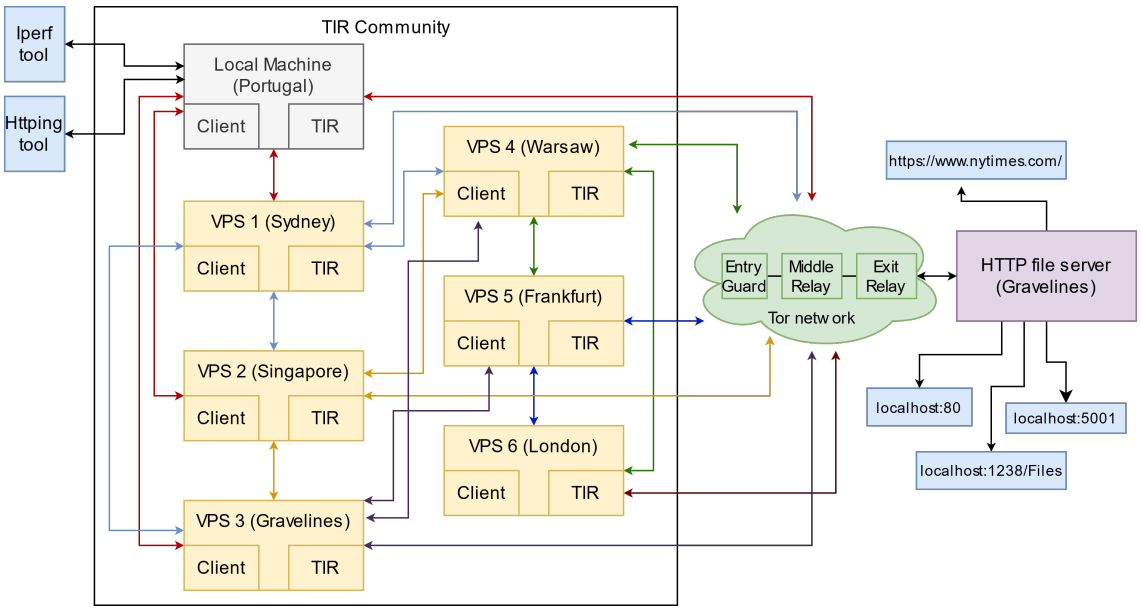


Figure 5.1: test bench environment.

To test the performance metrics (throughput and latency conditions) we set the TIR Community, composed of seven machines (one local machine and the others are VPS distributed worldwide), each of them with Tor network access to request the final web server, the HTTP file server. Tor network is accessible for inbound traffic through input

Tor bridges (entry guard) and outbound Tor traffic via Tor exit relay node to the server. We have set the TIR connection to a maximum of three TIRs (since we have in total seven TIR, and we want to avoid request repetition), as we see on the figure, with the same coloured arrows per TIR connection, the different connections between TIRs, for example, the local machine can establish a connection with VPS 1, VPS 2 and VPS 3 (presented in red arrows), VPS 1 connects with VPS 2, VPS 3 (presented in blue) and so on.

To measure the throughput, we used the Iperf tool that runs the client mode on the local machine and the server mode on the VPS which has the HTTP file server, and to measure the latency we only need to run the Httping tool on the local machine. The configurations for these tools are:

- The throughput test executes on the local machine using, with client mode, a script to simulate 20 serial clients, with request content of 50 KB, the different buffer write sizes and 10 seconds interval between requests. For the server mode, we used different buffer read sizes.

- The latency test executes on the local machine, using 20 serial clients, with different buffer write/read sizes and 10 seconds interval between requests.

5.2.2.1 Implementation of TIR nodes

To understand whether our solution could generate unobservable traffic and ensure a good throughput metrics, we ran all of our tests on eight different machines:

- **Local** machine running a 64 bits Linux Ubuntu 20.10, Intel Hexa-core i7-8750H CPU 2.20GHz Processor, 16 GB of RAM and 100 Mbps Bandwidth.

- **OVHcloud VPS** machine hosting OVH Datacenter, running a 64 bits Linux Ubuntu 20.04, vCore 4 (VPS, w/ dedicated 4-Cores) Processor, 8 GB of RAM, NVMe SSD 160 GB Storage and 1 Gbps Bandwidth.

- **Six OVHcloud VPS** machine hosting OVH Datacenter, running a 64 bits Linux Ubuntu 20.04, vCore 1 (VPS, w/ dedicated 1-Cores) Processor, 2 GB of RAM, NVMe SSD 40 GB Storage and 250 Mbps Bandwidth.

To facilitate our tests with more flexibility, TIR nodes executes within Docker containers, provisioned either with 2 vCPU and 2 GB of RAM, or 1 vCPU and 1 GB RAM.

5.2.2.2 Distribution and location of TIR instances

Our test bench considers the use of clients running from laptops (usually running on domestic wireless local area networks) using private IP addressing and network address translation supported by the typical domestic wireless routers. To conduct our experiments, we located test clients (running TIR) in remote virtual machines, emulating the

use of other clients in remote domestic networks, these instances are in Sydney, Singapore, Gravelines, Warsaw, Frankfurt, and London (shown in figure 5.1). Finally, the traffic from the K-anonymization circuits supported in the P2P client-environment is redirected from the client-based TIR components to the real Tor network, using different selected Tor input nodes. The traffic sent in this inter-networked environment (namely, HTTP, HTTPS/TLS, as well as, IP and ICMP traffic) is sent to final targets in a cloud-enabled virtual machine. With this strategy, we can capture and observe traffic flows in different segments: initial client, in the K-anonymized circuits and TIR outbound traffic for Tor input nodes, and traffic appearing as inbound traffic for final targets, allowing us to observe in each experiment, the intended correlation observations.

We observe that the test bench environment allows us to run multiple instances of clients (as multiple TIR instances running in a final client machine and in multiple client instances running in Cloud VMs), in physically separated nodes. Client-based TIR instances interact with the Tor network via the Tor bridges in real input Tor nodes (namely entry and exit nodes).

5.2.3 Experimental analysis and observations

5.2.3.1 Vanilla observations

In the subsections below, we explain what are the evaluation criteria and observation metrics that we are going to consider. Local and cloud side client benchmarking and the testing targets, in order to, in the next section 5.3, retrieve some baseline indicators of Tor vanilla observations from the Tor metrics website [91], where is stored all the users, servers, traffic, and performance information and compare those with our solution metrics. These results will determine how much our solution impacts the overall performance against the vanilla Tor.

5.2.3.2 TIR evaluation criteria and observations

We will discuss the principal objectives of evaluation criteria and observations of the TIR operation, retrieving the fundamental performance metrics, such as throughput, latency, bandwidth, jitter, and traceroute. For evaluation purposes, the TIR installed in every machine as the same configurations, using a packet size of 1024 bytes. It is important that the packet size is constant, as we explain later, we can get more accurate unobservability results. Every machine as the required software for the normal usage of the TIR community, as explained in section of the previous chapter 4.3.

5.2.3.3 Local client benchmarking

For the local client benchmarking, we set up on the local machine one client and TIR instance, as shown grey in figure 5.1. Once we set the other global VPS machines to run,

the local machine uses the respective tool to measure the performance of the requests that are sent through the other TIRs to the end machine (HTTP file server).

5.2.3.4 Cloud-side benchmarking

In the cloud-side benchmarking, we set every VPS machine with one client and TIR instance, as shown in yellow boxes in figure 5.1. We configured the network community of each TIR from 1 to 4 TIRs. Within this range, we can simulate a scenario where a TIR sends requests directly through Tor, by using only 1 TIR (himself) or uses up to 4 different volunteered TIRs to do the multi-path routing communication strategy.

5.2.3.5 Testing targets

In the figure 5.1, the HTTP file server fetches file content from local files in the server directory (localhost:1238/Files/[Name]), in order to response the client file requests, which is the normal operations of our environment. Additionally, the server is used for performance measurements, having a latency test that accesses the local server environment (localhost:80) endpoint, and to the content of the HTTPS web server of The New York Times (https://www.nytimes.com/), as well for throughput measurements, running a local Iperf server instance on (localhost:5001) using the command (iperf -s).

5.3 Reference indicators of Tor network usage and operation

In this section, we present a set of basic indicators extracted from Tor user metrics [91], summarizing the principal figures around the use of the Tor network usage and operations. To better compare against our results, we took these metrics in the same timeline (day 22 to 30 Mar 2021) we tested the performance of our system.

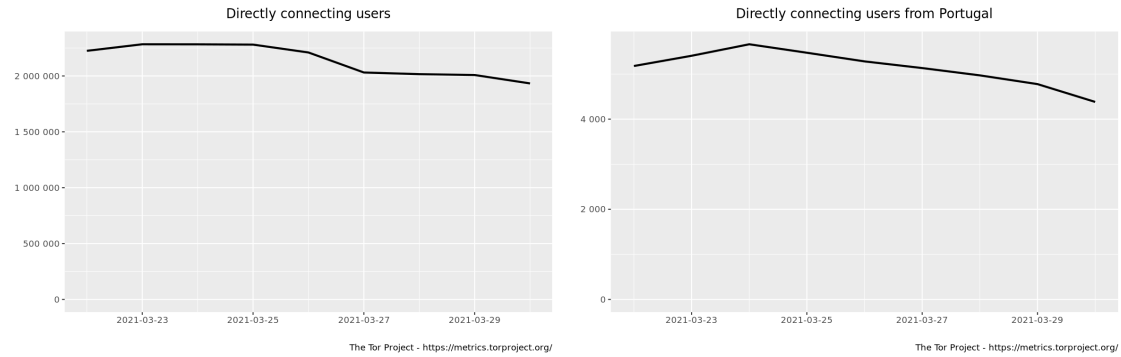
5.3.1 Tor Clients

There are approximately 2,250,000 directly connected global clients (fig. 5.2a) to the Tor network where 5000 clients are from Portugal (fig. 5.2b), it excludes clients connecting via bridges. These estimates are derived from the number of directory requests counted on directory authorities and mirrors.

In these figures, we can observe certain stability in the number of users throughout the timeline, it's important to have this consistency because Tor network reliability and performance depends on the number of connected users, and it will affect the experimental analysis in 5.3.13 and 5.4.1.

5.3.2 Tor bridge users

The estimated number of bridge global clients connecting via bridges is 40,000 (fig. 5.3a and 80 of those from Portugal (fig. 5.3b. These numbers are derived from directory



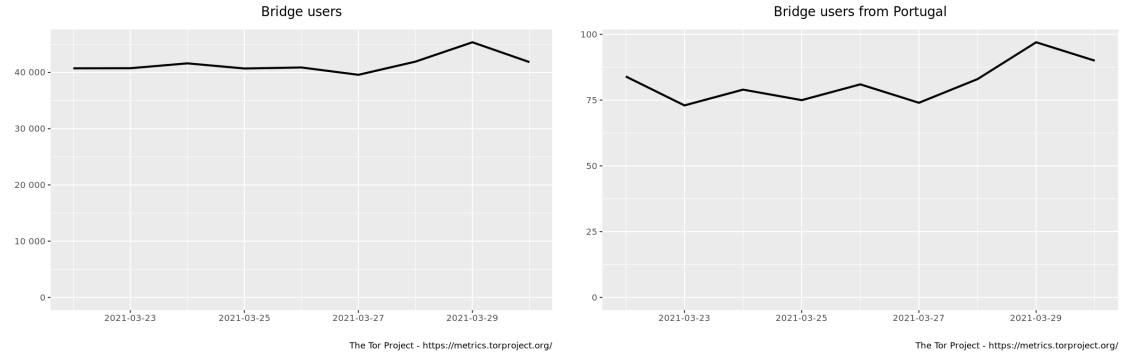
(a) Global users. (b) Portugal users.

Figure 5.2: Global and Portugal users directly connected to Tor (taken from [91]).

requests counted on bridges.

The presented figures show a certain stability in the number of global bridge users using the Tor bridge throughout the timeline, there is a bit of increase deviation (75 to almost 100) in the number of Portugal bridge users. There could be two reasons for this, one is that the number of users depends on the Tor bridge availability, meaning that the unblocked Tor bridge servers have increased, or simply more Tor bridge users were online on that period.

Tor bridge users have a significant impact in our experiment evaluation. We could have a situation where the number of users was much higher, proportionally to the number of Tor bridges, meaning that Tor needed to provide bridges from different regions to connect to, affecting the prototype's performance, or in the worst case, couldn't have available bridges.



(a) Global users. (b) Portugal users.

Figure 5.3: Global and Portugal bridge users (taken from [91]).

5.3.3 Number of running relays

The global number of running relays and bridges in the Tor network is 6300 and 1800 (fig. 5.4).

The number of global relays is one of the most important parameters to consider during our experimental evaluation. Since our test's baseline relies on the number of active relays, the more relays we have at our disposal, the more possible ways Tor nodes has to reach the intended destination server and more Tor node combinations are available to circumvent censorship. The same logic applies to bridges.

The discrepancy between relay and bridge nodes is because of the usage, different technical requirements and legal implications. Relay nodes can be entry or guard, middle and exit relay. Entry and middle relays rarely receive abuse complaints, but requires minimal maintenance efforts and bandwidth usage. Exit relays have the greatest legal exposure and liability of all the relays. Tor bridges are nodes in the network that are not listed in the public Tor directory, which makes it harder for ISPs and governments to block them. Basically, they are unpublished relays, users that are behind censored networks can use bridges to access the Tor network. The number of bridges is lower than the relays for two reasons, they require additional configuration and more computational effort [59] since they have an extra layer of security and there are more uncensored than censored regions worldwide, so there is less need to circumvent the authorities.

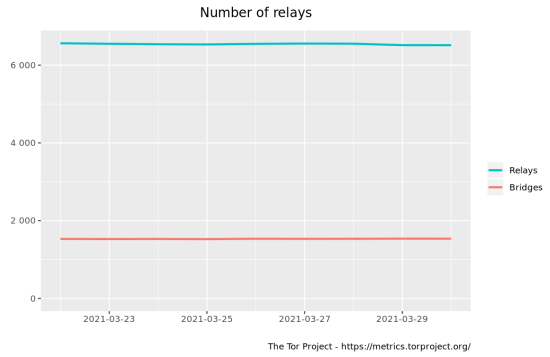
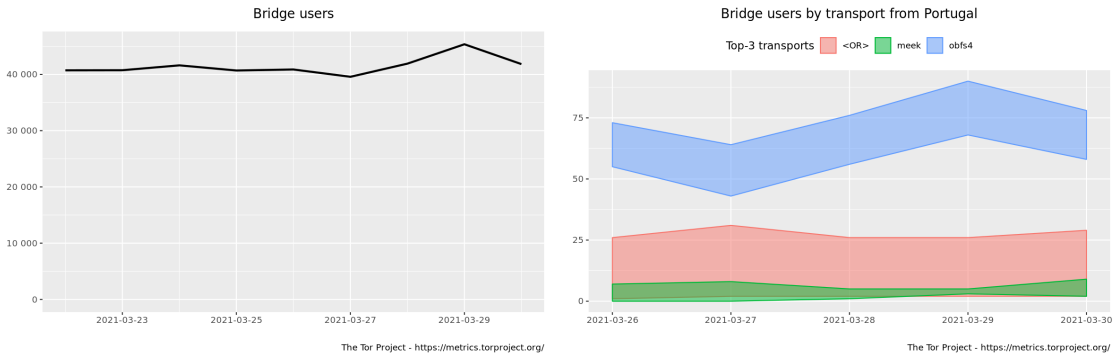


Figure 5.4: Number of relays (taken from [91]).

5.3.4 Bridge clients by transport support

The graph 5.5a shows the estimated number of global clients connecting via bridges is 40,000 using any pluggable transport. Bridges resolve client IP addresses of incoming directory requests to country codes, and they distinguish connecting clients by the transport protocol, which may include pluggable transports. Even though bridges don't report a combination of clients by country and transport, it's possible to derive and graph lower and upper bounds from existing usage statistics. In fig. 5.5b we have the estimation of bridges users by transport from Portugal using the top 3 transports: default OR, meek and obfs4 protocol. As we previously discuss, obfs4 is the most popular, as well as in Portugal.



(a) Global users. (b) Portugal users.

Figure 5.5: Global and Portugal bridges users by transport support (taken from [91]).

5.3.5 Bridge clients per IP addressing

This graph 5.6 shows the estimated number of global clients connecting via bridges. Bridges distinguish connecting clients by IP version, so that graphs are available for both IP versions 4 and 6. The approximate estimation of bridge users using IPv4 is 40,000 and IPv6 is 35. A bridge can support IPv4 or IPv6 by announcing an IPv4 or IPv6 address and port for the OR protocol. In our experimental evaluation, we are using IPv4, we must notice that, because the difference of IPv4 users is much higher than IPv6 users, it indicates that the number of IPv6 bridge nodes are much lower than IPv4 bridges [87]. If we used client for version IPv6, it could significantly influence our performance analysis into worse results.

(a) IPv4 users. (b) IPv6 users.

Figure 5.6: Bridges users using IPv4 and IPV6 addressing (taken from [91]).

5.3.6 Tor IP addressing observations

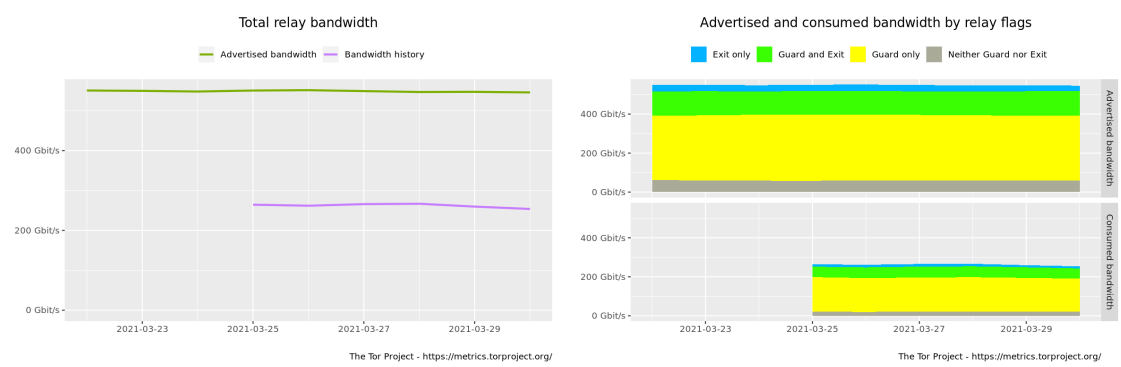
Since there is a different number of Tor nodes running IPv4 and IPv6, there is a connection limitation to consider when TIR nodes use one of these versions, because there is a gap between them is approximate 39,965.

5.3.7 Total aggregated bandwidth estimation

The graph 5.7a shows the total advertised and consumed bandwidth of all relays in the network. The estimated advertised bandwidth is approximately 550 Gbits/s and consumed bandwidth of 280 Gbits/s. We conclude that, the bandwidth usage by other users wouldn't affect the prototype performance measurements, since with this observation, we can see that there is a gap of 270 Gbits/s between advertised and consumed bandwidth, meaning that almost half of the total aggregated bandwidth needed to be consumed to reach the advertised limit.

5.3.8 Measured consumed bandwidth by Tor relay types

The graph 5.7b shows advertised and consumed bandwidth of relays with "Exit" and/or "Guard" flags assigned by the directory authorities. It's interesting to see that the advertised and consumed bandwidth of "Exit only" is equals to the total advertised and consumed bandwidth of all relays shown in 5.7a. Tor measures the bandwidth from the client request until it arrives at the final server, being the Exit node the last to deliver the request.



(a) Bandwidth estimation. (b) Consumed bandwidth by Tor relay types.

Figure 5.7: Consumed Tor relay Bandwidth (taken from [91]).

5.3.9 Advertised bandwidth distribution

The graph 5.8 shows the distribution of the advertised bandwidth of relays in the network. Each percentile represents the advertised bandwidth that a given percentage of relays does not exceed (and that in turn the remaining relays either match or exceed). For example, 75% of relays advertise at most the bandwidth value shown in the 75th percentile line (and the remaining 25% advertise at least that amount). In the graph it's included the maximum, third quartile, median, first quartile and the minimum.

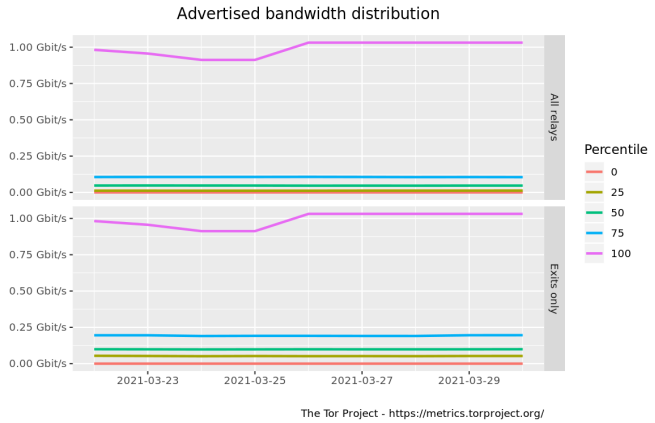


Figure 5.8: Advertised bandwidth distribution (taken from [91]).

5.3.10 Latency observation

The graph 5.14 shows the overall performance when downloading 50 KB size static file over Tor from a server on the public internet. Download times from multiple analysis sources (op-hk5, op-nl5, op-us5) include complete downloads of the shown file size. The graph shows the range of measurements from the first to the third quartile and highlights the median. It's omitted the slowest and fastest quarter of measurements from the graph. This observation is important in order to establish a comparative baseline against our solution's latency measurement, where in 5.3.13.2, we discuss and evaluate both cases.

5.3.11 Circuit round-trip times

The graph 5.10b shows the round-trip latencies of circuits used for downloading static files of different sizes over Tor, from a server on the public internet. Round-trip latencies are measured as the time between sending the HTTP request and receiving the HTTP response header. The graph 5.10b shows the median of measurements as a thick line, the range of measurements from first to the third quartile as ribbon, and the highest and lowest non-outlier measurements as thin lines, with the first, second, and third quartile: 400 ms, 250 ms and 600 ms latency, respectively. In our solution, we have a minimum of 5 hops and a maximum of K + 5, where K is the number of available TIR and their clients, while the normal Tor as 5 hops (considered using the default 3 relay nodes). The addition of more hops consequently adds an extra latency to our solution.

5.3.12 Throughput Observations

The graph 5.10a shows the calculated throughput when downloading static 50 KB size files over Tor, from a server on the public internet. The graph shows the median of measurements as thick line, the range of measurements from first to third quartile as ribbon, and the highest and lowest non-outlier measurements as thin lines. We use this

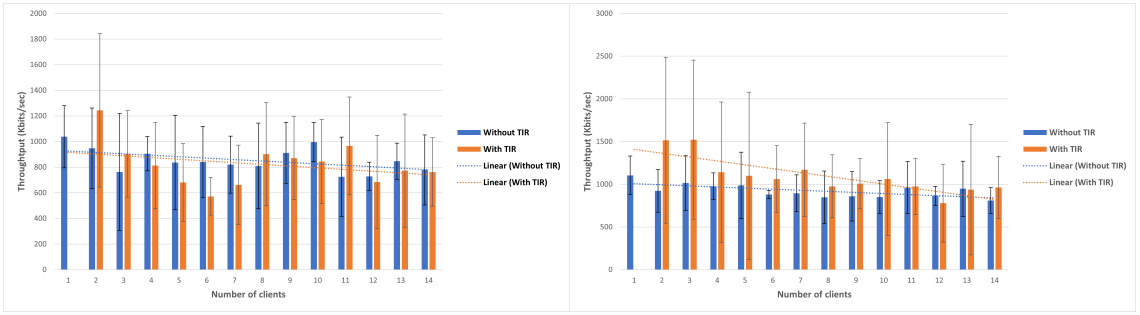
observation as a baseline to compare to our solution's throughput measurement, where in 5.3.13.1, we discuss and evaluate both cases.

5.3.13 Local client observed vanilla Tor performance

In order to compare the performance of the vanilla Tor operation with our solution, we conducted several experiments, which led to the following benchmarks of throughput 5.3.13.1, latency 5.3.13.2.

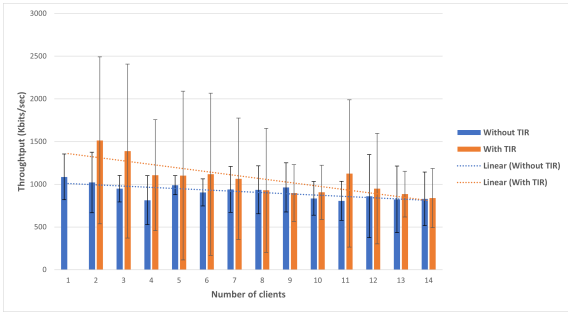
5.3.13.1 Iperf measurements

To measure the channel throughput, we used Iperf3. The client runs an instance through TIR and consequently through a specific 3-relay path performing the measure to the server. Figure 5.9 (represented in blue, without the TIR solution) depicts the achieved throughput based on the average bandwidth of 10 runs according to the increasing number of connected K users receiving chaff data. Instead, on same figure (represented in orange, with the TIR solution) depicts the achieved throughput according to the increasing number of connected K users receiving actual data through the current Tor operation.

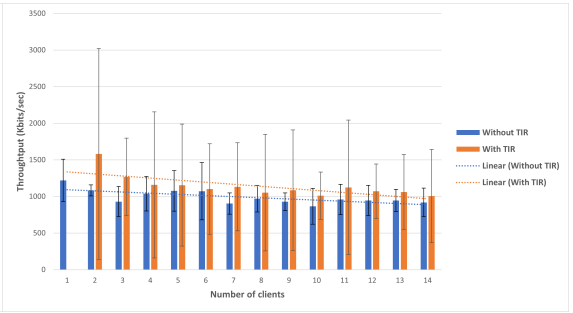


(a) Throughput with 512 bytes.

(b) Throughput with 1024 bytes.



(c) Throughput with 2048 bytes.

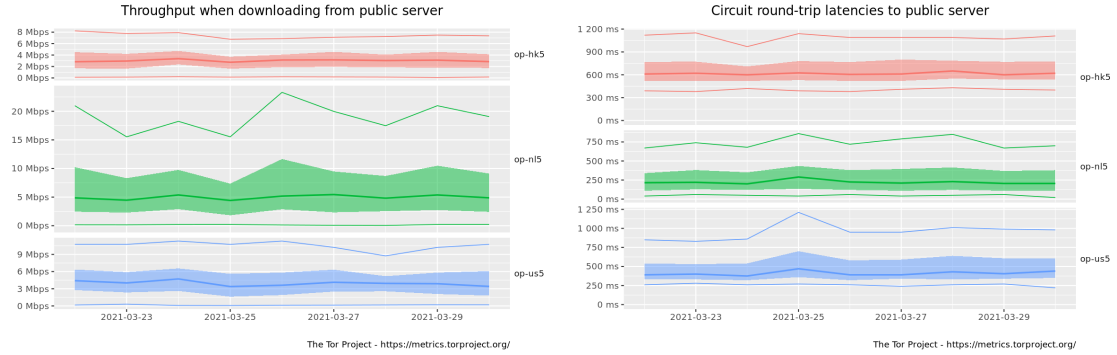


(d) Throughput with 4096 bytes.

Figure 5.9: System throughput measurement with different packet chunk sizes, being (a) 512, (b) 1024, (c) 2048 and (d) 4096 bytes, with standard deviation and linear tendency.

As we observe in figure 5.9, we conducted throughput measurement experiments for 4 different packet chunks sizes 512, 1024, 2048 and 4096 bytes with all 10 runs used the same Tor input nodes' relay configurations. Even though relays advertise bandwidth

as part of the internal Tor coordination and unlike advertised bandwidth per connection rates are not public, hence we conducted a throughput experiment (in 10 runs) without the use of TIR under the configuration of the same relay to measure the actual behaviour per client throughput of the testing relays, we observed an approximately 1 Mbps throughput (denoted as a signalized line, identified as Tor throughput baseline). Our results show that the one client can achieve up to, approximately, 1 Mbps of 1,5 Mbps maximum achieved by these relays when using the TIR tunnelled solution. With a higher number of connected clients, TIR decreases at a potential rate and thus reducing the available throughput. Even though, this reduction is supposed to be more critical when most clients use the K-anonymization P2P network, in fact, receiving useful data instead of chaff, because processing chaff frames is less burdensome than data frames since the former discards upon reception and the latter require receiving all chunks and deliver the traffic content to Tor. In our results using up to 14 clients, we have, sometimes, slightly better throughput results using data frames compared with chaff frames since we test the solution on an OVH VPS environment where some machines are in a closer region to the final server than the local machine and some Tor bridges and relays [32] hosted by OVH VPS itself [77]. With the increasing number of clients, we can observe that the linear tendency of the throughput of using TIR solution (in orange) will decrease more compared to without the TIR solution (in blue).



(a) Throughput when downloading static 50 KB files.

(b) Circuit round-trip latencies.

Figure 5.10: Throughput and latency measurements over Tor to public server (taken from [98]).

The graph 5.10a shows the median of measurements as a thick line, the range of measurements from first to the third quartile as ribbon, and the highest and lowest non-outlier measurements as thin lines, with the first, second, and third quartile: 3,5 Mbps, 5 Mbps and 2,5 Mbps respectively. If we analyse the hops of the normal Tor traffic circuit, we can count, in the standard scenarios using default settings of 3 relay nodes, 5 hops (client, 3 Tor relays and web target). In our solution, we have a minimum of 5 hops (Client / TIR node, 3 Tor relays and web target), the case where we only have 1 TIR available and a maximum of K + 5 (K Client / TIR nodes, 3 Tor relay and web target), where K is

the number of available TIR and their clients. From this observation, we conclude that with the increase in the number of TIR nodes available, the number of hops increases accordingly and decreases the throughput of our solution's performance, and we noticed that the increase of the packet chunk size slightly improves the throughput.

As the major results of the presented throughput operations, we can say that the TIR solution using K-anonymized circuits supported by TLS tunnelling has a lower throughput compared with the normal Tor operation. We notice that with the increase of the packet size, we had a small increase in the throughput values, independently of the connected clients and TIRs.

5.3.13.2 Httping measurements

To measure the latency of the channel, we conducted a round trip time (RTT) measure observed by the final clients. Using ICMP to measure the RTT, through the PING tool over the Tor network was not possible, since Tor does not route any other packets (namely ICMP) apart from TCP. Thus, we used the HTTPING tool instead. This tool performs an HTTP or HTTPS request and measures the time to receive the first byte of the header–header time to the first byte.

In the following figures 5.11, 5.12 and 5.13 (represented in blue) we depict the average latency of 10 runs according to the increasing number of K connected clients (TIR instances) receiving chaff data. Compared against, represented in orange, the achieved throughput observed with the increasing number of K connected clients receiving actual data through the Tor network. For this experiment we used 4 different packet chunks sizes 512, 1024, 2048 and 4096 bytes.

- We measured the ICMP echo request-reply from the HTTP file server endpoint localhost:80 to the local machine, with (represented in orange) and without (represented in blue) the TIR solution (shown in figure 5.11). From this figure, we can observe that the latency without TIR solution ranges between 600 and 1500 ms and with the TIR solution between 700 and 2000 ms.

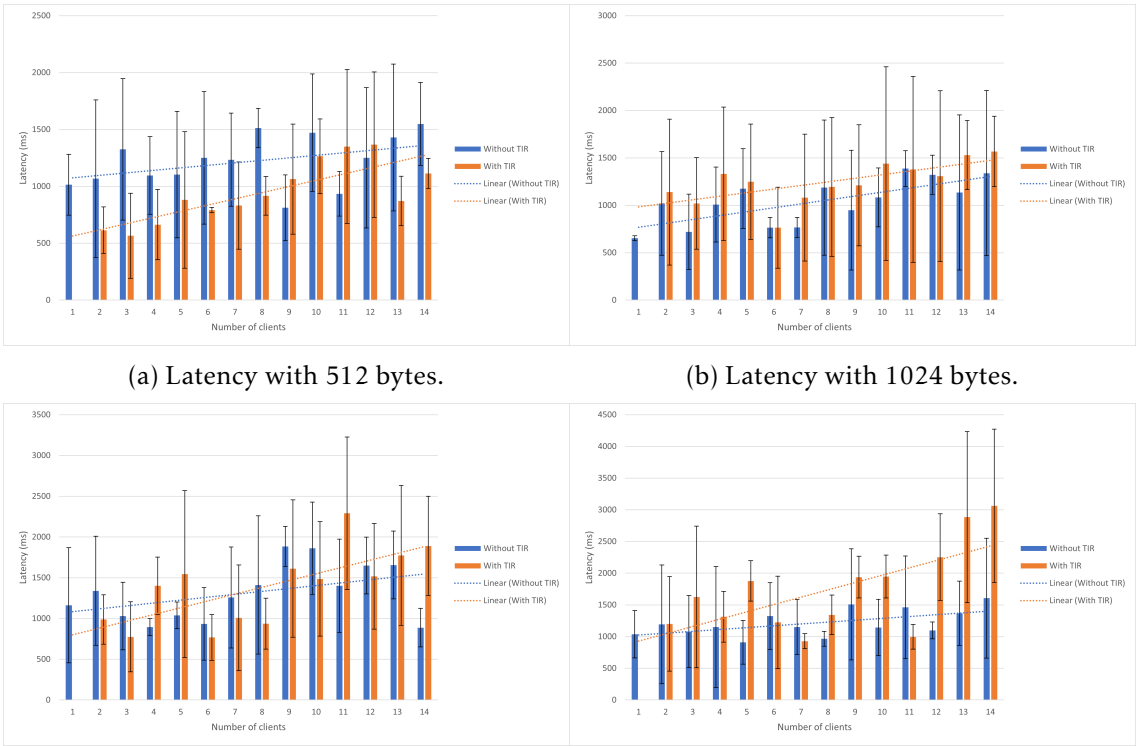
- We measured the HTTP file request by extracting 50 KB local files of the HTTP file server, from the HTTP file server endpoint localhost:1238/Files to the local machine, with and without the TIR solution (represented in figure 5.12). From this figure, we can observe that the latency without TIR solution ranges between 600 and 1500 ms and with the TIR solution between 600 and 2300 ms.

- We measured the HTTPS file request by extracting 50 KB web content of the New York Times website, from the HTTP file server endpoint https://www.nytimes.com/ to the local machine, with and without the TIR solution (represented in figure 5.13). From this figure, we can observe that the latency without TIR solution ranges between 900 and 1500 ms and with the TIR solution between 800 and 2400 ms.

The graph 5.14 shows the latency observation by completing a 50 KB file request in the normal Tor usage. Using this graph as a baseline to compare with our latency results shown in 5.12 and 5.13. We conclude that, both latencies results are very similar when using a few TIR instances, but using more TIR instances, we significantly aggravate the latency values.

As we can observe, the increase in the number of connected TIR instances causes the latency to increase, as we predicted. Data frames cause a slightly higher increases in latency than chaff frames. We can observe that the latency linear tendency of using TIR solution on figures 5.11, 5.12 and 5.13 (in orange) will increase more compared to without the TIR solution (in blue), for all the different packet chunk sizes. Contrary to the throughput performance increase with the increase of packet size, the latency decreases with higher packet sizes, because of the waiting for all the existing junk in the buffer data to finish sending throughout the Tor, which drives to extra latency.

It is important to notice that the bandwidth control measure is also in place (described in more detail later on). Therefore, the delivering process of data frames' content (to the Tor network) may be intentionally delayed as adjustable reactive measures against possible active attacks.



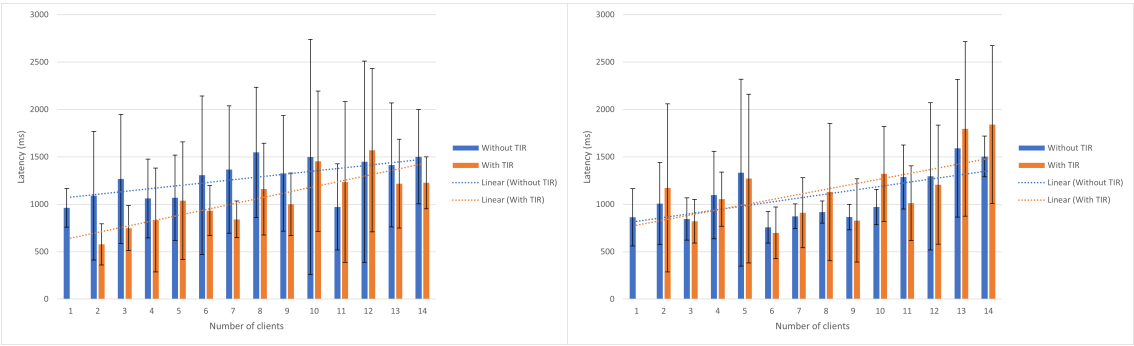
(a) Latency with 512 bytes.

(b) Latency with 1024 bytes.

(c) Latency with 2048 bytes.

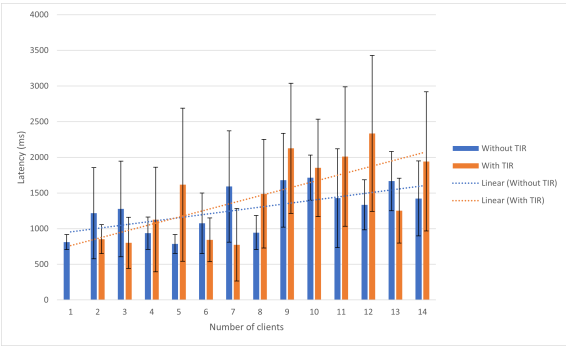
(d) Latency with 4096 bytes.

Figure 5.11: Server latency measurement with different packet chunk sizes, being (a) 512, (b) 1024, (c) 2048 and (d) 4096 bytes, with standard deviation and linear tendency.

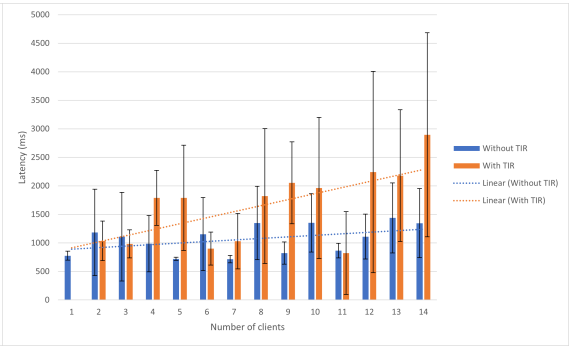


(a) HTTP request latency with 512 bytes.

(b) HTTP request latency with 1024 bytes.



(c) HTTP request latency with 2048 bytes.



(d) HTTP request latency with 4096 bytes.

Figure 5.12: HTTP 50 KB file request measurement with different packet chunk sizes, being (a) 512, (b) 1024, (c) 2048 and (d) 4096 bytes, with standard deviation and linear tendency.

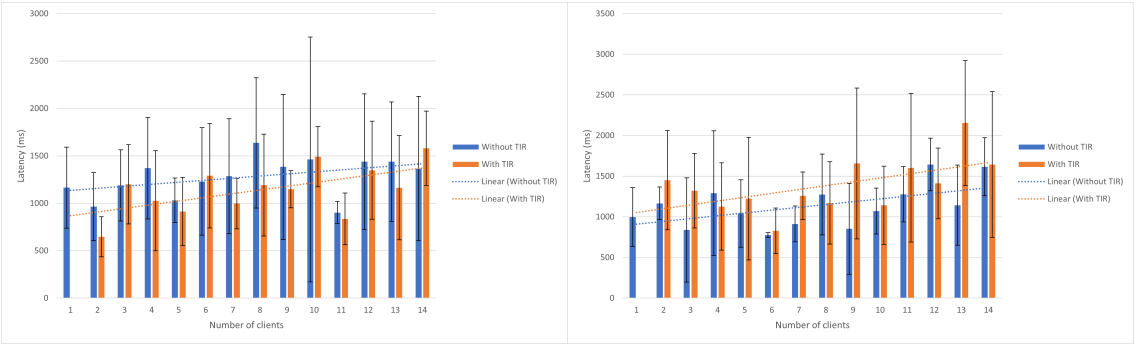
5.3.13.3 Traceroute observations

Here we have traceroute from the local machine located in Portugal to the HTTP file server located in Gravelines, where we observed a maximum of 30 hops with 60 byte size packets (traceroute default values).

We couldn't route the full path from the local machine, through Tor network to the HTTP file server because of the Tor circuit, because it is extended one hop at a time, and each relay along the way knows only which relay retrieved the data and which relay it is delivering data to. No individual relay ever knows the complete path that a data packet has taken. The client negotiates a separate set of encryption keys for each hop along the circuit to ensure that each hop can't trace these connections as they pass through.

5.3.13.4 Jitter observations

Tor connections experience large fluctuations on the network, significantly above regular Internet connections. These major perturbations result from congestion on Tor relays, which arises from the imbalance between Tor's capacity and clients' bandwidth needs, therefore, many Tor packets are fragmented and re-packetized due to unreliable network conditions. For best performance, we must keep the jitter below 20 ms (milliseconds). If



(a) HTTPS request latency with 512 bytes.

(b) HTTPS request latency with 1024 bytes.

(c) HTTPS request latency with 2048 bytes.

(d) HTTPS request latency with 4096 bytes.

Figure 5.13: HTTPS 50 KB file request measurement with different packet chunk sizes, being (a) 512, (b) 1024, (c) 2048 and (d) 4096 byte, with standard deviation and linear tendency.

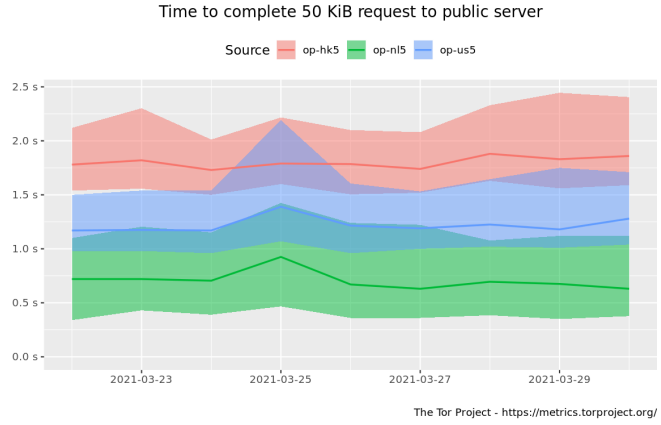


Figure 5.14: Latency observation by completing 50 KB file request (taken from [91]).

```
traceroute to 51.83.75.29 (51.83.75.29), 30 hops max, 60 byte packets
 1  router.home (192.168.1.1)  0.671 ms  4.608 ms  4.568 ms
 2  10.22.127.254 (10.22.127.254)  13.085 ms  12.054 ms  13.010 ms
 3  10.137.218.17 (10.137.218.17)  13.995 ms  13.955 ms  13.922 ms
 4  10.255.184.94 (10.255.184.94)  19.161 ms  17.532 ms *
 5  * * *
 6  * be2313.ccr31.bio02.atlas.cogentco.com (154.54.61.101)  32.644 ms  37.405 ms
 7  be2324.ccr31.mad05.atlas.cogentco.com (154.54.61.130)  44.986 ms  46.521 ms  46.781 ms
 8  mad-mad2-pb1-ptx.es.eu (91.121.131.97)  46.448 ms  46.410 ms  46.373 ms
 9  * * *
10  * * *
11  * * *
12  be102.gra-g1-nc5.fr.eu (91.121.215.176)  55.524 ms  57.798 ms  56.540 ms
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  vps-f3057f45.vps.ovh.net (51.83.75.29)  56.858 ms  55.359 ms  53.673 ms
```

Figure 5.15: Traceroute to HTTP file server.

in our results we exceed the 30 ms threshold, then it would cause a noticeable impact on the quality of any web application [31] or stream application. Since we are using Tor, being a high-latency network, it is normal that in our results we have higher jitter values compared with the normal web client jitter. The graph 5.16 presents our measurement of a single client (TIR), using StarTrinity [88] software tool. We achieved a network jitter value of approximately 27ms average in a time interval of 45 seconds, with a 3,24s standard deviation, and the network randomly dropped 1% of the packets.

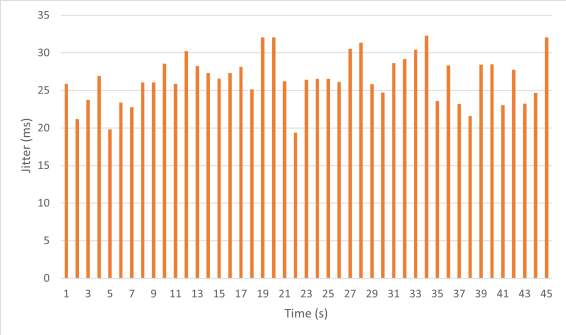


Figure 5.16: Client's jitter.

5.4 TIR benchmarking and experimental observations

5.4.1 Bandwidth measurements

In figure 5.17 we see the measurement of the client (local machine) bandwidth using different packet chunk sizes, resulted from the test bench environment (previously explained in 5.2.2). We transmitted approximately 50 KB content, for the 512, 1024, 2048 and 4096 bytes packet size we had 4638, 4839, 5358 and 5365 Kbits/sec of bandwidth, respectively. We can see that with the increase of the packet size, we can observe a better estimation of transfer rate.

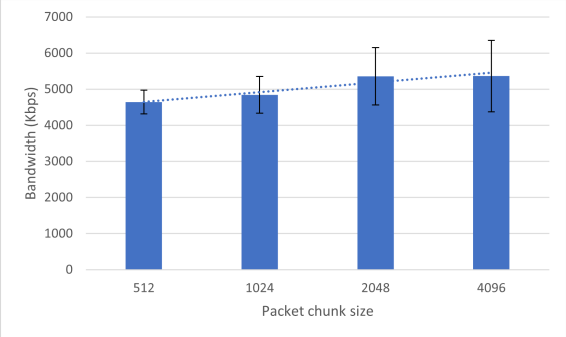


Figure 5.17: Client's bandwidth measurement using different packet chunk sizes: 512, 1024, 2048 and 4096 bytes.

5.5 Experimental observations of unobservability metrics

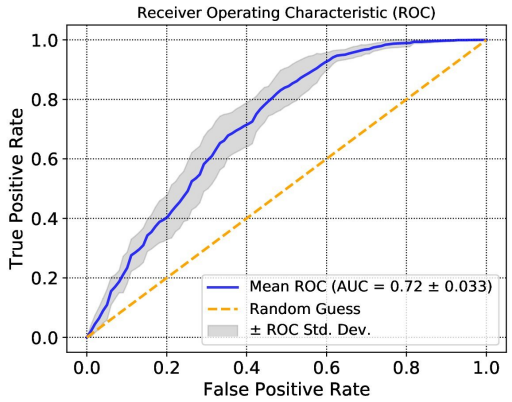
In this section, we present the results from three traffic correlation observations. We observed these in a sequential dependency order, meaning that the previous results influence the next observation. We already discussed that the objective of a censor entity is to analyse the traffic flow in the entry and exit nodes of the Tor network to find a correlation between these two traffics. In our solution, we try to hamper these observations by heightening the traffic indistinguishably by enabling the selection of different packet chunk sizes and using the multi-path routing strategy, which introduces some anonymization in the entry Tor circuits. The TIR itself introduces a pre-anonymization effect on the input traffic, it resizes the incoming packets into smaller or larger ones, depending on the initial TIR parameterization, shown in the first observation 5.5.1. Next, in the second observation 5.5.2, we increment the number of TIR to see the effect of traffic segmentation. Finally, the third observation 5.5.3 determines how efficient our solution is on hiding covert data in chaff traffic. As we previously explained in 5.1.3, we measure the traffic unobservability, using the XGBoost model classifier by observing the AUC (the measure of separability) and the ROC (probability curve). With these observations, we can determine how much the model can distinguish between input and output traffic (for the first and second observation) and between covert and chaff data (on the third observation). We use the same packets captured file with an average packets sizes of 4000 bytes to inject into the TIR node, where we probe the input traffic and the digested output traffic. In the graphs below, we represent the ROC curve plotted with TPR against the FPR, where TPR is on the y-axis and FPR is on the x-axis and the AUC value is the area under the ROC curve. We also have the random guess, meaning that a binary classifier as inability to get better results than random guessing, a condition in which the true-positive rate equals to the false-positive rate, presented as the line TPR=FPR. Note that all the TIR nodes setup for the observations are deployed in the local machine.

5.5.1 Excepted unobservability strengthening with TIR input circuits

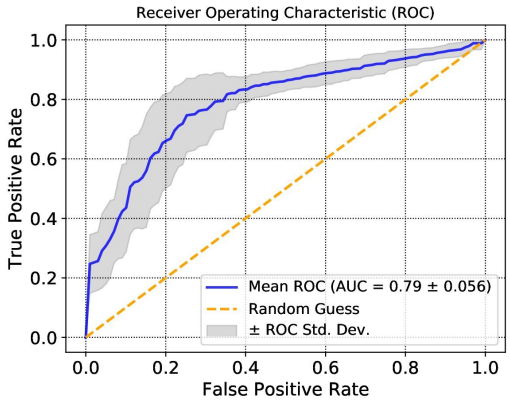
These observations present the starting point, where we select the ideal TIR parameterization in order to proceed with the next observations. The idea of this first observation is to use one TIR node and to observe the effect of traffic unobservability that one TIR can provide, when comparing to the segmentation of produced output traffic, against the initial injected traffic onto the TIR node. In other words, how much one TIR can pre-anonymize the initial traffic. The only parameter we regulate is the packet sizes of the output traffic to 512, 1024, 2048 and 4096 bytes. After analysing for the four different packets sizes, we got the graph 5.18a using 512 bytes with an AUC of 0.72, graph 5.18b using 1024 with an AUC of 0.79, graph 5.18c using 2048 bytes with an AUC of 0.83 and finally graph 5.18d using 4096 bytes with an AUC of 0.98. We can see that with smaller packet sizes, the ROC curve slightly moves towards the random guess, the reason for this is that we are injecting a packet size of 4000 bytes on average, which is a similar packet size to the output traffic, making them almost identical when compared, the bigger the gap difference between packet sizes, the harder will be to predict incoming packets compared to the outgoing packets, presenting lower TPR against FPR values. So we can conclude that the optimal solution is the observation 5.18a, which is the baseline for the next observation 5.5.2.

5.5.2 Unobservability against traffic analysis

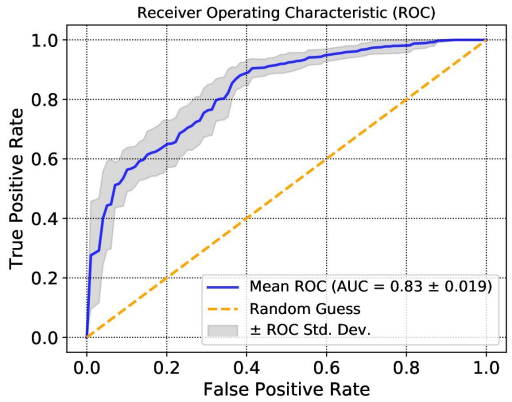
In this observation, we intend to show how, with the addition of TIR nodes, we can increase the unobservability of the outgoing traffic against the incoming traffic, using the optimal solution's parameterization from the first observation described above. For this experiment, we set up 1 up to 4 TIR nodes instances; we injected the packets sample into one TIR and used this TIR node to analyse the incoming packets and the packets that reached the Tor input node. For each TIR node added, we compared the incoming flow with the outgoing flow to the Tor input node. Since we are using the multi-path routing strategy between TIR nodes, the traffic evenly distributed. The injected packets disseminate from the observed TIR to the others, this will affect the number of outgoing packets to the Tor in the observed TIR, the more distributed this number is between the TIR nodes, the harder it will be to correlate both traffic, meaning a better unobservability. Once we analysed the two flows in one TIR, we got the graph 5.19a same graph from the optimal observation with an AUC of 0.72, graph 5.19b with 2 TIR running, with an AUC of 0.69, graph 5.19c with 3 TIR running with an AUC of 0.65 and at last graph 5.19d with 4 TIR running with an AUC of 0.59. We conclude that with higher number of running TIR, there are more possible ways to disperse the initial traffic, thus achieving better results of AUC, meaning that the ROC curve moves towards the random guess line.



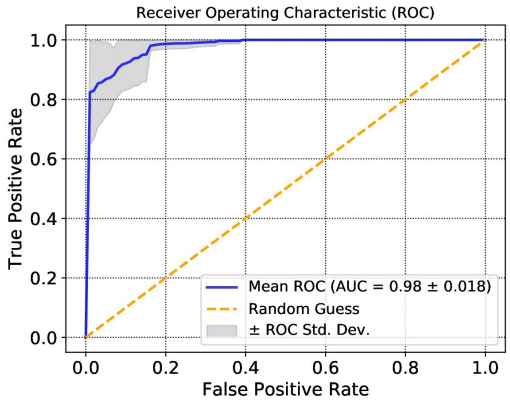
(a) Configured to 512 bytes.



(b) Configured to 1024 bytes.



(c) Configured to 2048 bytes.

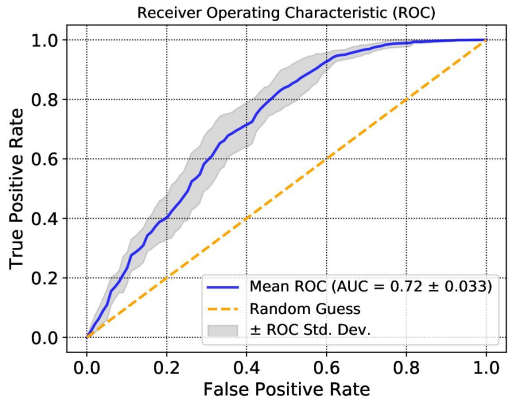


(d) Configured to 4096 bytes.

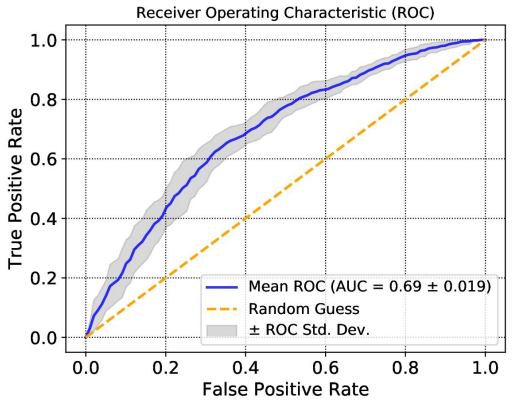
Figure 5.18: TIR input circuit correlation analysis using XGBoost, using one TIR configured with different packet chunk sizes, being (a) 512, (b) 1024, (c) 2048 and (d) 4096 bytes.

5.5.3 Indistinguishably between chaff and covert data

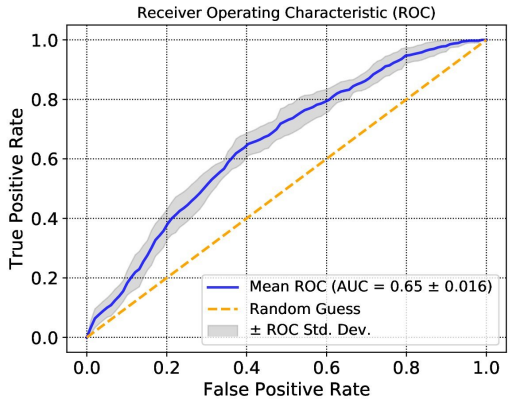
In this last observation, we intend to measure the indistinguishability between chaff and covert data by detecting the amount of covert data in the chaff traffic. For the chaff data, we used the same traffic as explained above in the other two observations, and for the covert data, we used a packet captured file with only TCP packets. To compare both traffic, we first captured the total outgoing TIR node chaff traffic and saved it into a packet captured file. Next, we used 1 up to 4 TIR nodes instances (identical set up from 5.5.2, and in this case, we inject both chaff and covert traffic, and we captured the outgoing traffic of one TIR node. Again, because of the multi-path routing strategy between TIR nodes, the traffic is shared, and this effect affects the number of observed outgoing packets against the total outgoing traffic. Finally, to correlate both outgoings traffics, we analyse the total chaff traffic and the portion of mixed chaff and covert traffic from one TIR, giving us an estimated AUC and ROC curve for the 1 up to 4 set up TIR nodes instances. For one TIR node we got the graph 5.20a with an AUC of 0.69, graph 5.20b with 2 TIR running,



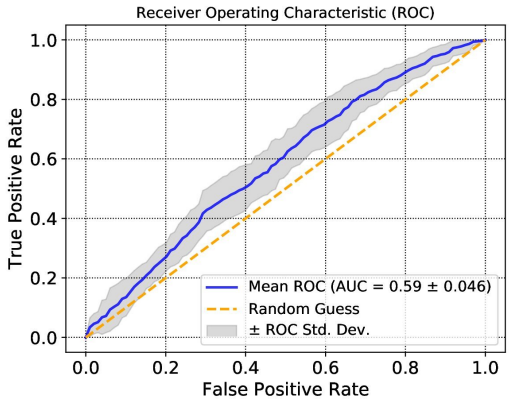
(a) One TIR setup.



(b) Two TIR setup.



(c) Three TIR setup.



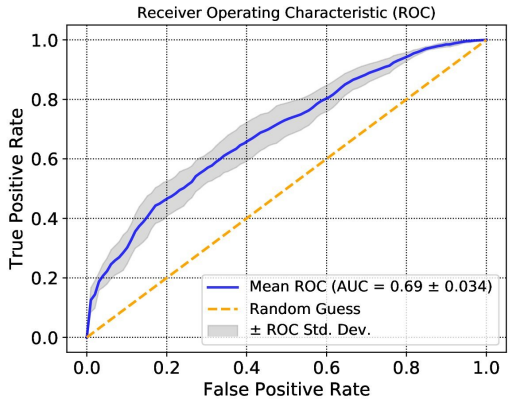
(d) Four TIR setup.

Figure 5.19: Traffic analysis using XGBoost, using one to four TIRs, configured with 512 bytes packet chunk size.

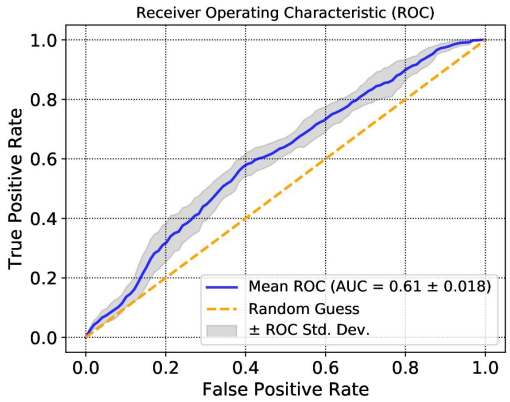
with an AUC of 0.61, graph 5.20c with 3 TIR running with an AUC of 0.57 and at last graph 5.20d with 4 TIR running with an AUC of 0.54. With the increase number of TIR nodes, we get evenly dispersed traffic amongst TIR nodes, and we can get a better indistinguishability between the chaff and covert traffic. In this way, with higher number of TIRs, the ROC curve tends to lean towards the random guess because we get less TPR against the FPR.

5.6 TIR utilization of resources

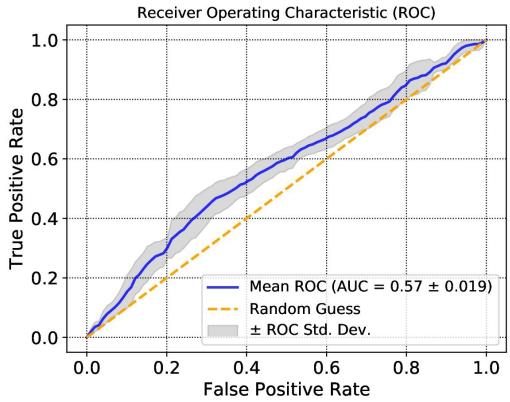
To understand the minimum requirements for the TIR system to run in a machine, in this section, we gathered information about CPU and workload 5.1 and the network metrics 5.2. To measure the resources used by one TIR instance2, we ran a benchmarking test on a single TIR node. Next, we set up an environment where we wanted to see the impact of having 4 TIR nodes interacting with each other. Consequently, we see that with more TIR nodes, each TIR will handle multiple parallel requests, slightly degrading its



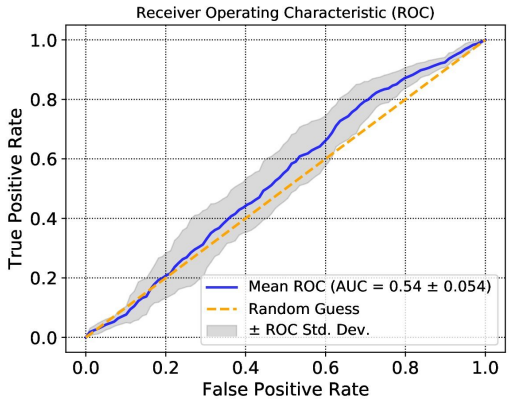
(a) One TIR setup.



(b) Two TIR setup.



(c) Three TIR setup.



(d) Four TIR setup.

Figure 5.20: Chaff and covert data analysis using XGBoost, with one to four TIRs, configured with 512 bytes packet chunk size.

performance.

5.6.1 CPU and workload

Here we show the CPU and workload and some other metrics in table 5.1. We used the *top* program, which provides a dynamic real-time view of a running system. It can display system summary information and a list of processes or threads currently being managed by the Linux kernel. If we compare the percentage of processor used employing 1 and 4 TIR nodes, we can see that with the increment of TIR nodes, the CPU usage percentage slightly increases, the size of pages, real-memory text size of the process and the amount of shared memory by task. We must take into account that, there is a huge increase in memory use with 4 TIR nodes. This is because we are testing in a single machine, the page sizes and real memory increase exponentially because the 4 TIRs are communicating with each other, each using Stunnel service and Tor service.

Table 5.1: CPU and workload metrics using 1 and 4 TIR nodes.

Description	One TIR	Four TIRs
The percentage of the processor that is used in the last interval.	1%	1,2%
The size of the pages in kilobytes.	11932	1118520
The sum of real-memory data (resident set) and real-memory (resident set) text size of the process.	3742	267680
The sum of real-memory data (resident set) and real-memory (resident set) text size of the process.	4	4
The real-memory text size of the process.	7515	481224
The real-memory data size of the process.	0	0
The amount of shared memory used by a task.	481	28992

5.6.2 Network requirements

We collected the network metrics for the system requirement (shown in table 5.2). We used the *nmon* program with *n* option, that displays and records local system network information. Likewise, we only used request to retrieve 50 KB size files, which checks with the size of data of 1 TIR process being received (from the Tor network) and transmitted (to its client) is equals to what we requested. With 4 TIR nodes, we see the parallel effect of receiving and transmitting more data compared with only 1 TIR, as well the number of packets, the average size of packets and the peak value of received and sent data.

Table 5.2: TIR network metrics using 1 and 4 TIR nodes.

Description	One TIR	Four TIRs
Data that are received in kilobytes per second in the interval	50.0	168.1
Data that are transmitted in kilobytes per second in the interval	50.0	168.1
Number of packets that are received in the interval	22.5	43.5
Number of packets that are sent in the interval	22.5	43.5
Average size of packet that is received in the interval	2278.1	3961.0
Average size of packet that is sent in last interval	2278.1	3961.0
Peak value of received data in kilobytes per second	103.4	201.0
Peak value of sent data in kilobytes per second	103.4	201.0

5.7 Summary and validation remarks

This chapter presented our experimental evaluation of TIR for the validation of the proposed solution using our current available prototype. As reported in the chapter, we conducted an extensive evaluation, analysis and experimental observations focused on performance and unobservability metrics.

Our observations show the validity of the solution as a contribution to enforce anonymization and privacy-communication on the Internet, with strengthening mechanisms for the Tor internetworking environment. Our unobservability evaluation experiments targeted in the validation of the TIR design and implemented prototype can be analysed under the following unobservability arguments:

- **Anonymity Preservation**: Covers both end-users and final targeted servers intermediates by the TIR Tor strengthening K-anonymization network censorship-resistant

privacy-enhanced system (CRPES). In this dimension of unobservability, the solution preserved the end-user anonymity, not revealing identifiers (such as IP addresses, IP-reverse resolutions or location) in sending/retrieving information for correlation attacks targeted in the Tor input and exit circuits.

- **Unlinkability**: The "end-user to targeted server" path (and vice versa) is unlinkable, in the sense that the censor (not omnipresent for controlling all the links in all AS regions and links on the Internet) cannot correlate and learn who is the end-user, considering that the opponent unknown all the dynamic pre-staging paths in the ad hoc organization of the TIR usable instances running in different AS-regions with locations all around the Internet. We mention the TIR has a "friendly" design, allowing that the K-input anonymization circuits can be organized or reorganized under ad hoc settings and the anonymization circuits can also use in the future different protocol-encapsulations including media-streaming carrying transports. TIR instances can act as "private" TIR appliances running and accessible through cloud-enabled transport protocols with heterogeneous forms of traffic encapsulation. Then, the unlinkability property can be generalized to avoid traffic analysis observers in the whole system, not only necessarily a censor, considering a "game" in which the opponents cannot block in practice all possible inbound/outbound traffic intercrossing AS regions, and, traffic disabling procedures for all carrier transports or all cloud computing services in different AS region.

- **Incoercibility**: This principle addresses our solution by considering that the censors are not credible because the node is outside the control of the censor activities or because it cannot comply in practice with more powerful censor's demands, nor practical acceptable "full isolation" procedures (even considering the most aggressive political regimes in censorship activities).

- **Deniability**: A node in the whole CRTES environment promoted by the conjugation of TIR with Tor system (either end-users, TIR nodes, Tor input bridges and relays, Tor intermediary relays, Tor rendezvous nodes, Tor output relays and final servers), arise reasonable doubts in the correlation of internetworking activities in useful time. TIR nodes usually enforce this property through encryption and secure steganography, and we must notice that nothing hinders in the design of the proposed solution that payloads sent or forwarded cannot adopt this type of protection, as covert channels supported as packet payloads. TIR achieves the objective of providing K-anonymized input circuits that can generate indistinguishable channels with better immunity against traffic correlation deanonymization attacks provided by censorship opponents. We observed that our TIR solution provides a reasonable throughput and latency needed for typical network activities (e.g., browsing), not aggravating the baseline latency and throughput conditions as observed currently

when using the vanilla Tor solutions and related protocols such as onion routing environment.

As of the last remarks from our observations, we can argue that TIR shows good resiliency as a strengthening solution against Tor correlation attacks. We notice that powerful active attackers performing packet drops or delays do not identify clients behind the TIR pre-staged Tor circuits. From our experiments, we understood how the TIR configuration parameters are related to network usage and end-users observed performance optimizations, to preserve unobservability without degrading the performance.

Chapter 6

Conclusions

The problem of Internet censorship activity against Internet users is today a well-known reality. Different organizations, conduct censorship actions, such as governmental authorities, Internet service providers, or content-distribution players. Sometimes those entities, in a collaborative effort, in so-called "state-level censorship activities" — a term that emerged in the related research area. Censorship activities are conducted in different countries, with different purposes. Current technology allows an interested censorship party (for instance, an ISP) to close monitor the communications of any user in a certain AS (or Autonomous System) routing region. This allows uncovering of data such as political views, consumer preferences, healthcare data, and other private information. In the research agenda and literature, we find an extensive investigation on a variety of research goals and contributions around the problem. We can characterize these contributions in two main categories: on one side, many researchers try to propose techniques, mechanisms, and tools, to fight against censors (regarded as adversaries), in order to offer better solutions for censorship circumvention and privacy preservation of Internet users; on the other side, there are considerable advances in powerful censorship methods, including advanced machine learning-based techniques and tools, for real-time traffic analysis and correlations for deanonymization of users' traffic, and to fight against the ongoing methods that have been proposed for privacy-enhanced communication. Both sides of this "arms' race" produces interesting and innovative solutions, as two different facets of the problem. This dissertation is more aligned with the study and investigation of mechanisms for censorship circumvention improvements, to mitigate the consequences of Internet censorship activities, considered in this case censors as "adversaries".

The Tor network is the more popular internetworking environment for Internet-based traffic anonymization. Tor has been many times in the centre of the above research arena. At the same time, Tor and the related provided tools, are used by a large community of

users, for the good and also for the bad reasons. By using the Tor network, users want to remain anonymous when using the Internet for traffic exchanges, in particular for Web browsing or for the access to targeted Web-enabled applications and services. The Tor network deployed mechanisms allow, in its operation, that both senders and receivers can exchange Internet traffic, remaining as anonymous endpoints, while communicating with possible confidentiality guarantees. In the dissertation, we studied the base mechanisms implemented today in the Tor system for this purpose.

Anonymity networks, such as Tor, have been designed to defeat network surveillance and traffic analysis activities, to preserve the anonymity of users when they use Internet applications and related standard protocols. Unfortunately, despite the interesting solutions that have been incorporated, the current Tor specification and its operation have vulnerabilities exploitable by traffic correlation attacks, in particular, those based on recent advanced machine learning methods used today by censors. Traffic correlation between input and output Tor circuits, and the more or less easy enumeration of Tor nodes in specific AS (Autonomous Systems) routing regions, allow censorship activities to discover mappings between destination targets and users in the origin of the related traffic. These correlations can also be improved by adversaries with access to multiple hops in Tor end-to-end circuits, which can improve, with high accuracy, the deanonymization of involved endpoints. This defeats Tor's purpose as an anonymous system with sufficient expected privacy preservation to its users.

The most successful correlated attack to date uses combinations of machine learning approaches for real-time traffic inspections. Some observed traffic features allow the analysis of network flows, by correlating inter-packet leaving versus inter-packet arrival times and involved packet lengths, even when the payloads of such packets are protected by secure protocols in the TCP/IP stack, such as the case of HTTPS and TLS.

In this dissertation, we focused on improving the use of the current Tor system, to make it efficient and robust against correlation attacks conducted by worldwide passive adversaries. We focus the approach on the avoidance of traffic correlations examining flows on Tor established circuits, by observing ingress traffic from users on Tor input nodes and egress traffic of output nodes to the final targeted systems. The baseline of our approach is on the use of k-anonymity input circuits, in a pre-staged multipath environment of volunteering nodes between the user and the input Tor bridges. These nodes can be randomly managed in the context of ad hoc communities, to route packets in TLS tunnelled connections from each sender to multiple sources of Tor input connections, maybe located in different AS regions. The main idea in the approach is to mitigate the hypothesis for input/output Tor traffic correlations by decoupling the input traffic with the help of those pre-staged nodes. The goal is to reduce the possibility for an adversary to map the complete input traffic from a user with observed output traffic on a targeted internet site used as a destination.

6.1 Main contributions

Following the stated hypothesis, we proposed the TIR solution. We designed TIR as a censorship-resistant Tor pluggable transport candidate solution that leverages the notion of indistinguishable and TLS tunnelled k-anonymity circuits used to decouple the user traffic from traffic segments on Tor input nodes.

We implemented TIR in a prototype for the validity of the proposed solution and to conduct an experimental assessment testbench. With our experimental observations, we obtained interesting results for the viability of the solution, as well as, to obtain input results for the performance evaluation and unobservability criteria provided with the designed approach.

The TIR prototype is available as an open-source solution that can be used by researchers for complementary studies and possible future improvements. We performed a thorough evaluation of the current TIR implementation for dynamic adaptation capabilities, to manage the trade-off between unobservability, throughput, bandwidth, and resistance against possible network perturbations. These perturbations can be injected by censors on traffic flows directed to Tor input nodes to study the effects on the Tor output traffic.

6.2 Open issues

In the initial achieved goals targeted as the dissertation objectives, some open issues emerge as primary actions for subsequent tasks. The first direction is to further broaden the scope of evaluations on the assessment of non-observability criteria and analysis of correlation avoidance. Our initial results show promising evidence that some machine learning techniques used in censorship activities and in recent research proposals around the Tor traffic analysis are ineffective under the protection of the TIR solution, compared with the observation of the Tor environment, as it is used today. However, the improvement of the observations in large-scale settings is an obvious step for a required deep analysis of traffic indistinguishability criteria. At the same time, there is a space to study the TIR unobservability criteria using other machine learning techniques using different complementary traffic flow attributes in different traffic headers and payload features. This could be also extended to analyse the non-observability and non-correlation criteria in different supported protocols carried by the TIR multipath environment. In this direction, it is necessary for an in-depth evaluation, where large amounts of synthetic flows must be generated, based on real full anonymous Tor onion services and traffic traces.

Another open issue that can be immediately addressed from the current TIR design and implementation, is the optimization of the prototype to better manage large and concurrent client workloads. We anticipate that there is a space for improvement in the current implementation of the TIR software that could provide better support for TIR nodes to process and to manage concurrent traffic flows received from multiple user

sources and multiple protocols and tunnelled in multiple TLS connections proxied to different Tor input bridges.

6.3 Future work directions

Our work on the TIR design, implementation and experimental evaluation allow us to identify other interesting directions for future research. We term. We present the following directions as work that could be addressed in medium and long term:

- The implementation of TIR should be enhanced with the support for remote software attestation running in TIR nodes, which in turn would help bridges and clients to attest that the communicating parties are running trusted and unmodified TIR releases. In this direction, we can also include the possibility to reimplement TIR as a trusted container that can run in a more controlled and isolated environment, for example, backed by hardware-enabled technology, such as ARM Trust Zone or Intel SGX enclaves.

- In the line of TIR optimization and to support TIR's performance-sensitive operations, like bridges' load balancing behaviour and k-circuits' dynamic rearrangements, the solution could be improved, requiring further assessment when facing a large pool of users and more exigent in/out traffic workloads. Specifically, better load balancing capabilities and dynamic selection of different routing tunnels may help more resource-constrained nodes to provide a reasonable quality of service, with expected improvements in maintaining proper throughput and latency conditions. In its turn, k-circuits rearrangements — regarded as a constrained optimization problem — is a possible challenging research topic demanding an exhaustive analysis and a comprehensive algorithm to find the best balance between client workloads, the redistribution of new users among available k-circuits to prevent bandwidth starvation, and a membership management environment to deal with the dynamic addition/choice/removal of TIR nodes.

- Future work should also focus on other strategies to create and manage more robust indistinguishable channels. In this direction, there is a possibility to use alternative methods for traffic tunnelling strategies for TIR, for example using unobservable traffic encapsulation between TIR nodes carried by overlaying with video-conferencing protocols. These encoding techniques show high resistance against traffic correlation and deep-leaning methods, representing today one of the most promising approaches for censorship circumvention.

- There is an interesting study that can be done to take advantage of spared network approaches that may be meaningful for users with network restrictions (e.g., supporting the TIR environment on mobile networking and 5G support).

- At last, a topic for envisioned future work is to look at how to tackle incentives for participation in a TIR ecosystem. To this end, it may be important to design and implement an award, scoring and autonomic acknowledgement system, in which users that provide nodes with the best TIR capabilities in a certain community of users must also have more benefits from the TIR nodes provided by others.

Bibliography

[1] B. A., P. I., and W. R. "TorScan: Tracing Long-Lived Connections and Differential Scanning Attacks." In: vol. 7459. 2012, Retrieved 29 June 2020. DOI: 10.1007/978-3-642-33167-1_27.

[2] C. A. and T. M. "Data and structural k-anonymity in social networks." In: Privacy Security and Trust in KDD: Second ACM SIGKDD International Workshop, Aug 2009, Retrieved 3 Junly 2020.

[3] J. A., W. C., J. R., S. M., and S. P. "Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries." In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*. CCS '13. New York NY USA: Association for Computing Machinery, 2013 , Retrieved 26 June 2020, 337–348. ISBN: 9781450324779. DOI: 10.1145/2508859.2516651. URL: https://doi.org/10.1145/2508859.2516651.

[4] M. A., W. T., J. R., and J. A. "Understanding Tor Usage with Privacy-Preserving Measurement." In: *Proceedings of the Internet Measurement Conference 2018*. IMC '18. New York NY USA: Association for Computing Machinery, 2018, Retrieved 24 June 2020, 175–187. ISBN: 9781450356190. DOI: 10.1145/3278532.3278549. URL: https://doi.org/10.1145/3278532.3278549.

[5] N. A. and S. V. "Robust de-anonymization of large sparse datasets." In: IEEE Symposium on Security and Privacy, May 2008, Retrieved 4 Junly 2020.

[6] *Australian Cybersecurity Centre - Australian Government, Defending Against the Malicious use of the Tor Network*. Retrieved 20 Feb 2021. Oct 2020. URL: https://www.cyber.gov.au/acsc/view-all-content/publications/defending-against-malicious-use-tor-network.

[7] C. B. and I. G. "Slitheen: Perfectly Imitated Decoy Routing throughTraffic Replacement." In: Proceedings of the 2016 ACM SIGSAC Conference on Computerand Communications Security Vienna Austria October 24-28 2016: ACM, 2016, Retrieved 1 May 2020, 1702—1714. DOI: 10.1145/2976749.2978312. URL: http://doi.acm.org/10.1145/2976749.2978312.

[8] D. B., N. S., and L. R. "DeltaShaper: Enabling Unobservable Censorship-resistant TCP Tunneling over Videoconferencing Streams." In: Proceedings on Privacy Enhancing Technologies 4, 2017 , Retrieved 16 May 2020, 1–18. URL: https://petsymposium.org/2017/papers/issue4/paper15-2017-4-source.pdf.

[9] D. B., N. S., and L. R. "Effective Detection of Multimedia Protocol Tunneling using Machine Learning." In: *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore MD: USENIX Association, 2018 , Retrieved 9 Jul 2020, pp. 169–185. ISBN: 978-1-939133-04-5. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/barradas.

[10] K. B., Y. F., K. R., and J. S. "Detecting and Evading Censorship-in-Depth: A Case Study of Iran Protocol Whitelister." In: USENIX Association, August 2020, Retrieved 10 Set 2021. URL: https://www.usenix.org/conference/foci20/presentation/bock.

[11] K. B. and T. T. "A practical approximation algorithm for optimal k-anonymity." In: Data Mining and Knowledge Discovery, Jul 2012, Retrieved 3 Jul 2020.

[12] U. B. *How does Tor actually work?* Retrieved 8 April 2021. 2019. URL: https://hackernoon.com/how-does-tor-really-work-5909b9bd232c.

[13] A. C. *Digital civic engagement by young people*. CRC Press. Retrieved 8 April 2021. March 2020. URL: https://www.unicef.org/globalinsight/reports/digital-civic-engagement-young-people.

[14] C. C., D. H., E. A., and G. R. "Leaping Over the Firewall: A Review of Censorship Circumvention Tools." In: 18 (3 May 2019, Retrieved 27 June 2020), pp. 837–856. URL: https://freedomhouse.org/sites/default/files/inline_images/Censorship.pdf.

[15] D. C. "Differential privacy." In: 33rd International Conference on Automata Languages and Programming Volume Part II Berlin (DE), May 2008, Retrieved 4 Junly 2020.

[16] N. C. "Traveling the silk road: a measurement analysis of a large anonymous online marketplace." In: International Conference on World Wide Web, 2013, Retrieved 26 Feb 2021.

[17] W. C., T. H., B. S., and S. M. "An empirical evaluation of relay selection in tor." In: Network and Distributed System Security Symposium (NDSS), Feb 2013 , Retrieved 29 June 2020.

[18] M. Ca. and A. M. "Botnet over tor: The illusion of hiding." In: International Conference On Cyber Conflict, 2014, Retrieved 22 Feb 2021.

[19] *Cyberwarfare*. Wikipedia. Retrieved 13 May 2021. URL: https://en.wikipedia.org/wiki/Cyberwarfare.

[20] A. D., C. O., and P. G. "Analyzing China's Blocking of Unpublished Tor Bridges." In: Baltimore, MD: USENIX Association, Aug 2018, Retrieved 10 Dec 2020. URL: https://www.usenix.org/conference/foci18/presentation/dunna.

[21] B. D., S. N., R. L., and N. V. "Poking a Hole in the Wall: Efficient Censorship-Resistant Internet Communications by Parasitizing on WebRTC." In: New York, NY, USA: Association for Computing Machinery, 2020, Retrieved 10 Dec 2020. ISBN: 9781450370899. DOI: 10.1145/3372297.3417874. URL: https://doi.org/10.1145/3372297.3417874.

[22] F. D. "Threat modeling and circumvention of Internet censorship." In: PhD thesis EECS Department University of California Berkeley, 2017 , Retrieved 9 Jul 2020.

[23] G. D. "Shining a Light on Policing of the Dark Web: An Analysis of UK Investigatory Powers." In: *The Journal of Criminal Law* 84.5 (2020, Retrieved 28 June 2020), pp. 407–426. DOI: 10.1177/0022018320952557. eprint: https://doi.org/10.1177/0022018320952557. URL: https://doi.org/10.1177/0022018320952557.

[24] R. D. and G. K. "One fast guard for life (or 9 months)." In: *In HotPETs*. Jul 2014, Retrieved 30 June 2020.

[25] R. D., N. M., and P. S. "Tor: The Second-Generation Onion Router." In: Proceedings of the 13th USENIX Security Symposium August 9-13 2004 San Diego CA USA: USENIX, 2004, Retrieved 1 June 2020, 303––320.

[26] T. D. "The Nano Age of Digital Immunity Infrastructure Fundamentals and Applications: The Intelligent Cyber Shield for Smart Cities." In: Rocky (2017): CRC Press, Retrieved 6 June 2020, pp. 210–211. ISBN: 978-1-351-68287-9.

[27] E. E., K., and D. K. "Protecting Privacy Using k-Anonymity." In: *Journal of the American Medical Informatics Association* 15.5 (2008, Retrieved 28 June 2020), pp. 627–637. ISSN: 1067-5027. DOI: 10.1197/jamia.M2716. eprint: https://academic.oup.com/jamia/article-pdf/15/5/627/21382979/15-5-627.pdf. URL: https://doi.org/10.1197/jamia.M2716.

[28] D. F., C. L., R. H., P. W., and V. P. "Blocking-resistant communication through domain fronting." In: PoPETs 2015.2, 2015, Retrieved 10 June 2020, 46––64. DOI: 10.1109/PERCOM.2009.4912774. URL: http://www.degruyter.com/view/j/popets.2015.2015.issue-2/popets-2015-0009/popets-2015-0009.xml.

[29] S. F. and E. W. "The use of TLS in Censorship Circumvention." In: (2019, Retrieved 27 June 2020). URL: https://www.freehaven.net/anonbib/papers/ndss2019_03B-2-1_Frolov_paper.pdf.

[30] *Format-Transforming Encryption (FTE)*. Retrieved 9 Set 2021. URL: https://fteproxy.org/.

[31] M. G. *Acceptable Jitter and Latency for VoIP: Everything You Need to Know*. Retrieved 10 Jul 2021. 20 Dec 2018. URL: https://getvoip.com/blog/2018/12/20/acceptable-jitter-latency.

[32] *GoodBadISPs*. Retrieved 16 April 2021. URL: https://gitlab.torproject.org/legacy/trac/-/wikis/doc/GoodBadISPs.

[33] *Google Research Blog: Speed Matters*. Retrieved 10 Nov 2020. URL: https://research.googleblog.com/2009/06/speed-matters.html.

[34] H. and J. "Breaking Through the Ambivalence: Journalistic Responses to Information Security Technologies." In: *Digital Journalism* (August 2019, Retrieved 27 June 2020), pp. 1–19. DOI: 10.1080/21670811.2019.1653207.

[35] A. H., G.C., and N. B. "Cirripede: Circumvention Infrastructure using RouterRedirection with Plausible Deniability." In: CCS '11: Proceedings of the 18th ACM conference on Computer and communications security: CCS, October 2011, Retrieved 1 May 2020, 187—200. DOI: https://doi.org/10.1145/2046707.2046730. URL: https://people.cs.umass.edu/~amir/papers/CCS11-Cirripede.pdf.

[36] A. H., T. R., N. B., and A. C. "I want my voice to be heard: IP over Voice-over-IP for unobservable censorship circumvention." In: 20th Annual Network and Distributed System Security Symposium NDSS 2013 San Diego California USA February 24-27: The Internet Society, 2013, Retrieved 15 May 2020. URL: http://internetsociety.org/doc/i-want-my-voice-be-heard-ip-overvoice-over-ip-unobservable-censorship-circumvention.

[37] J. H. and T. B. "Guidelines for creation, selection, and registration of an Autonomous System (AS)." In: IETF, Retrieved 25 June 2020, sec. 3. DOI: 10.17487/RFC1930.RFC1930. URL: https://tools.ietf.org/html/rfc1930.

[38] J. H. and G. D. "k-fingerprinting: A robust scalable website fingerprinting technique." In: *25th USENIX Security Symposium*. Austin, TX, USA, 2016 , Retrieved 9 Mar 2021, pp. 1187–1203.

[39] J. H. and A. H. "CacheBrowser: Bypassing Chinese Censorship without Proxies Using Cached Content." In: *CCS '15*. 2015, Retrieved 6 Jul 2020.

[40] R. H., S. P., H. S., R. J., J. A., E. C., B. L., M. L., B. L., and J. W. "Measurement and analysis of child pornography trafficking on p2p networks." In: International Conference on World Wide Web, 2013, Retrieved 21 Feb 2021.

[41] Z. H. and H. A. "Practical Censorship Evasion Leveraging Content Delivery Networks." In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. New York NY USA: Association for Computing Machinery, 2016, Retrieved 6 Jul 2020, 1715–1726. ISBN: 9781450341394. DOI: 10.1145/2976749.2978365. URL: https://doi.org/10.1145/2976749.2978365.

[42] M. I. and H. M. "SkypeMorph: Protocol Obfuscation for Censorship Resistance." In: Computer Science University of Waterloo Ontario Canada, 2013, Retrieved 11 May 2020. URL: https://uwspace.uwaterloo.ca/handle/10012/7262.

[43] S. I., L. T., and A. T. "Routing monitoring." In: May 2010, Retrieved 3 Junly 2020. URL: https://patentimages.storage.googleapis.com/71/99/e8/29637c37783437/US7710885.pdf.

[44] *Internet censorship circumvention*. Retrieved 1 May 2020. URL: https://en.wikipedia.org/wiki/Internet_censorship_circumvention.

[45] B. J. and H. A. "How China Detects and Blocks Shadowsocks, Proceedings of the ACM Internet Measurement Conference." In: IMC '20. New York, NY, USA: Association for Computing Machinery, 2020, Retrieved 8 April 2021, 111–124. ISBN: 9781450381383. DOI: 10.1145/3419394.3423644. URL: https://doi.org/10.1145/3419394.3423644.

[46] G. J., J. R., and H. N. "How Low Can You Go: Balancing Performance with Anonymity in Tor." In: August 2013, Retrieved 28 June 2020, pp. 164–184. ISBN: 978-3-642-39076-0. DOI: 10.1007/978-3-642-39077-7_9.

[47] J. J., J. A., D. A., B. N., and C. M. "Defending tor from network adversaries: A case study of network path prediction." In: Privacy Enhancing Technologies, Jun 2015, Retrieved 30 June 2020.

[48] K., N. A., R. M., R. I., and S. J. "Anonymity with Tor: A Survey on Tor Attacks." In: arXiv, Sep 2020, Retrieved 26 Feb 2021. URL: https://arxiv.org/pdf/2009.13018.pdf.

[49] J. K., D. E., A. J., C. J., G. L., D. M.s, and W. S. "Decoy Routing: Toward Unblockable Internet Communication." In: *USENIX Association* (2011). Retrieved 1 May 2020. URL: https://www.usenix.org/legacy/event/foci11/tech/final_files/Karlin.pdf.

[50] S. K., G. G., and B. V. "How India Censors the Web." In: *12th ACM Conference on Web Science*. WebSci '20. Southampton, United Kingdom: Association for Computing Machinery, 2020, Retrieved 10 Set 2021, 21–28. ISBN: 9781450379892. DOI: 10.1145/3394231.3397891. URL: https://doi.org/10.1145/3394231.3397891.

[51] S. L., M. S., and N. H. "Facet: Streaming over Videoconferencing for Censorship Circumvention." In: Proceedings of the 13th Workshop on Privacy in the Electronic Society WPES 2014 Scottsdale AZ USA November 3: ACM, 2014 , Retrieved 16 May 2020, 163—172.

[52] Z. L., S. H., and D. L. "DeTor: Provably Avoiding Geographic Regions in Tor." In: Proceedings of the 26th USENIX Security Symposium, August 2017, Retrieved 30 June 2020, p. 4. ISBN: 978-1-931971-40-9. URL: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/li.

[53] A. M., B. K., and G. I. "Enhancing Tor's Performance Using Real-Time Traffic Classification." In: *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. CCS '12. New York NY USA: Association for Computing Machinery, 2012, Retrieved 28 June 2020, 73––84. ISBN: 9781450316514. DOI: 10.1145/2382196.2382208. URL: https://doi.org/10.1145/2382196.2382208.

[54] A. M. and Y. C. M. V. "Lastor: A low-latency as-aware tor client." In: IEEE Symposium on Security and Privacy, May 2012, Retrieved 10 June 2020.

[55] D. M., M.S., and A. K. *SOCKS Proxy Primer: What Is SOCKs5 and Why Should You Use It?* Retrieved 29 Nov 2020. 2019. URL: https://securityintelligence.com/posts/socks-proxy-primer-what-is-socks5-and-why-should-you-use-it/.

[56] H. M., B. L., M. D., and I. G. "SkypeMorph: protocol obfuscation for Tor bridges." In: the ACM Conference on Computer and Communications Security CCS'12 Raleigh NC USA October 16-18: ACM, 2012 , Retrieved 10 May 2020, 97––108. DOI: https://doi.org/10.1145/2382196.2382210. URL: https://dl.acm.org/doi/10.1145/2382196.2382210.

[57] N. M., B. A., and H. A. "DeepCorr: Strong Flow Correlation Attacks on Tor Using Deep Learning." In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. New York NY USA: IEEE, Oct 2018 , Retrieved 25 June 2020, 1962–1976. DOI: 10.1109/MILCOM.2018.8599680. URL: https://doi.org/10.1145/3243734.3243824.

[58] R. M., A. H., and V. S. "CovertCast: Using Live Streaming to Evade Internet Censorship." In: PoPETs 2016.3, 2016 , Retrieved 16 May 2020, 212–225. URL: http://www.degruyter.com/view/j/popets.2016.2016.issue-3/popets-2016-0024/popets-2016-0024.xml.

[59] S. M., C. T., and J. C. "Dissecting tor bridges: A security evaluation of their private and public infrastructures." In: Network and Distributed System Security Symposium (NDSS), Feb 2017, Retrieved 25 June 2020.

[60] Y. M., K. W., and D. A. "Understanding the use of circumvention tools to bypass online censorship." In: 18 (August 2014, Retrieved 27 June 2020), pp. 837–856. DOI: https://doi.org/10.1177/1461444814548994. URL: https://journals.sagepub.com/doi/abs/10.1177/1461444814548994.

[61] *Measuring Performance with the RAIL model*. Retrieved 10 Nov 2020. URL: https://developers.google.com/web/fundamentals/performance/rail.

[62] G. A. S. N. "Privacy preservation in big data using k-anonymity algorithm with privacy key." In: International Journal of Computer Applications, Nov 2016, Retrieved 2 Junly 2020.

[63] M. N., H. Z., and A. H. "The Waterfall of Liberty: Decoy Routing Circumvention that Resists Routing Attacks." In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security CCS 2017 Dallas TX USA October 30 - November 03: CCS, Retrieved 8 May 2020, 2037––2052. ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3134075. URL: http://doi.acm.org/10.1145/3133956.3134075.

[64] R. N., O. S., A. Z., P. G., and M. S. "Measuring and mitigating AS-level adversaries against Tor." In: *CoRR* abs/1505.05173 (2015, Retrieved 27 June 2020), pp. 837–856. URL: http://arxiv.org/abs/1505.05173.

[65] *obfs2 (The Twobfuscator)*. Retrieved 20 June 2020. URL: https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/tree/doc/obfs2/obfs2-protocol-spec.txt.

[66] *obfs3 (The Threebfuscator)*. Retrieved 20 June 2020. URL: https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/tree/doc/obfs3/obfs3-protocol-spec.txt.

[67] *obfs4 (The Obfourscator)*. Retrieved 20 June 2020. URL: https://gitweb.torproject.org/pluggable-transports/obfs4.git/tree/doc/obfs4-spec.txt.

[68] K. P., S. E., and T. S. "Marionette: A Programmable Network Traffic Obfuscation System." In: 24th USENIX Security Symposium USENIX Security 15 Washington D.C. USA August 12-14: USENIX Association, 2015, Retrieved 12 May 2020, 367––382.

[69] W. P., K. R., M. M., H. M., S. S., L. S., and W. E. "Spoiled Onions: Exposing Malicious Tor Exit Relays." In: vol. 8555. Jul 2014, Retrieved 29 June 2020. DOI: 10.1007/978-3-319-08506-7_16.

[70] *Patriot Act*. Wikipedia. Retrieved 1 May 2021. URL: https://en.wikipedia.org/wiki/Patriot_Act.

[71] T. T. Project. Retrieved 10 Jan 2021. URL: https://www.torproject.org/.

[72] T. project. *Tor: Onion Service Protocol*. Retrieved 17 Mar 2021. URL: https://2019.www.torproject.org/docs/onion-services.

[73] J. R., T. M.w, and H. N. "Privacy-Preserving Dynamic Learning of Tor Network Traffic2." In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS '18. New York NY USA: Association for Computing Machinery, 2018, Retrieved 24 June 2020, 1944––1961. ISBN: 9781450356930. DOI: 10.1145/3243734.3243815. URL: https://doi.org/10.1145/3243734.3243815.

[74] M. R., P. S., and D. M. "Anonymous connections and onion routing." In: *IEEE Journal on Selected Areas in Communications* 16.4 (May 1998, Retrieved 27 June 2020), pp. 482–494. DOI: 10.1109/49.668972.

[75] R. R., R. S., M. B., V. O., L. E., and A. E. "Decentralized control: a case study of Russia." English. In: *Proceedings, 2020 Network and Distributed System Security Symposium*. Network and Distributed Systems Security Symposium 2020 , NDSS 2020 ; Conference date: 23-02-2020 Through 26-02-2020. Internet Society, 2020, Retrieved 10 Set 2021, pp. 1–18. DOI: 10.14722/ndss.2020.23098.

[76] S. R., T. C., D. C., F. J., and H. P. "Unraveling an old cloak: K-anonymity for location privacy." In: ACM Workshop on Privacy in the Electronic Society, Oct 2010, Retrieved 3 Jul 2020.

[77] *Relay Search*. Retrieved 16 April 2021. URL: https://metrics.torproject.org/rs.html#search/ovh.

[78] rfc3546. *Transport Layer Security (TLS) Extensions*. Retrieved 29 June 2020. URL: https://tools.ietf.org/html/rfc3546.

[79] C. S., B. C., and S. C. "IP Covert Channel Detection." In: *ACM Trans. Inf. Syst. Secur.* 12.4 (2009, Retrieved 30 June 2020). ISSN: 1094-9224. DOI: 10.1145/1513601.1513604. URL: https://doi.org/10.1145/1513601.1513604.

[80] C. S., P. G., P. M., and K. A. "Detecting Traffic Snooping in Tor Using Decoys." In: 2011, Retrieved 29 June 2020, pp. 222–241. DOI: 10.1007/978-3-642-23644-0_12.

[81] J. S., M. A., and N. C. "Pitfalls of covert channel censorship circumvention." In: Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security(Berlin Germany): ACM, 2013 , Retrieved 15 May 2020, 361—372.

[82] K. S. and N. C. "Measuring the longitudinal evolution of the online anonymous marketplace ecosystem." In: USENIX Security Symposium, 2015, Retrieved 22 Feb 2021.

[83] P. S. and L. S. "Generalizing Data to Provide Anonymity when Disclosing Information." In: Mar 1998, Retrieved 4 Junly 2020. URL: http://www.sdl.sri.com/papers/344/.

[84] R. S. *To Block or not to Block, Crossing Walls in a Censored Internet*.

[85] T. S. and J. M. "A hybrid machine learning approach to network anomaly detection." In: *Information Sciences* 177.18 (2007, , Retrieved 27 June 2020), pp. 3799–3821. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2007.03.025. URL: http://www.sciencedirect.com/science/article/pii/S0020025507001648.

[86] *ScrambleSuit*. Retrieved 9 Set 2021. URL: https://www.cs.kau.se/philwint/scramblesuit/.

[87] *Servers Bridges by IP version*. Retrieved 9 Set 2021. URL: https://metrics.torproject.org/bridges-ipv6.html.

[88] *StarTrinity software company*. Retrieved 9 Jul 2021. URL: http://startrinity.com/.

[89] E. T., B. K., A. M., D. R., and G. I. "Changing of the Guards: A Framework for Understanding and Improving Entry Guard Selection in Tor." In: *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*. WPES '12. New York NY USA: Association for Computing Machinery, 2012, Retrieved 30 June 2020, 43—54. ISBN: 9781450316637. DOI: 10.1145/2381966.2381973. URL: https://doi.org/10.1145/2381966.2381973.

[90] M. T. *Stunnel*. Retrieved 6 June 2020. 28 Dec 2002. URL: https://www.stunnel.org/index.html.

[91] *Tor metrics*. Tor project. Retrieved 6 June 2020. URL: https://metrics.torproject.org/userstats-relay-country.html.

[92] *Tor Project: A child garden of pluggable transports*. Retrieved 29 June 2020. URL: https://trac.torproject.org/projects/tor/wiki/doc/AChildsGardenOfPluggableTranspor

[93] *Tor Project: How to report bad relays*. Retrieved 29 June 2020. Jul 2014. URL: https://blog.torproject.org/how-report-bad-relays.

[94] *Tor Project: Reporting bad relays*. Retrieved 24 June 2020. Sep 2018. URL: https://trac.torproject.org/projects/tor/wiki/doc/ReportingBadRelays.

[95] *Tor Project: Tor directory specification*. Retrieved 25 June 2020. URL: https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt.

[96] *Tor Project: Tor faq*. Retrieved 24 June 2020. 2019. URL: https://2019.www.torproject.org/about/overview.html.en.

[97] *Tor Project: Tor pluggable transports*. Retrieved 25 June 2020. URL: https://2019.www.torproject.org/docs/âĂĺpluggable-transports.

[98] *Tor throughput metrics*. Tor project. Retrieved 14 April 2020. URL: https://metrics.torproject.org/onionperf-throughput.html.

[99] *Tortunnel*. Retrieved 20 June 2020. URL: https://github.com/moxie0/tortunnel.

[100] G. V., E. C., R. S., K. C., and L. S. "Network Traffic Obfuscation: An Adversarial Machine Learning Approach." In: *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*. Los Angeles CA USA: IEEE, 2018, Retrieved 25 June 2020. DOI: 10.1109/MILCOM.2018.8599680.

[101] C. W., S. C., and F. M. "Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis." In: Proceedings of the Network and Distributed System Security Symposium NDSS 2009 San Diego California USA 8th February - 11th February 2009, 2009, Retrieved 9 May 2020. URL: http://www.isoc.org/isoc/conferences/ndss/09/pdf/14.pdf.

[102] E. W., S. W., I. G., and J.H. "Telex: Anticensorship in the Network Infrastructure." In: *USENIX Association* (2011). Retrieved 2 May 2020. URL: https://www.usenix.org/legacy/events/sec11/tech/full_papers/Wustrow.pdf.

[103] G. W. "Going dark: Terrorism on the dark web." In: Studies in Conflict and Terrorism, 2016, Retrieved 27 Feb 2021.

[104] *XGBoost Documentation*. Retrieved 10 Jul 2020. URL: https://xgboost.readthedocs.io/en/latest/.

[105] S. Y., E. A., V. L., L. O., R. J., C. M., and M. P. "RAPTOR: Routing attacks on privacy in tor." In: USENIX Security Symposium Washington D.C., Aug 2015, Retrieved 8 May 2020.

[106] G. Z. and U. H. "A distributed k-anonymity protocol for location privacy." In: *2009 IEEE International Conference on Pervasive Computing and Communications*. Galveston TX USA, 2009, Retrieved 28 June 2020, pp. 1–10. DOI: 10.1109/PERCOM.2009.4912774.

[107] Z. Z. and M. S. "Bypassing Tor Exit Blocking with Exit Bridge Onion Services." In: Proc. of CCS 2020 – ACM SIGSAC Conference on Communications and Computer Security, 2020, Retrieved 20 Feb 2021.