Anonymous communication in Local Area Networks

Tim Christian Sauer
1162083

Master Thesis
VS-2025-10M

Examined by
Prof. Dr. rer. nat. Frank Kargl
Prof. Dr. Steffen Wendzel

Supervised by
M.Sc. Juri Dispan

Institute of Distributed Systems
Faculty of Engineering, Computer Science and Psychology
Ulm University

May 28, 2025

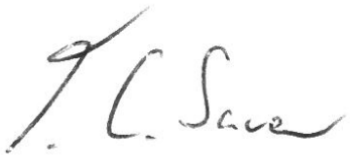
Issued: May 28, 2025

I hereby declare that this thesis titled:

Anonymous communication in Local Area Networks

is the product of my own independent work and that I have used no sources or materials other than those specified. The passages taken from other works, either verbatim or paraphrased in the spirit of the original quote, are identified in each individual case by indicating the source.
I further declare that all my academic work was written in line with the principles of proper academic research according to the official "Satzung der Universität Ulm zur Sicherung guter wissenschaftlicher Praxis" (University Statute for the Safeguarding of Proper Academic Practice).

Ulm, May 28, 2025



Tim Christian Sauer, student number 1162083

ABSTRACT

In an age where everything is increasingly interconnected, more and more sensitive data is generated. It is essential that this data is protected from misuse. One aspect of this is that metadata of communications is not publicly visible or able to be uncovered by an attacker. This includes information on communication partners or any data that allows conclusions on them. While the problem of privacy-preserving communication on the Internet is addressed by a magnitude of work, there is little contribution to tackle this challenge in local networks. This thesis gives an overview of existing solutions for anonymous communication in general. Based on this related work, multiple directions for potential solutions in LANs are drafted. The solution of implementing a local Anonymous Communication Network (ACN) is chosen. Using The Onion Router (TOR), which was originally designed for the Internet, an anonymous LAN is set up and evaluated. As a local network does not provide the decentralized infrastructure of the Internet, we assess the provided anonymity of a local TOR deployment. For this case we re-evaluate the feasibility of existing attacks against TOR in the context of a local network. We are able to show that the privacy guarantees provided by TOR hold in a local deployment. However, performance measurements show the magnitude of TOR's known performance limitations: TOR heavily underestimates available bandwidth. We are able to show that TOR's congestion control algorithm is responsible for this problem. Furthermore, the load on the network is unevenly distributed over relays. As a result of these limitations, we conclude that TOR is not an adequate solution for providing anonymity in local networks.

CONTENTS

1 INTRODUCTION

The Internet is one of the most prominent technology of the 21st century, connecting people around the world. Its building blocks are networks, which is why it is also called a network of networks. One type of these smaller networks that make up the Internet is the *Local Area Network* (*LAN*). LANs are used to connect devices in homes, universities, or businesses.

As each of us communicates and sends sensitive information (e.g., health information, online banking information, etc.) over networks every day, improving their privacy has become a critical concern. Over the years, a variety of methods has been developed to enhance privacy in networks. These methods include encryption techniques, anonymization protocols, secure communication channels, and more. While the challenge of confidential information transfer is solved by end-to-end encryption, where data remains encrypted on its way from sender to receiver, open questions remain. One of these questions is how network users can communicate with each other without anyone being able to determine who is communicating with whom. While an attacker may not be able to read the encrypted content of a connection, the fact that two people are communicating can get them into trouble. Consider an investigative journalist communicating with a whistleblower: If their connection is leaked, it could cause serious trouble for both of them. Motivated by this challenge, solutions such as TOR [24], GNUnet [31], and other *Anonymous Communication Networks* (ACN), have been developed. However, these solutions only address this problem for global communication on the Internet. They rely on the decentralized nature of the Internet: Because there are many parties involved, distributing information across many of these parties makes it impossible for a single attacker to de-anonymize users.

A second motivation for this thesis comes from the field of cybersecurity and might seem unrelated at first glance. When performing an assessment, *Red Teams* need to remain undetected after the initial infiltration of an infrastructure. Furthermore, they are often challenged to exfiltrate sensitive data from the internal network of their objective. While these two disciplines seem far apart thematically, what helps Red Teams stay 'under the radar' might be similar to what helps users communicate privately within LANs.

1.1 THESIS SCOPE

We now describe the scope of this thesis, which is to answer the following question:

How can it be ensured that devices in a LAN can communicate privately without any single attacker knowing which two (or more) devices communicate with each other?

More specifically, this is achieved by answering the following research questions:

1. Which existing techniques can be adapted to enable anonymous communication in a LAN as well as help red teamers to stay undetected after infiltrating a network?
2. What can a prototype of such a LAN look like, and how can it be implemented?
3. Does this prototype solve the problem sufficiently in terms of performance and privacy guarantees?

1.2 THESIS STRUCTURE

The structure of this thesis corresponds to the process of our conducted research. In Chapter 2 we give an introduction to terms and concepts that are central to this thesis. To set a basis for our contributions, we need to gain insights on the topics we focus on. To tackle this task, we conduct interviews with experts from the fields of Red Teaming, as well as with administrators who maintain local networks on a daily basis. Methods, results, and discussion of these interviews are given in Chapter 3. Following the interviews, we come to the conclusion that not both topics can be pursued. We therefore conclude to focus on anonymous communication in LANs. Before we start implementing a solution, we need to define against which type of attacker our solution should protect and how we define the term *anonymity*. This is done in Chapter 4. In order to get an overview of how communication in networks can be anonymized in general, we introduce and examine existing solutions in Chapter 5. Most of the discussed ACNs are designed for usage on the Internet. Nevertheless, we argue that concepts from these infrastructures can be transferred to our problem. With the knowledge from existing solutions and expert interviews, we identify two categories of solutions. We describe these categories in Chapter 6. Subsequently, we discuss advantages and disadvantages of these fundamentally different solutions and choose one of them as our concept. In Chapter 7, we propose a setup for an anonymous LAN, based on our chosen concept. Additionally, we introduce our further methodology for assessing our solution. The assessment takes two aspects into account: while the primary goal is to provide anonymity for clients in the LAN, a sufficient performance of the envisioned solution is important, too. In a first step, we simulate TOR in a university-sized LAN to see how it behaves under the load of several thousands of clients. Furthermore, we examine the privacy guarantees of our proposed solution. In Chapter 8 we discuss details of deploying TOR in a LAN. Also, further implementational details on our simulations are given. Finally, we present the results of applying our methodology in Chapter 9, discuss them in Chapter 10, and conclude this thesis in Chapter 11.

2 BACKGROUND

This chapter gives an introduction to terms and concepts that are central to this thesis.

2.1 RED TEAMING

Red Teaming is a type of assessment that is conducted by security experts to assess the security of an IT infrastructure. While traditional IT security assessments focus on the security of single components, Red Teaming targets the entire infrastructure of an organization/company/etc. and aims to achieve a set goal (e.g., exfiltrating sensitive data, such as passwords, or accessing sensitive infrastructure, such as backup systems). This also includes the people involved in processes, for example, a company's *Blue Team*, which are responsible for reacting to attacks on the infrastructure. Often, the goal of a Red Teaming assessment is achieved by exploiting multiple vulnerabilities in an infrastructure, ranging from social engineering up to the takeover of clients. During the entire assessment, the Red Team aims to stay undetected to reach their goal. One major question this raises is how to exfiltrate data from the subject's network to the outside world. In order to circumvent Intrusion Detection Systems, client-side antivirus software, and further measures, undetected communication channels have to be established. In the remainder of this thesis, we call these channels *covert communication channels*.

2.2 ANONYMITY

In literature on ACNs, several definitions of the term *anonymity* exist. Mazurczyk et al. classify these definitions/types into three distinct categories [39]:

1. The first of these three types is hiding the identities of communication parties. This means, the identities of the sender or receiver (or even both parties) are hidden from the rest of the network.

2. The second type is hiding the fact that a communication is taking place (hiding the communication process).

3. The third type focuses on hiding the communication content. This is achieved by applying cryptographic techniques to the content of a message.

Kuhn et al. go one step further and define so-called *privacy notions*, which formally define more types of anonymity, all ranging between the first two types given in the above-mentioned list [44]. These privacy notions allow us to classify existing ACNs. Furthermore, their work proposes a hierarchy of privacy notions, which allows us to directly compare the anonymity provided by existing ACNs.

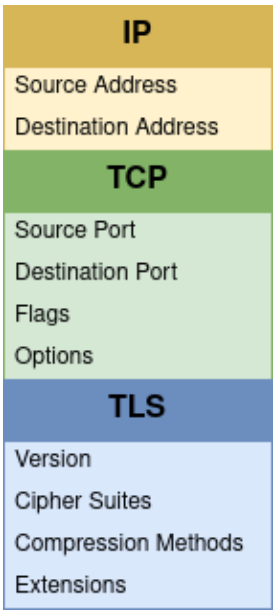


Figure 2.1: Identifying metadata in the stack of a TLS-handshake.

2.3 ANONYMITY IN NETWORKS

In communication that uses the *Internet Protocol* (IP), the sender and receiver of an IP packet can trivially be identified by their IP addresses. However, not only IP addresses can identify users, or more specifically their devices. Often, a packet's entire stack contains multiple attributes. Combining these attributes allows drawing conclusions about the communication parties. Consider the example of a TLS-handshake, shown in Figure 2.1 (only IP and higher layers are shown). We manually inspect the packet stack of a recorded handshake and pick out exemplary attributes that show a high variation in their values. In combination, these values can help to identify their communication parties. Therefore, simply rewriting or removing IP addresses is not sufficient for anonymizing communication parties. The goal of ACNs is therefore to hide not only the IP addresses of sender and receiver but also this identifying metadata and therefore increase the level of anonymity for their users.

2.4 PUBLIC KEY CRYPTOGRAPHY

The concept of *public key cryptography*, also known as *asymmetric cryptography*, can be used by a sender to encrypt a message to a ciphertext so that only the intended sender can decrypt the ciphertext and read the message. The principle is based on a pair of keys: the first part is the public key, which can be made available to anyone. The second part is the private key, which must be kept secret by its owner. Consider Alice and Bob and the key pair Key_{Bob}, which consists of the private key $Key_{Bob_{priv}}$ and $Key_{Bob_{pub}}$. $Key_{Bob_{pub}}$ is known to Alice (e.g., because Bob posted his public key on his website), and $Key_{Bob_{priv}}$ is kept secret by Bob. Alice now wants to send a confidential message, m, to Bob without leaking it to anyone sniffing on their communication. Therefore, she encrypts m

to a ciphertext c, where $c = encrypt(m, Key_{Bob_{pub}})$. She then sends c to Bob. Bob, upon receiving c, decrypts it to m, where $m = decrypt(c, Key_{Bob_{priv}})$.

2.5 SYMMETRIC CRYPTOGRAPHY

In contrast to public key cryptography, *symmetric cryptography* is not based on a key pair. It uses one key that must only be known to parties that should be able to learn a plaintext that was encrypted to a ciphertext. Thus, any communication party involved must know the symmetric key. The advantage of symmetric cryptography over public key cryptography is its higher performance. Consider Alice and Bob and a shared symmetric key K. Alice now wants to send a message m to Bob via an insecure communication channel. She encrypts m to a ciphertext c where $c = encrypt(m, K)$. Subsequently, she sends c to Bob. Bob, who also possesses K, can proceed with decrypting c to m, where $m = decrypt(c, K)$.

2.6 LOCAL AREA NETWORK

Typically, a LAN is defined by the geographical area it covers. In contrast to a *Metropolitan Area Network (MAN)*, its area is more limited. A LAN typically covers a home network, a company network, a campus/university network, etc. In this thesis, the aspect of size is not the key criterion. We rather define a LAN by its technical details. The primary characteristic of a LAN, as defined by us, is that it is not segmented. This thesis does not address the given challenge over several VLANs through firewalls or give a solution that guarantees anonymous communication over the borders of network segments. This is for the following reason: One purpose of a segmented network is, that devices in these different segments are isolated from each other. Additionally, firewalls can be deployed to enforce that only defined sets of devices are allowed to communicate with each other. For this purpose, devices are usually identified by their IP or MAC addresses. However, the goal of anonymous communication is to *prevent* identification of devices. Therefore, we leave the challenge of anonymity in more complex LANs for future research.

2.7 VPN

A *Virtual Private Network* (VPN) allows to establish a secure connection between a remote host and a network that are connected through an untrusted network, in which nodes might eavesdrop on traffic. By routing all traffic through an encrypted tunnel, an eavesdropper is only able to see the IP addresses of the remote device and the VPN server. Hence, all destinations that are contacted by the remote device are only visible to the VPN server. This enables users to communicate privately, even in an 'insecure' network. This technology is especially interesting for us because it protects against our attacker model, which we define in Chapter 4: If an attacker observes packets encrypted in a VPN tunnel at some point in the network, e.g. a switch or router, they will not be able to see any characteristics of the packet. Since the payload of the packets is encrypted, the attacker is only able to learn its unencrypted metadata, i.e. the IP address of

the client and the VPN server. However, this approach brings some drawbacks with it:

- The VPN server has full access to all data of all users.
- The VPN server must be in the LAN; otherwise, local services are not accessible.
- Client-side configuration is required.

3 EXPERT INTERVIEWS AND IMPLICATIONS

In order to understand the challenges of red teamers and network administrators, we conduct interviews with them. First, we present the methodology our interviews follow. Subsequently, we present the results. Finally, conclusions for the further process of our work are drawn. Administrators are responsible for campus networks of three German universities (Tübingen, Karlsruhe, and Ulm). Red teamers are experts from an international IT security consultancy.

3.1 METHODOLOGY

In this section we describe the structure of the interviews and the intents behind the questions. From the insights of these interviews, we derive further directions for our work.

3.1.1 RED TEAMERS

In our expert interview, red teamers were asked the following questions:

1. How do blue teams/defense systems try to detect attacks/a red team?
2. How do red teamers communicate out of an infiltrated network, undetected?
3. What are the challenges in communicating out of an infiltrated network?

3.1.2 ADMINISTRATORS

In the interview with administrators the following questions were asked:

1. Which kind of data about users is stored to maintain the network, e.g. for debugging or troubleshooting?
2. If this data is taken from you, would this still allow operation of the network?
3. What are you already doing to protect the user's privacy, and what are the problems?

3.2 RESULTS

In the following, the insights from expert interviews with red teamers and Administrators are given.

3.2.1 RED TEAMERS

In this section we present the insights from interviews with red teamers. The structure of these interviews is presented in Subsection 3.1.1.

Interviews with experts from the field of Red Teaming gave insights into their attack strategies, as well as into defense techniques used by Blue Teams. Blue Teams often use a combination of multiple technologies to detect intrusions into their network. These solutions include endpoint protection tools such as Windows Defender, Sophos, and more. Often, data from these tools is collected on a central platform such as an ELK stack[1]. On these platforms, Blue Teams then analyze data and aim to detect attacks. In rare cases, protection is outsourced to external providers who then take responsibility for attack detection. What all solutions have in common is, that deep packet inspection is not applied to detect attacks. While some providers claim to apply machine learning models on packet data, experience shows that even these techniques fail to detect intrusions reliably. One commonly used method to evade detection mechanisms is domain fronting. In domain fronting, communication is performed via a trusted system such as a Content Delivery Network (CDN). In communication to the CDN, a connection to a malicious destination, such as a Command-and-Control (C2) server, is hidden. As the payload is encrypted, detection systems only see the legitimate communication to a trustworthy, reputable, and commonly used CDN, which is also used in ordinary user traffic.

Red teamers also reported, that communication from infiltrated networks is held to a minimum as they never know the exact detection mechanisms. Only if communication to the outside is absolutely necessary it is performed. Covert communication channels are usually chosen from the set of commonly used, benign channels in the infiltrated network. This shows that in Red Teaming, even the fact that communication to the outside world takes place can render the whole assessment unsuccessful. Therefore, the problem in Red Teaming falls into the second category of information hiding: hiding the communication process (see Section 2.2 for details).

3.2.2 ADMINISTRATORS

In this section we present the insights from interviews with administrators. The structure of these interviews is presented in Subsection 3.1.2.

Interviews with administrators of university networks provided an overview of the data stored to maintain these networks. The following data is stored for a limited amount of time:

1. DHCP Leases, including IP-address, MAC-address, hostname (if sent by the client)

2. OpenVPN logs including remote IP-address, user, internal IPv4- and IPv6-address

1 An ELK stack consists of three components: *Elasticsearch* is a distributed search engine. *Logstash* is a tool to collect log data from several sources and combine them into one place. *Kibana* is a tool for data visualization of log and protocol data. All in all, an ELK stack can be used to collect log and analysis data from several systems, combine them in one place, and analyze them there.

3. RADIUS logs for WLAN authentication, including user, MAC-address, Access-Point-ID

4. RADIUS logs for VPN authentication, including user, remote IP-address.

All of this data falls into the category of personal data. Therefore, this data is only allowed to be collected because maintaining the network requires it. This is the case when, for example, a VPN-connection must be debugged or other services affecting one specific person must be maintained. After a fixed time of one to two weeks (in the case of our interviewees), this data must be deleted. After this fixed time, no personal data is kept by the network administrators. Despite the potential privacy issues coming with this data, administrators underlined the importance of this data for debugging and monitoring purposes. Another point that was discussed in the interviews is the triviality of identifying devices that are wired to the network. When a user connects their device to a hardware network port in their office, it is obvious to an administrator which user this is about.

3.3 IMPLICATIONS OF RESULTS

In the following, we discuss the results of the conducted interviews.

3.3.1 RED TEAMERS

Insights gained in the interviews with red teamers raise the question of whether it makes sense to research solutions for red teamers. This direction is questionable, as there does not seem to be the need for new approaches to hide communication. As encountered in the interviews, choosing covert communication channels among the set of channels that are also used in legitimate traffic is sufficient. This issue is further discussed in Subsection 3.3.3.

3.3.2 ADMINISTRATORS

Interviews with administrators reveal into which category of information hiding the problem of providing anonymity in a network falls. When looking into legal regulations, European law defines 'personally identifiable information' in the following way: *personal data means any information relating to an identified or identifiable natural person; an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person.* [73]. We therefore argue that anonymity is reached when all data in a network that falls into this category is hidden from a potential attacker. This can be achieved by encryption or any other method that prevents identification based on these identifiers. The crucial point in this legal matter is that no person can be identified. Thus, anonymity in networks falls into the first category of information hiding: hiding the identities of communication parties. This observation is one of the foundations for our definition of the term 'anonymity'. Our formal definition is provided in Section 4.2.

Furthermore, insights from the interviews show the difficulty of finding a balance between anonymity and practicability. It appears to be a questionable approach to withdraw the currently stored data from administrators, as this kind of data seems to be essential for maintenance work and debugging. Instead, we decide to focus on an attacker who has access to less or at least less sensitive data. This brings us away from the administrator as an attacker who has access to unencrypted traffic to a more realistic attacker model. We present this attacker model in Section 4.1.

3.3.3 GENERAL OBSERVATIONS

At first glance, the problems in Red Teaming and anonymity in LANs are strongly related. However, the conducted interviews show that the two problems fall into distinct categories of information hiding. Consequently, solutions for these problems differ more than we expected initially. This also becomes clear when classifying these two problems using the approach of Kuhn et al. [44], introduced in Section 2.2. Covert communication channels fall into the category of *Communication Unobservability*. In contrast to that, anonymous communication in networks roughly falls into the category of *Sender Unobservability* (depending on the exact definition of the term 'anonymity'). Applying the author's proposed hierarchy yields that any system that provides a covert communication channel also solves the problem of anonymous communication in networks. However, implementing such a system in a LAN is infeasible [44]. Therefore, it seems an unachievable goal to find a solution that solves both problems and is usable in practice. Thus, we have to make a decision on which problem we set our focus. Pursuing both of them is beyond the scope of this thesis. Furthermore, the introduced task of hidden communication in Red Teaming does not seem to be as much of an issue as we assumed in the beginning of this thesis. Therefore, we decide to focus on research towards anonymous communication in LANs.

4 ATTACKER MODEL AND DEFINITION OF ANONYMITY

This section introduces our attacker model. Based on this attacker model, we later define three types of attackers for which we evaluate if they can break the anonymity of our given solution. Furthermore, we define the term *anonymity*. This allows us to assess in our evaluation if our solution provides the desired anonymity.

4.1 DEFINITION OF ATTACKER MODEL

In our attacker model, an adversary with limited compute power is assumed to gain access to a network node, such as a router or switch, either through administrative privileges or via a previously successful attack. This includes not only administrators but also attackers from the outside, who manage to sniff traffic at some critical point in the network. This could be a switch, router, or some other point of control. Hence, our attacker model includes all persons who are able to sniff on network traffic at some point of control in the network. However, only attackers with partial insight into network data are considered. In literature, such an attacker is also called *non-global adversary*. We explicitly do *not* consider an attacker who has insight into all data in the network. Exact parameters, e.g. how many points of control they can observe, are defined with specific attackers we consider (see Chapter 7 for details). Furthermore, our attacker model considers an attacker that is able to actively participate in the network.

In our evaluation, we assess if our proposed solution provides the desired anonymity. In this process, we describe multiple specific attackers, their capabilities, and the infrastructure they manage to compromise. These types of attackers are all based on this definition. See Chapter 7 for these specific attackers.

4.2 DEFINITION OF ANONYMITY

Until this point of the thesis, the term *anonymity* was used without further definition. In order to later assess our proposed solution, we define it now. Our approach seeks to provide *Sender-Message Unlinkability* as defined by Kuhn et al. [44]. This means that an attacker should not be able to tell which message was sent from which sender. Note that this also implies, that an attacker has *no* insight into the payload of a message. Based on the formal definition given in the work mentioned above, we can later assess our prototype (see Section 7.2). We choose this definition of anonymity because it fulfills the observation made in Subsection 3.3.2: If linking a message to a specific user is impossible, this implies that *any information relating to an identified or identifiable natural person* [73] is covered. We therefore argue that with our definition of anonymity, we successfully protect any 'personal data', as it is defined by European law, from an attacker. Furthermore, anonymity within the network should be the responsibility of the administration and not the clients. Thus, we require our solution to need as little effort as possible for clients. This also implies, that clients should not be required

to participate in our solution as an active part of it (e.g. by relaying data for other clients). Furthermore, this definition of anonymity is even stronger than the desired anonymity demanded in Section 1.1. On top of that, reaching the defined kind of anonymity seems like a realistic goal, as most solutions introduced in Chapter 5 reach the same or a similar type of anonymity [44].

With our definition of anonymity come certain criteria that a solution has to meet in order to fulfill our definition. Our definition is directly inferred from existing work [44] which also lists the criteria that are needed to reach this level of anonymity. In order to reach anonymity in the defined sense, an attacker should not be able to solve the following challenge:

Let there be two batches of messages, $M_1 = (m_{1_1}, \ldots, m_{1_n})$ and $M_2 = (m_{2_1}, \ldots, m_{2_n})$ (note that both sets have the same number of messages), with S_1 being the author of all messages in M_1 and S_2 analogous. When arbitrarily picking one of the sets and sending it through the proposed solution, the attacker observing the communication should not be able to tell if the set belonging to S_1 or the set of S_2 was sent. Intuitively, this means that an attacker should not be able to tell if an arbitrary set of messages was sent by S_1 or person S_2.

5 ANONYMOUS COMMUNICATION NETWORKS

In order to build a solution, as proposed in Section 1.1, we present a number of prominent ACNs, as well as conceptual constructs, aiming to provide anonymity in networks. In their survey from 2016, Shirazi et al. define five categories in which ACNs can be divided [76]:

1. Mixnet-based protocols
2. Onion Routing protocols
3. DHT-based protocols
4. DCNets
5. Miscellaneous

Based on *one* chosen candidate from each category, we introduce existing technologies. Subsequently, we discuss which concepts could be adapted for our solution.

5.1 MIXNET-BASED PROTOCOLS

Mix networks were first introduced by Chaum in 1981 [16]. This concept is based on public key cryptography and so-called *mixes*. The goal of this form of network is to anonymize the sender and receiver of a message so that an attacker can not observe who is communicating with whom. Consider two correspondents, x and y, and some node m acting as the mix. If x wants to send a packet p to y via m, they build the following message M_{xy} with Key_w being the public key of node w, $Addr$ being their address, and N being a nonce increased security[1].

$$M_{xy} = enc_{Key_m}(N_1, enc_{Key_y}(N_0, p), Addr_y) \tag{5.1}$$

It is obvious that when only a single mix lies between x and y, that this mix knows that x communicates with y. Hence, not only one, but multiple mixes must be chained between x and y. Consider the adjusted example with an additional mix n. A message M'_{xy} sent from x to y via n and m would look like this:

$$M'_{xy} = enc_{Key_n}(N_2, enc_{Key_m}(N_1, enc_{Key_y}(N_0, p), Addr_y), Addr_m) \tag{5.2}$$

In this case neither any involved node nor an attacker can link x and y as correspondents. This is because n only knows the source node x but not the destination. However, m knows the destination y but does not know that the packet was sent by x.

In addition to these steps of encryption, the mixes always collect batches of messages, sort them in lexicographical order, and then forward them. Thus, an attacker can not identify incoming and outgoing messages by observing their

1 In fact, an attacker could learn that $X = Y$ if $K(X) = K(Y)$ held true. Therefore, a nonce is appended to the actual message.

order. Mix networks even protect against attackers that can see the all traffic in the network. However, this brings the cost of a delay until a batch of messages is full and can be shuffled.

One of the first systems that implements mix networks for a practical use case is *Mixmaster* [55]. This protocol specifies how emails are split, encrypted multiple times, and sent through a series of nodes so that an adversary can not observe the destination of an email. As emails do not require low-latency transfer, a delay induced through message shuffling is unproblematic. Danezis et al. propose *Mixminion* [22], which refines Mixmaster. Important improvements to Mixminion are to allow users to also reply to received emails anonymously. Secondly, Mixminion uses TLS to encrypt forwarded data between nodes. Additionally, there are several further features that extend performance and provide anonymity of Mixminion over Mixmaster.

Further systems also implement the approach of mix networks and set its concept into practice [69, 9, 22].

5.2 ONION ROUTING PROTOCOLS

TOR (The Onion Router) [24] is based on the concept of Onion Routing [72]. Onion Routing heavily builds on mix networks (explained in Section 5.1). As in mix networks, traffic is routed through a series of nodes (three in this case), called relays, between origin and destination. In contrast to mix networks, messages are not enqueued and shuffled by a mix node but rather forwarded directly [82]. Thus, no batches have to be filled, and therefore packets are routed without an extra delay. On the other hand, packets are not shuffled, which makes it easier for attackers to correlate traffic and break anonymity. Still, due to the multiple encryption layers, each node can only see the next and previous node, and no single instance knows both the source and destination of the traffic.

Nevertheless, TOR does not protect against an attacker who has insight into all routed traffic. Due to the decentralized nature of the Internet, this weakened anonymity still provides sufficient protection against most attackers. With this cost of anonymity comes a more practical usability, as messages can be routed without additional delay. All in all, TOR gets its security from routes that are, in their entirety, not possible to observe by an attacker [82]. Next to TOR, there exist numerous other solutions in literature that also employ Onion Routing [46, 19, 17, 35].

As our given approach, presented in Chapter 7 deploys TOR in a local environment, we provide a deeper introduction into TOR in this section.

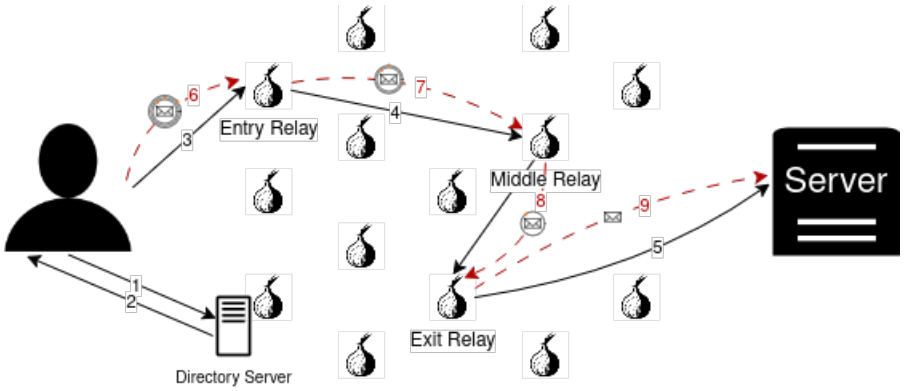


Figure 5.1: The infrastructure of TOR. A client fetches information on available relays (1-2). The client establishes a circuit towards their destination (3-5). Subsequently, they send their traffic. Relays route the traffic towards its destination (6-9).

TOR is implemented on Layer 4 (Transport Layer) of the ISO/OSI model. Thus, it does not implement its own network infrastructure but uses existing IP connections. Any communication between the involved entities is encrypted using TLS. In order to establish a connection through TOR and send traffic, a client takes the following steps (see Figure 5.1):

1-2 The client fetches information on available relays from a *directory server*.

3-5 Based on this information, the client chooses a number of relays and establishes its connection, called *circuit*, relay after relay.

6-9 The client sends its packet(s), which are encrypted in multiple layers, through the established circuit. Each relay in the circuit decrypts one encryption layer to gain the necessary routing information.

Following this coarse introduction into how communication channels are established and used, we now further describe the involved components.

5.2.1 TOR DIRECTORY SYSTEM

The main task of a directory server is to store and distribute the TOR *network consensus*. The consensus is built by all *directory authorities*, based on the information they have on deployed relays. By conducting this step, false information can be eliminated, and the trustworthiness of the information is ensured. The consensus includes information about each active TOR relay, such as its bandwidth, location, and status, which is required by TOR to select relays for their circuits. The network consensus is updated regularly and signed by directory authorities. Directory authorities are trusted nodes in the network that build the consensus and sign it cryptographically. Their role is to ensure the integrity and authenticity of the information. They have the responsibility to verify which relays are currently operating and can be trusted. To build the directory system more reliable, additional *directory mirrors* are deployed. Directory mirrors hold copies of the network consensus and can be queried by clients.

5.2.2 RELAYS

Relays are the key components of TOR's infrastructure. They are volunteer-operated and forward traffic between clients and the destinations of their traffic. There are three types of relays in TOR:

Entry Relays / Guard Relays

These are the first relays that a TOR client connects to when establishing a circuit. The entry relay knows the client's IP address but does not know the destination of the client's traffic. It serves as the initial hop in the circuit and is responsible for forwarding traffic to the second relay, the middle relay.

Guard relays are an extension of entry relays. Instead of randomly choosing a new entry relay for each circuit, TOR uses another principle for choosing the entry relay. Having a new entry relay in each circuit increases the chance that at some point one of the selected entry relays will be a bogus one and therefore increase the feasibility for a number of attacks. Hence, TOR clients select two guard relays which they use as their entry relays for any of their circuits [68]. These guard relays only change every 3.5 months. Note that we use the terms guard relay and entry relay interchangeably from now on. In this case we always talk about guard relays.

Middle Relays

They form the intermediary layer in a TOR circuit. Middle relays do not know the origin or the destination of the traffic. They only forward the traffic between the Entry and exit relay. This layer ensures that neither the entry nor the exit relays can identify the complete path.

Exit Relays

These are the final hops in a TOR circuit. The exit relay decrypts the data and sends it to the destination. While the exit relay knows the destination address, it does *not* know the original source (the client) and thus maintains the user's anonymity.

5.2.3 ESTABLISHING CIRCUITS

When a client connects to TOR, they build a circuit by first selecting three relays: a guard relay, a middle relay and an exit relay. This selection is met, based on the consensus gotten from a directory server/mirror. Attributes like a relay's bandwidth, availability, owner[2] are taken into consideration when selecting relays.

In the next step, the client establishes a connection to the guard relay. This means that the relay and client perform an authenticated key exchange. The

2 TOR relays can be operated by anyone who can provide a computer with a stable Internet connection and sufficient hardware resources for performing the required cryptography operations. For details on running TOR relays, check https://blog.torproject.org/new-guide-running-tor-relay/.

exchanged key is later used to encrypt payload data. This process is repeated for the middle and the exit relay. Note that any communication towards the middle and exit relay is performed via the guard relay, using the key material that was just exchanged.

5.2.4 ONION SERVICES

Onion Services, also called *Hidden Services*, are the server complement to clients in TOR. When TOR is used by a client to anonymously communicate with a website, communication between the exit relay and the website is happening unencrypted[3]. To tackle this problem and also hide the IP address of the server, providers can deploy their website as an Onion Service. When being started, an Onion Service builds circuits to multiple Relays. These circuits are 3 hops long and created just as they are for clients. The last relays in these circuits are called *Introduction Points*, as they introduce the Onion Service to the TOR network. Subsequently, the Onion Service provides its URL[4] via the network.

When a client sends a request to a `.onion` address via TOR, they query the network for the information *at which relay they can connect to the Onion Service*. In the next step, they build a circuit to one of the Onion Service's Introduction Points. At this Introduction Point both circuits, the client's and the service's, connect, and communication is initiated. Because client and service 'meet' at this point, it is also called *Rendezvous Point*.

5.3 DHT-BASED PROTOCOLS

GNUnet is a peer-to-peer network, which is interesting for us, as it focuses on anonymity for participating hosts. By its design, it guarantees that *the link between provider and consumer is completely anonymous* [31]. In contrast to solutions like TOR, GNUnet is not designed as an intermediate system between a user and the Internet but as an isolated peer-to-peer network. To provide anonymity for its peers, GNUnet uses *GNUnet Anonymity Protocol* (GAP) [8] for routing. This protocol obfuscates the sender and receiver of a packet by making sent and forwarded packets look identical. In contrast to Onion Routing, no layered encryption is needed: each packet is only encrypted once. To still hide communication partners, the sender of a packet stores the packet's origin and overwrites the sender field with their own identity. By applying this source rewriting, no attacker who observes the packet can know if the current source is actually the sender or only a node that routed the packet. As every node is linked to as many other nodes as possible, the packet eventually reaches its destination. A time to live field prevents loops in routing. GNUnet is counted as a DHT-based protocol because a *Distributed Hash Table* (DHT) is used to find nodes in the network, which are then used for relaying packets. Another DHT-based protocol is Torsk [53], which uses TOR but replaces its directory system and moves relay

3 However, the application tunneling its traffic via TOR can, of course, use encryption itself. For example, visiting a website via TOR can still be encrypted by using HTTPS/TLS instead of plain HTTP.

4 URLs of Onion Service always have URLs with the top-level domain `.onion`.

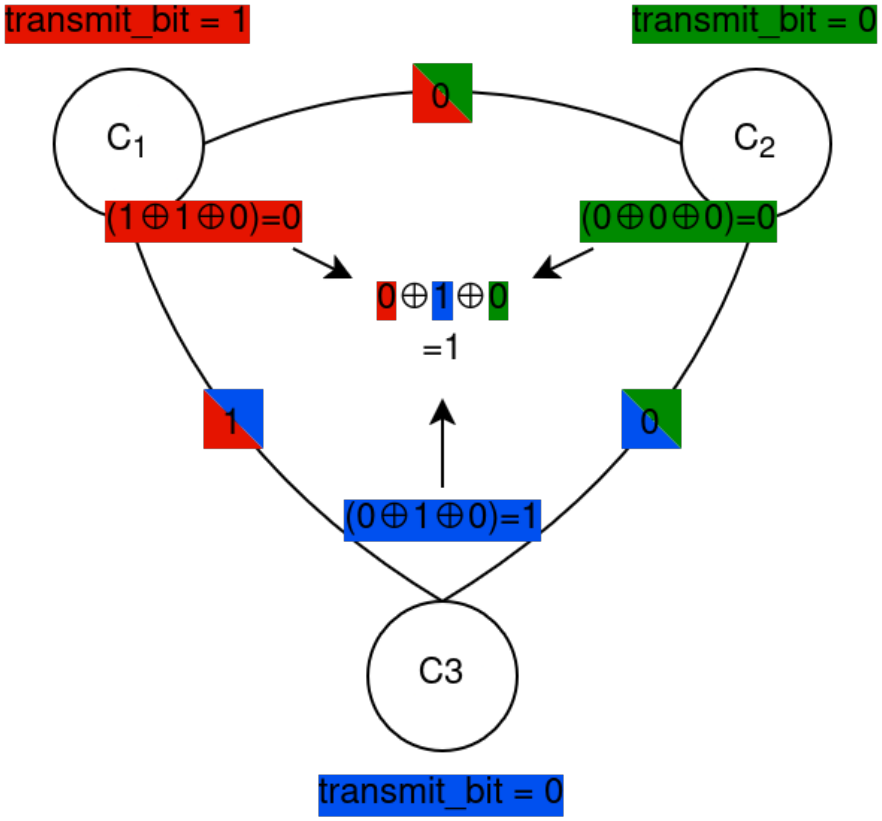


Figure 5.2: Basic principle of the *Dining Cryptographers Problem*, introduced by D. Chaum (1988).

information to the DHT. The authors argue that this replaces one of the bottlenecks of TOR, which is the bandwidth that is consumed by directory system lookups. Solutions like NISAN [63] and Octopus [83] also build the lookup of nodes on a DHT. They seek to tackle the problem of biased path selection, a common problem of DHT-based ACNs, which is induced by malicious nodes.

5.4 DCNETS

The *Dining Cryptographers Problem*, which was first introduced by Chaum in 1988 [15], sets the basis for *Dining Cryptographers Nets*, short *DCNets*. This concept is shown in Figure 5.2. Consider three cryptographers (C_1, C_2, C_3) having dinner together. They get notified by the waiter, that somebody anonymously paid their bill. To find out if one of them (without noticing who exactly) or some 4th person/entity paid for them, they conduct the illustrated algorithm. Every one of them exchanges a shared secret (1 bit in this simplified case) with their neighbor. Subsequently, they all xor their two shared bits and the information if they are the anonymous sponsor (1 if yes, 0 else). All three of them broadcast the result. These three resulting bits are xorred again. This final result tells them if one of them paid for the dinner (if the bit is 1) or if an external entity did so (if the bit equals 0). If one of them is the donor, it still is not revealed who exactly paid. This basic principle can be extended to support more parties and longer

messages. With *Herbivore*, Goel et al. propose a peer-to-peer network building on DCNets [29, 77]. Participants are assigned to cliques of communication partners. Network Nodes can be traced back to their clique but are anonymous among their fellow clique members. Communication is performed in rounds, which are again separated into three phases: In the Reservation Phase, transmission slots are assigned to participants. The slot-based communication is used to prevent collisions of messages. Subsequently, the Transmission Phase is entered. Each participating node now sends their message in their reserved slot. In the third phase of a round, it is checked whether all nodes completed the transaction of their data. Additionally, this phase allows nodes to leave and enter the clique without crashing the protocol.

In contrast to Herbivore, *Dissent* seeks to use a client/server approach [20, 87] to build an ACN based on DCNets. Compared to earlier solutions like Herbivore, this system reaches anonymity among larger groups and outperforms existing DCNet approaches in terms of performance. However, Dissent is prone to malicious participants that disturb communication. Furthermore, compared to solutions from other categories than DCNets, it still lacks sufficient performance.

Corrigan-Gibbs et al. propose a solution (*Verdict*), which builds on Dissent and extends their approach [21]. They address the problem of malicious clients disturbing communication. In Dissent this problem is already addressed, but participants blocking the network are only able to be detected retrospectively. In contrast to that, Verdict uses public key cryptography to already ensure that clients conform to the protocol *before* communication can be sabotaged by malicious clients.

However, all the portrayed DCNet solutions lack the capabilities for low-latency communication. This challenge is addressed by *PriFi*, a more recent approach introduced by Barman et al. [4]. PriFi is the closest existing solution to our work, as it aims to solve the problem in local networks. This objective is tackled, using a set of external (outside the local network) nodes on the Internet, called guards. Additionally, existing infrastructure at the edge of the local network is used as *relays*. Typical for DCNets, communication is performed in rounds. Consider Figure 5.3, which, in a simplified version, shows one round of communication in PriFi. Each client shares a key with each guard. The slot owner is allowed to send their message, all other clients must send only zeros. Each client creates a ciphertext in which they xor all their shared keys and their message. This sum is sent to the relay. Furthermore, each guard sends the XOR sum of all its shared keys to the relay. At the client, all these components are xorred again. Any of the shared keys is contained exactly twice; the first version is from a guard and the second one from a client. Therefore, the keys cancel out each other. What remains are only the xorred messages. As all but one message only contain zeros, the message of the slot owner remains at the client. By applying the explained procedure, the relay can not now from which client the message stems and forwards it to its receiver, which is contained in the message.

PriFi was implemented as a prototype and tested in a LAN with 100 clients. Sending a message from one client to a guard and back to the client takes 100 ms. Latency increases linearly with the number of clients participating in the network. Consequently, it is questionable if PriFi scales in a larger network with thousands of clients. Additionally, identification of malicious clients takes 1.5 s in the 100-client network and increases linearly with an increasing number of clients.

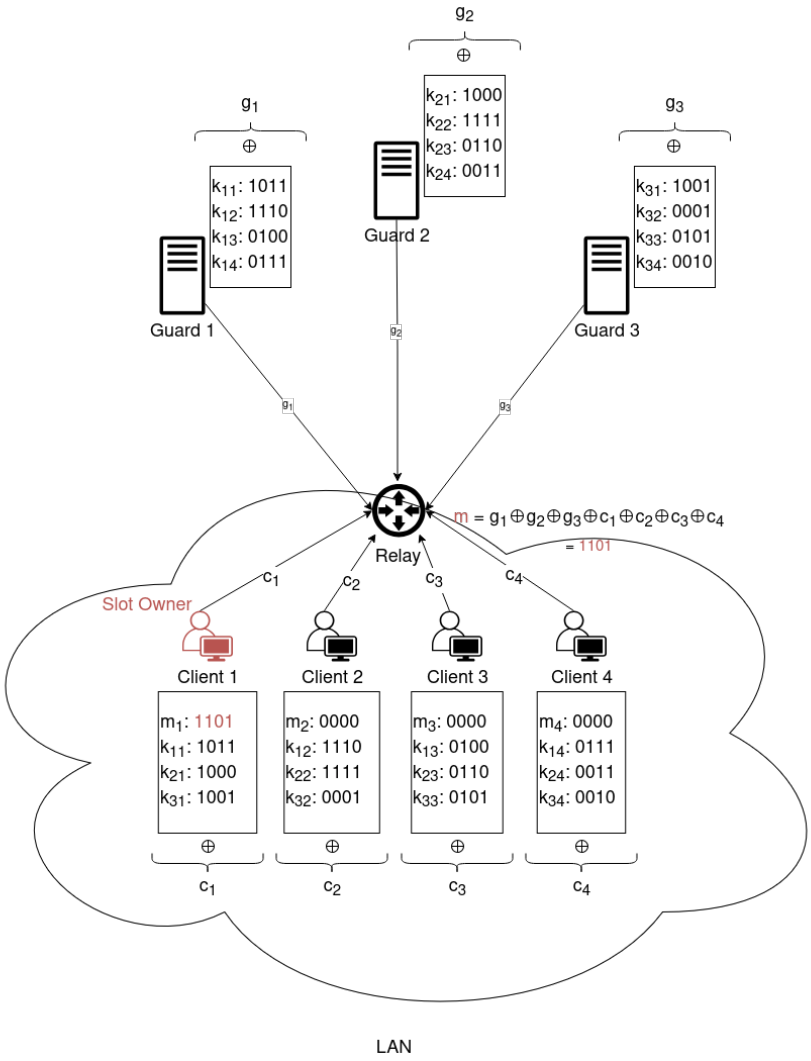


Figure 5.3: Infrastructure of PriFi [4]. One round of communication is shown.

5.5 MISCELLANEOUS

I2P (Invisible Internet Project) [70] is a peer-to-peer network that works as an overlay for the Internet. It focuses on communication inside the network and not on the outside Internet. Furthermore, it proposes some new concepts. It improves the concepts of TOR and Onion Routing: First, communication is unidirectional. This means that traffic takes another way back and forth. Hence, timing attacks and analyzing requests and responses at the same node are harder. Secondly, it introduces the concept of Garlic Routing: Packets from the same source are bundled. Each clove (packet) is encrypted with the public key of its destination. Then cloves are bundled to a bulb, and then the bulb is encrypted with multiple layers, just as in Onion Routing. Traffic is then routed, and only at the last node are cloves exposed. The last node then routes cloves towards their destination. This method has multiple advantages compared to Onion Routing. First, by bundling multiple messages, fewer connections have to be established. This reduces the traffic load in the network. Furthermore, bundling of messages makes analysis much harder for an attacker. One could even add multiple cloves

of decoy traffic to a bulb.

Another solution that does not fall into the latter categories is *Tarzan* [25]. Tarzan is also a peer-to-peer network. Hence, all clients participate in the routing of traffic.

5.6 DISCUSSION OF EXISTING SOLUTIONS

All of the presented solutions are strongly related to the concept of mix networks, presented by Chaum [16]. However, when mix networks are implemented, they heavily slow down the processing of messages. Therefore, most existing solutions modify this concept. Still, the approach of using several nodes to disguise a packet's source and destination is very prominent. Additionally, many of the presented technologies employ the concept of *decoy traffic*. This means that, next to the presented anonymity measures, random traffic is generated to prevent traffic analysis attacks. Furthermore, it seems hard to build a solution that is solely implemented in the network and not additionally on the client side. This is, because already the first link from a device to the network must be protected. It is questionable if this is a possible task. However, solutions like a VPN or proxy are easy to deploy on the client side and would greatly simplify this challenge. It could even be possible to block any communication that does not use this client-side entry point to guarantee that no conventional communication takes place in the network. All in all, the presented solutions are all intended to provide anonymity for their users in the context of the Internet. Whether the solutions build an overlay on the Internet or provide anonymous routing: None of the existing solutions (except for PriFi [4]) is able to provide anonymity in the context of LANs. Nevertheless, they provide valuable insights and therefore a basis for a prototype that can be deployed in local networks.

6 DRAFTING A LOCAL ACN

In this chapter we present thoughts on potential solutions, providing anonymity in LANs. The first part of this section introduces different ideas. Subsequently, these ideas are discussed. Finally, we decide which of the presented concepts is further pursued.

To get an overview of these solutions, we provide illustrations for each proposed prototype. A situation in a typical LAN without any anonymity measures applied is shown in Figure 6.1. In this infrastructure there is a LAN with two routers, connecting the network to the Internet. Several switches and access points connect clients to the network. An attacker sniffing at one of the switches can observe packets sent over the network. Therefore, they can link packets to their senders and receivers and gather information on specific devices.

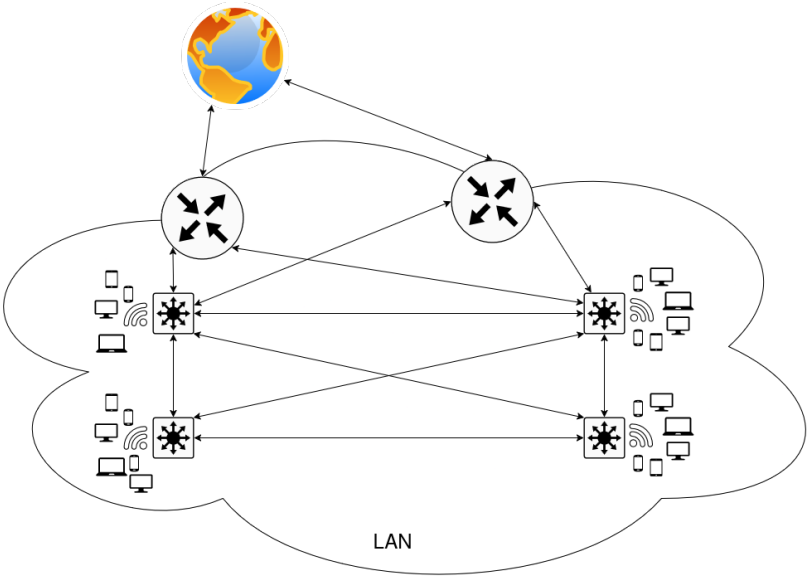


Figure 6.1: Standard infrastructure in a LAN.

6.1 TRUSTED NODE

Our first idea for a prototype builds on the concept of a *trusted node*. The general idea of this concept is shown in Figure 6.2. In this infrastructure, the original topology of the network is not changed. The key element of this idea is to encapsulate all traffic from the client in a VPN tunnel until it leaves the network. By applying this strategy, an attacker on a network node is not able to observe the packets that were originally sent by the clients anymore. However, they can still observe which client is present in the network and if a client communicates with some arbitrary destination. This scenario also assumes that the adversary has no access to the VPN server, as this is the point where packets are decrypted and forwarded to their destination inside or outside the network. For this reason the VPN-server is called *trusted node*.

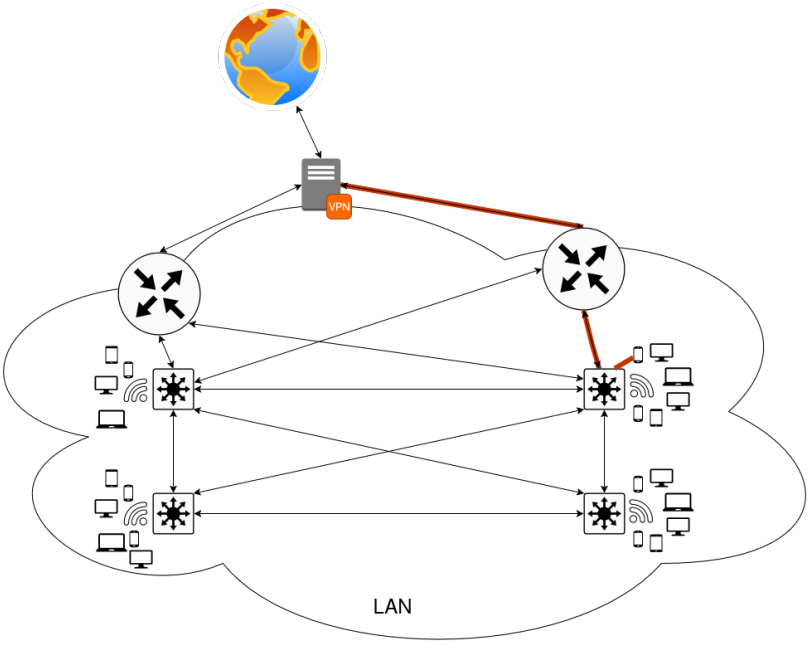


Figure 6.2: Solution with trusted node.

6.2 LOCAL ACN

Another approach is shown in Figure 6.3. Here, a local ACN is deployed to the LAN. By connecting to this ACN, the user's traffic is routed through a number of relays before being routed to its destination. This approach seeks to tackle the problem of having a single point of failure as in the first approach (trusted node): As the ACN consists of multiple nodes, there is no single point of control that could de-anonymize a user. However, the details of this concept highly depend on the deployed ACN. If, for example, TOR is chosen as the ACN, this means that the user's packets are hidden by Onion-routing (see Section 5.2). Note that in the case of TOR, this does not mean that users are connected to the globally deployed TOR network, but rather that a local instance of TOR is deployed, which is only accessible locally. Another advantage of using a local ACN is that internal services can directly be hosted in the ACN. In the case of TOR as Onion Services. Therefore, for reaching internal services, a user's traffic does not leave the ACN. An additional advantage of a local ACN is also, that the responsibility for nodes can be distributed over multiple entities, e.g. administrative units. Therefore, none of these entities can lift anonymity alone.

6.3 DISCUSSION OF POTENTIAL SOLUTIONS

The central component of the first solution is the trusted node. A successful attack on this node results in an exposure of *all* network traffic, as the trusted node is also a single point of failure. This is the biggest disadvantage of this infrastructure. The greatest advantage of this approach is that it is relatively easy to implement, as only a single node has to be deployed. In contrast to that, the biggest strength of running a local ACN is that there is no single point of

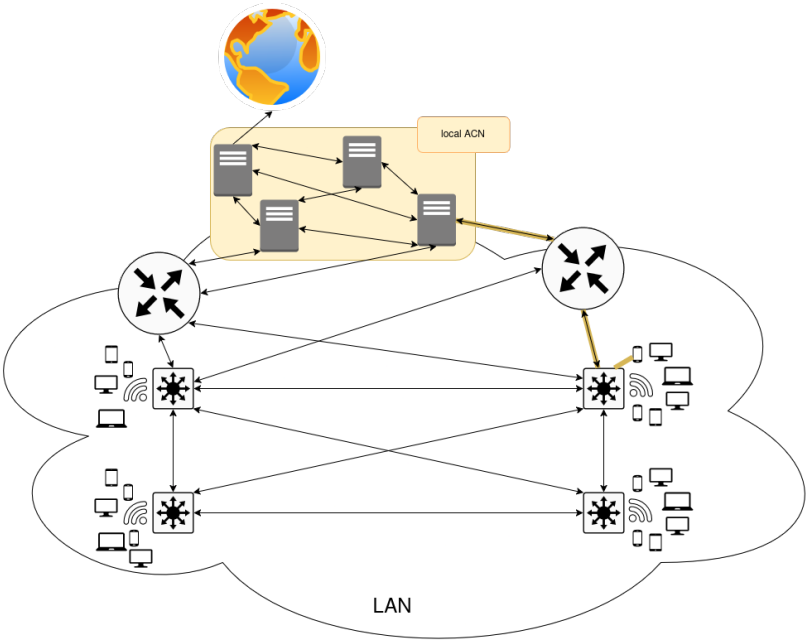


Figure 6.3: Solution with local ACN.

control left where an attacker could identify network participants. However, this approach is much more complicated to deploy: Multiple nodes have to be installed in the network, and running an ACN on these nodes also increases the network load. This is because a packet is not directly routed from a client to its destination but rather travels paths between multiple nodes before it is routed towards its actual destination. As shown in Chapter 5, there are technologies that could be adapted when using an ACN-based solution. Both ideas have in common, that they can be implemented on an existing network. Furthermore, only minor work has to be done by clients to use these solutions: In both cases, clients only need to install a VPN or proxy to their devices to initiate communication via the deployed solution.

Based on the discussed arguments, we decide to choose the solution of a local ACN for the further direction of this thesis. We make this decision mainly because we do not want to establish a single point of control in the network. Having multiple trusted gateway nodes for clients to choose from does not solve this problem, because at the chosen node all traffic routed is still accessible.

This leaves the question of whether we propose an entirely new solution or if a solution that could be deployed and solves our problem already exists. After getting an overview of existing approaches in Chapter 5, we state that there is only one technology that is suitable for our purpose, which is TOR. All other solutions have extensive drawbacks: DCNets lack the required performance for a LAN that exceeds a size of roughly 1000 clients [4]. Classical mix networks have a similar problem, as they have to wait for a number of packets to perform mixing on. DHT-based protocols and the protocols categorized as miscellaneous are mostly peer-to-peer networks. Consequently, all clients in the network need to participate as relays. This requirement violates our assumption that only little effort is required on the client side.

In contrast to these solutions, TOR is already widely deployed in another

context: the Internet. Therefore, its concept is well tested and actively maintained. Another reason is that its code base is available in an open-source fashion. Deploying TOR infrastructure is a trivial task, as it is done regularly. Next to these practical reasons, the design of TOR seems to be suitable not only for the Internet but also for LANs. One advantage is that the entire infrastructure can be deployed by network operators, and there are few changes to a client's system that have to be made.

7 DESIGN CONCEPT AND METHODOLOGY

In Chapter 6 we argue why we come to the conclusion that TOR is a fitting solution for our problem. This section outlines how we implement an anonymous LAN based on TOR. Furthermore, we present the methodology of evaluating its performance and anonymity. In the following, we use the example of a university to illustrate the concept. The reason for this is, that experts and with them, all required information for this context are available.

7.1 DESIGN CONCEPT OF AN ANONYMOUS LAN

In order to run TOR in a LAN, we use TOR's original implementation[1]. Thus, we do not change its code at any point. For TOR's design, refer to Section 5.2.

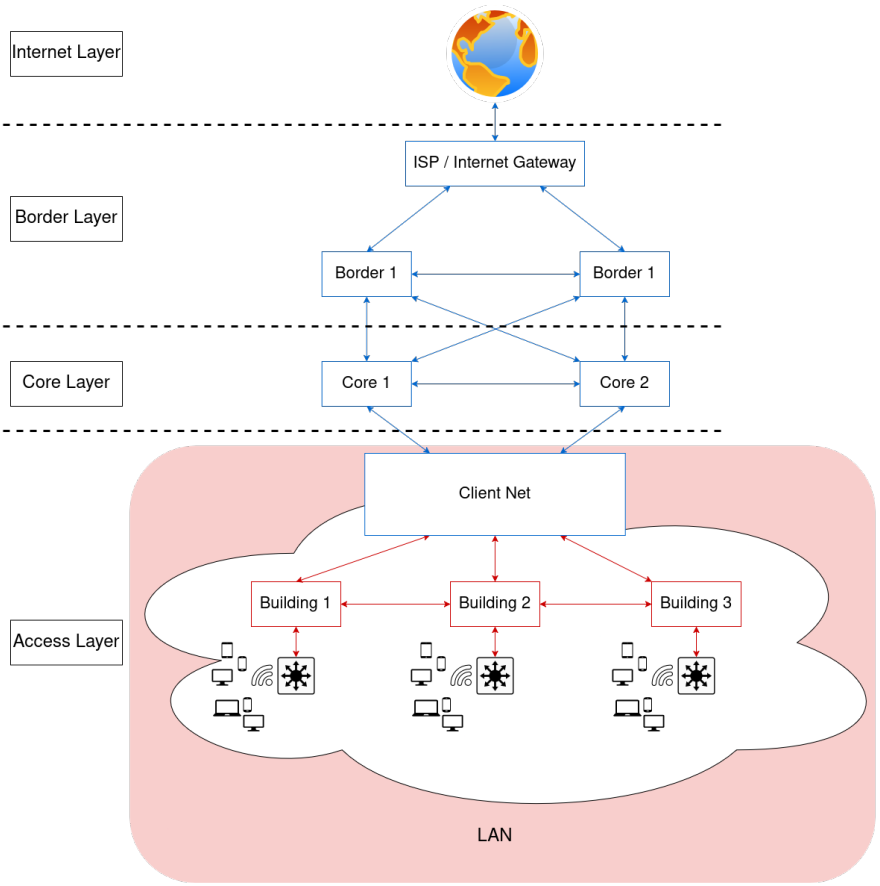


Figure 7.1: Typical LAN in the context of a university. Blue edges and network nodes represent IP-based routing infrastructure (Layer 3 of ISO/OSI), red edges and components represent switching infrastructure (Layer 2 of ISO/OSI).

1 For simulations TOR `Version 0.4.8` was used. This release is available at https://gitlab.torproject.org/tpo/core/tor/-/tree/release-0.4.8?ref_type=heads.

Consider Figure 7.1 which shows the infrastructure of a typical university network. Our concept focuses on the highlighted part of the network. We argue that this part of the network matches the definition of the term *LAN* as it is given in Section 2.6. Thus, our goal is to deploy TOR to this part of the university network. Figure 7.2 shows the changes that we apply to the network in order to deploy TOR to it. One can see, that three additional nodes (compared to the

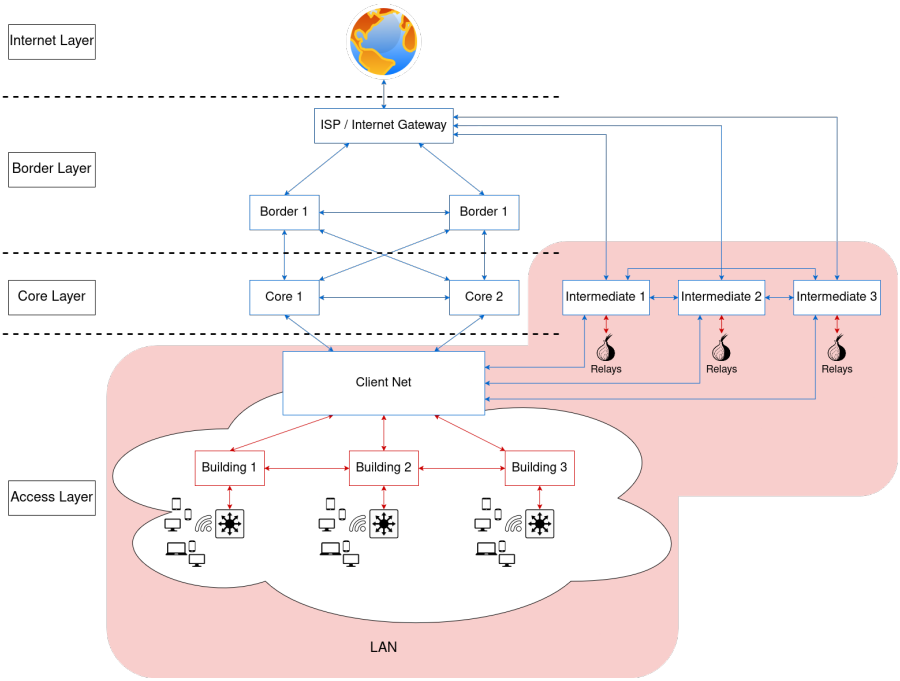


Figure 7.2: TOR, deployed to the LAN of a university. Three additional network nodes are deployed, each serving a number of TOR relays. Blue edges and network nodes represent IP-based routing infrastructure (Layer 3 of ISO/OSI), red edges and components represent switching infrastructure (Layer 2 of ISO/OSI).

LAN) are added between the client network and the gateway to the Internet. These nodes are used to connect multiple TOR relays to the network. Note that the direct link between client network and core layer of the university network still exists. We reason this decision in Subsection 7.1.1. The reason why we do not connect all TOR relays to a single node is that this could allow an attacker to observe connections from and to multiple relays much easier. Physically distributing the TOR relays therefore hardens our proposed system against attackers. Furthermore, all relays connected to the same network node are declared as one *family*. TOR's path selection algorithm never chooses more than one relay from the same family in the same circuit. We determine, that every family has to be administered by a distinct administrator or administrative unit. This is for the reason that no administrator should have access to more than one relay in a circuit at any point. In the example of a university network this can be reached by assigning the responsibility for families to different institutes. In other contexts for instance different departments can be tasked to run families of relays. Next to relays, one directory server is deployed in the network. We assume that this directory server is under the control of a trusted authority. Therefore, no malicious entity can run bogus TOR relays. Note, that this is only the case in our example. There could also be multiple directory servers deployed.

See Section 8.1 for reasons why we only use a single directory server.

In contrast to the global TOR, we are not using Onion/Hidden Services in the local TOR. The reason is, that there is no need for Onion Services in our context. Further implementational details on deploying TOR in a LAN are given in Chapter 8.

7.1.1 CLIENTS

One of our initial goals of this thesis is that there be only minimal effort for clients to participate in the anonymous LAN. As also the first link from client towards destination can potentially be observed by an adversary, anonymity must be 'initiated' already on the client side. For participating in TOR users typically route their traffic through a local proxy which relays it to the TOR network. This proxy can be contained in a browser or also be installed as a system-wide proxy. In our experiments the TOR proxy[2] was used, which requires only minimal configuration. Additionally, the official TOR browser[3] can be used, which already includes a proxy to connect to TOR.

We argue that using a proxy poses only a minimal user effort. Nevertheless, this requires the user to actively make a decision for increased anonymity in the LAN. Therefore, we raise the question if it makes sense to *enforce* communication via TOR. This can be achieved by disabling the links between the client network and the core layer of the network, shown in Figure 7.2. We acknowledge that this is a decision that has to be met individually for each network in which TOR is deployed.

7.2 EVALUATION OF THE PROPOSED SOLUTION

Our evaluation is twofold: First, we perform simulations of a LAN and a local deployment of TOR. The setup of the local TOR follows the principle given in the previous section. We then compare results and show which performance costs come with the adjusted infrastructure. The second part of this evaluation takes the anonymity aspect of our local solution into account: We define multiple attackers that get access to different parts of the network infrastructure. Subsequently, we investigate the consequences for the user's anonymity under the different assumptions.

7.2.1 EVALUATION OF NETWORK PERFORMANCE

This section focuses on how to simulate our approach in order to assess its performance. In this section we only mention exact numbers for the most important parameters. Exact values for further parameters are provided in Chapter 8. To realistically perform tests and evaluate the performance of our solution later, we first implement a LAN in a network simulation framework. Additionally,

2 The TOR proxy can be installed on Linux, macOS, and Windows and serves as a SOCKS proxy, which can be configured as a system-wide proxy to tunnel all packets via TOR.

3 For more information on the TOR browser, see https://tb-manual.torproject.org/about/.

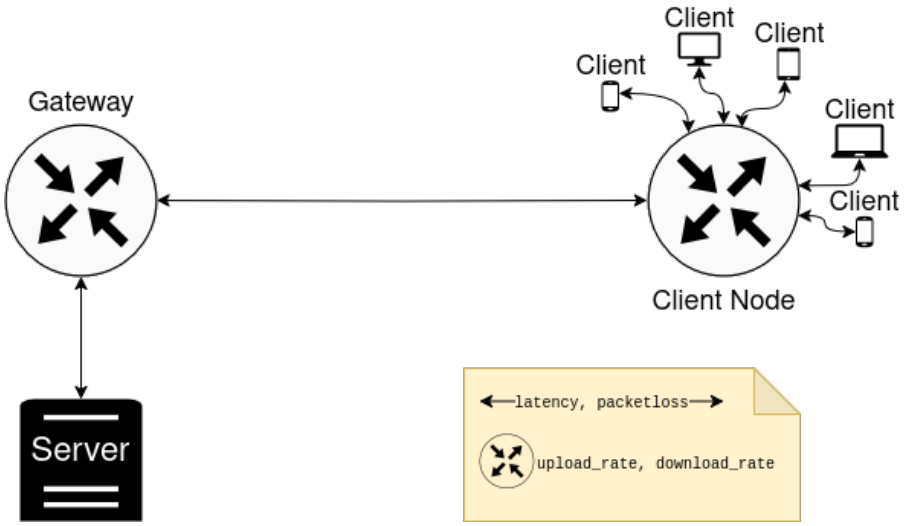


Figure 7.3: Simulation setup for a 'normal' LAN. The gateway connects the network to all external destinations. Devices are connected to the client node.

we simulate a number of clients connected to this LAN. This infrastructure is shown in Figure 7.3. It includes nodes and edges, which both represent *physical* network infrastructure. A node corresponds to a switch or router, while an edge represents a physical connection between two nodes. As one can see, simulation of the LAN includes three network-infrastructure components:

1. A client node with a number of connected clients
2. A gateway node which connects a server to the network. The gateway acts as a router to all services, hosted inside and outside of the LAN
3. Connections between gateway and server, gateway and client node and client node and clients.

Any of the network nodes has the attributes `upload_rate` and `download_rate`. These rates apply for any device connected to a node. Furthermore, any connection between two nodes or a node and devices has the attributes of `latency` and `packetloss`. With these attributes, characteristics of the connections can be set.

The reason for modelling the network with only a single client node that all clients are connected to is, that it keeps our setup simple. We choose to not simulate a more complex infrastructure because the goal of our simulations is to compare the performance between a 'normal' LAN and one with a local TOR deployed. In the LAN of a university, company, etc. there might be more connection points (e.g. access points, switches, etc.) for clients. However, we argue that using a single connection point for clients will still allow realistic simulations. We expect that the bottleneck of performance will not be network infrastructure, but TOR components. Thus, performance characteristics of a LAN can all be simulated using a single gateway connecting to all destinations inside and outside of the LAN. Attributes like upload and download speed, packet loss and latency can all be applied to this gateway and the connection between clients and the gateway.

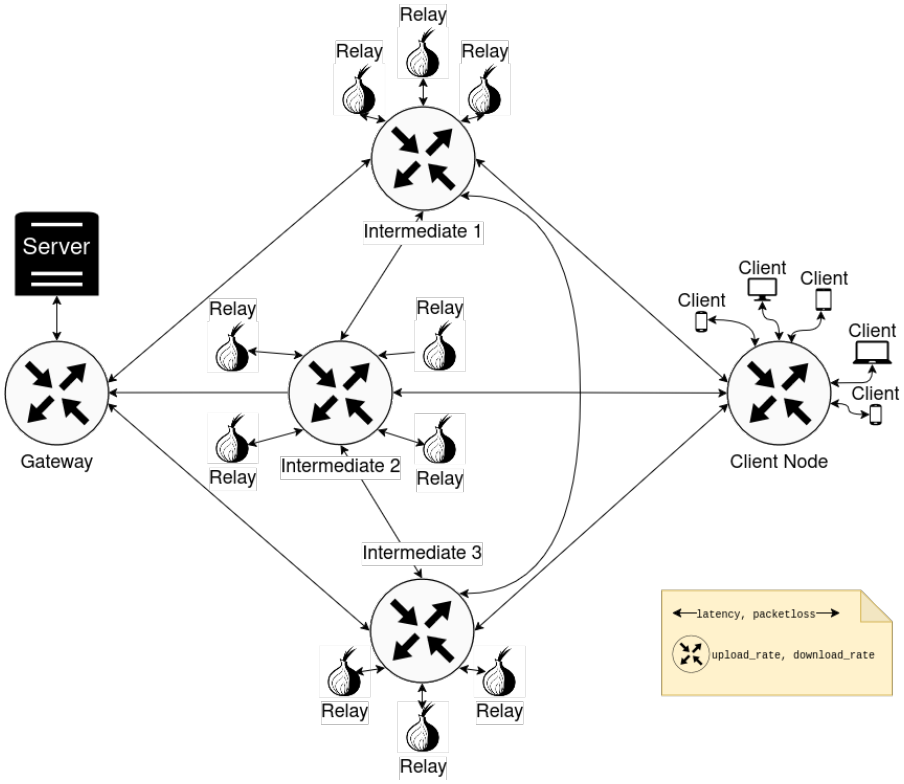


Figure 7.4: Adapted infrastructure from Figure 7.3. Three additional nodes are deployed between client node and gateway. Nodes are used to integrate a number of TOR clients into the network.

Based on the shown infrastructure for the LAN, we adapted this setup in a way that TOR can be deployed to it. The resulting network is shown in Figure 7.4.

As reported by administrators, in a typical university's LAN, the up- and download speed of nodes is set to 10 000 Mbit/s. This is a common parameter, which is provided by hardware in such a network. Furthermore, `packetloss` is set to 0.0001 and `latency` to 1 ms. Also, these values are typical values for a LAN and much lower than on the Internet. The throughput rate of the TOR relays is set to 70 MB, which can be realistically reached with state-of-the-art hardware[4] In contrast to these values, we knowingly do not define the number of relays in the TOR network. Also, the number of clients active in the network is not given at this point. These two parameters are provided with our simulation results in Chapter 9 as they are varied over the different simulations.

In order to perform measurements inside the network, we need to simulate traffic between clients and the gateway. We choose to create network traffic synthetically, mimicking three different traffic patterns. Our first traffic pattern simulates a client that opens a website. Hence, the client transfers a small amount of traffic to the gateway (request), and the gateway delivers the amount of data of the size of an average website. The second user scenario is a client uploading a file to the gateway. In this case the client transfers a large file to the gateway,

4 We contacted operators of high-performance TOR relays among the global TOR network. These values, as well as the required hardware, were reported to us. To reach throughput rates of approximately 800 Mbit/s an Intel i5 13500 or alternatively Intel i5 12500k is used by them.

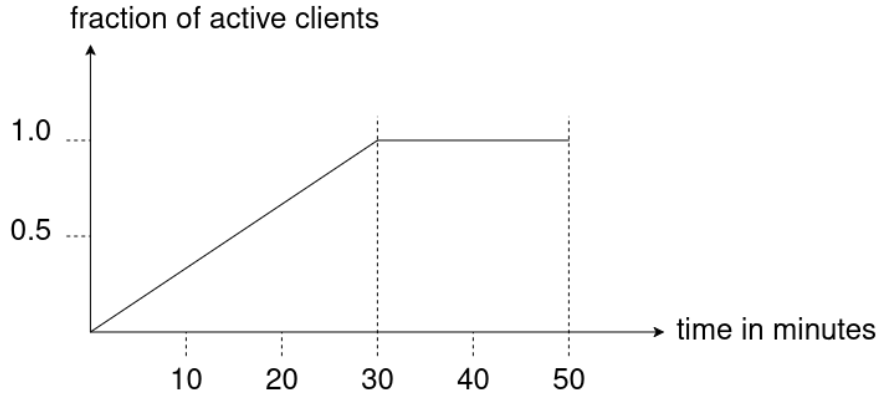


Figure 7.5: Distribution of clients active in the network during simulation. After a joining phase of 30 min, the maximum number of clients is active for a phase of 20 min.

which is consequently acknowledged by the gateway with a small-sized response. The last scenario is a client downloading a large file (e.g., a video). Therefore, the traffic pattern is simulated by a small request from client to gateway, followed by the transfer of a large amount of data from gateway to client. As a client typically performs different actions in sequence, we must simulate the delay between requests. This is done by using a statistical model. We assume that each client pauses for a while after it performs an action. For some clients this pause might last longer than for others. This pause is modeled by a normal-distribution, before the next action is executed. In consultation with network administrators, we defined packet sizes, inter-action timing and the probability that a specific action is executed. Parameters were set so that a previously defined upload and download rate are reached for each client. Based on administrators' values of experience, the peak traffic rate in a typical campus network is around 3000 Mbit/s for download and 500 Mbit/s for upload, with about 6000 clients being active at the same time. Details and exact parameters on the implementation of simulated traffic can be found in Chapter 8.

Bringing the pieces together, we simulate the process of a fixed number of clients joining the network over a time of 30 min. This joining is modeled by a uniform distribution. Hence, over the time of 30 min, the network load increases linearly. After joining the network, each client stays active until the timeout of 50 min is reached. Therefore, there is a maximum load in the network for 20 min, after which the simulation ends. Consider Figure 7.5 for the distribution of active clients in the network.

We run simulations, following this pattern with a varying number of TOR relays and different numbers of clients being active in the network. Exact parameters for these simulations are shown in Table 7.1. Note that we summarize single simulations of TOR as experiment series by omitting the numerical index (e.g. $TOR_A = \{TOR_{A_1}, ..., TOR_{A_n}\}$). We conduct a first set of simulations, TOR_A, to show how an increased number of clients affects the performance of TOR. In this series of experiments, the number of relays in the network stays constant. A second set of experiments, TOR_B, considers an average network utilization with 3000 clients. In these simulations the number of relays in the network is varied. We perform this series of simulations to observe changes in

Network	# Clients	# Relays	Experiment
LAN	3 000	–	LAN_1
LAN	6 000	–	LAN_2
Local TOR	500	144	TOR_{A_1}
Local TOR	1 000	144	TOR_{A_2}
Local TOR	2 000	144	TOR_{A_3}
Local TOR	3 000	144	TOR_{A_4}
Local TOR	4 000	144	TOR_{A_5}
Local TOR	5 000	144	TOR_{A_6}
Local TOR	6 000	144	TOR_{A_7}
Local TOR	3 000	72	TOR_{B_1}
Local TOR	3 000	96	TOR_{B_2}
Local TOR	3 000	120	TOR_{B_3}
Local TOR	3 000	144	TOR_{B_4}
Local TOR	3 000	168	TOR_{B_5}
Local TOR	6 000	72	TOR_{C_1}
Local TOR	6 000	96	TOR_{C_2}
Local TOR	6 000	120	TOR_{C_3}
Local TOR	6 000	144	TOR_{C_4}
Local TOR	6 000	168	TOR_{C_5}
Local TOR	3 000	192	TOR_{C_6}
Local TOR	3 000	216	TOR_{C_7}
Local TOR	3 000	240	TOR_{C_8}

Table 7.1: Configuration of the simulations.

the network with an increasing number of relays. These experiments are also run for a situation of high network utilization in TOR_C: 6000 clients, which result in a high network load, are active in the network, and again, for different numbers of relays, this simulation is run. For the TOR_B and TOR_C in which the number of relays changes, we also run simulations of a LAN, LAN_1 and LAN_2 with the respective number of clients. These experiments are conducted to show the trade-off in performance between a LAN without anonymity measures deployed, in contrast to a LAN with TOR deployed. Typical metrics for network performance are download and upload time for different file sizes. These values are provided in different plots that are generated using *TGenTools*[5]. An additional metric that we use is `Time to first Byte` (TTFB). The TTFB is the time between a request is issued and the first response byte. We use this metric, as it is typical for performance measurements in the global TOR. Additionally, it is often used in the context of website performance measurements, as it represents the delay which is observed by a user when loading a website. However, this time only reflects the time until the first byte is loaded and not until the full amount of requested data is received.

5 TGenTools is shipped with TGen, which was used to synthetically generate traffic and simulate uploads and downloads in the simulation network. For more information on TGenTools, refer to https://github.com/shadow/tgen/blob/main/tools/README.md.

7.2.2 ASSESSMENT OF PROVIDED ANONYMITY

The assessment of the local TOR's anonymity is done in two steps.

1. TOR itself satisfies our definition of anonymity given Section 4.2 [44, 3]. As it was originally designed for the Internet, we need to show that TOR's original assumptions are not violated by a local deployment.

2. We define three attackers which we argue to be typical for a local network. For each attacker, we analyze possible attacks and their feasibility.

Analyzing consequences of TOR in a LAN

TOR was originally designed as an ACN for the Internet. We therefore need to show that deploying it to a LAN does not violate its assumptions, which are needed for providing anonymity. We do this by investigating its assumptions, presented in TOR's design paper [24].

Analyzing attacks on local TOR

We argue that it makes sense to systematically go through the architecture of our proposed solution and list the components that could become targets of an attack. Reconsider Figure 7.2. In this illustration there are the following components that could be attacked by an adversary:

1. Intermediate nodes, which are responsible for routing between client nodes and gateway

2. Client node (Client Net), which connects clients to the network

3. Gateway (ISP / Internet Gateway), which serves as the connector to external destinations

4. Connections between all above listed nodes

5. Client devices connected to the client node via switches in client net

6. Connections between client devices and client node, including switches in client net

7. Relays connected to the intermediate nodes

8. Connections between intermediate nodes and relays

We state that the above-listed components are the ones that could potentially be attacked by an adversary and lead to broken anonymity. Therefore, we consider them in our attacker types, given at the end of this section.

Based on known attacks on the global TOR network and our defined attacker types, we conduct the assessment of our given approach. Our evaluation follows these steps:

1. Collect attacks on the global TOR from literature (**Collection**).
2. Select applicable attacks which match our defined attacker types and definition of anonymity from collected attacks (**Pre-Selection**).
3. Analyze, which of the selected attacks are also feasible against our local deployment of TOR (**Analysis**).
4. Evaluate which of the defined attacker types are a threat to the local TOR deployment.

In the collection phase, we select a recent survey on attacks against TOR to collect attacks. The pre-selection phase pre-selects a subset of these attacks. The idea of this step is to eliminate attacks that do not attack anonymity as we defined it or assume an adversary that does not fall into our attacker definition. We define our three attackers based on the components they control in a network. Thus, if an attack from literature assumes an adversary that controls components that do not match one of our attackers, this renders the attack irrelevant for us. These two conditions, which have to be met by an attack in order to get further analyzed, are called *formal assumptions*. Next to formal assumptions, there are *secondary assumptions*. Secondary assumptions are also components that are needed for an adversary to run the attack. In contrast to formal assumptions, these components are not network or TOR components. An example of a secondary assumption is that an adversary is able to run a web server or host a website. Secondary assumptions never render an attack irrelevant in the pre-selection phase but are considered in the analysis phase.

Based on the components that could become the object of an attack, we need to define a set of attackers that we consider. These attackers are defined in the following and cover the previously listed components. Consider Figure 7.6 for an overview of the defined attackers.

In our attacker types, we knowingly do *not* include an attacker who gains access on the client's device. We make this decision because an attacker who gains access to the victim's device does not need to compromise TOR to break the user's anonymity. Furthermore, we do not explicitly contain connections and switches between nodes and devices in our attacker types. This is because attacking nodes connected by these connections is analogous to attacking their network connections and leads to the same result. When we talk about 'access' to data in the context of the attackers, this means read access as well as write access. Hence, an attacker can not only sniff on data but also manipulate it. Note that traffic between client and entry relay is called *TOR ingress traffic*, whereas the term for traffic between exit relay and gateway is *TOR egress traffic*.

Administrator The first of the three attackers we consider is a malicious administrator. This means that this administrator uses any of their capabilities to break anonymity. We consider that this administrator is responsible for one family of relays in the network and has access to all the logs and routed packets on these relays. They also have access to the network node to which their family of relays is connected.

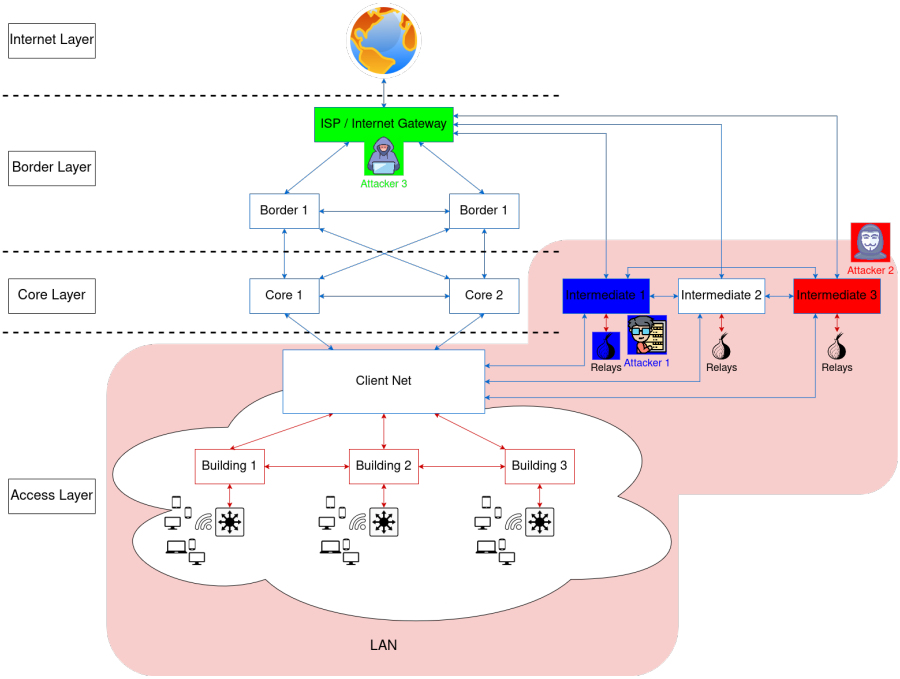


Figure 7.6: Our considered attackers. (1) Administrator with access to one family of relays. (2) Attacker inside the network with access on one network node. (3) Attacker on the network edge with access to the gateway.

Attacker inside the network Our second attacker has access to one of the network nodes *inside* the network. In the context of our simulation network (see Figure 7.4), this means that this attacker could have gained access to one of the following nodes: `Intermediate 1`, `Intermediate 2`, `Intermediate 3`, `Client Node`.

Attacker on the network edge The third attacker we consider is an adversary that gained access to the gateway that connects the local network to other networks, e.g., the Internet. In the context of our network simulations, this means that the attacker has gained access to the gateway.

8 IMPLEMENTATION

In this section we provide implementational details, which are essential for deploying TOR in a LAN. Furthermore, the implementation of simulations is described.

8.1 TOR IN A LAN

This part of the chapter describes important details on the implementation of TOR in a local network.

8.1.1 ROUTING VIA STATIC ROUTES

When relays are connected to different intermediate nodes (routers or switches), it is essential to ensure that traffic flows through the desired intermediate node. Reconsider Figure 7.4. For example, if traffic is directed from a client to a relay connected to `Intermediate 1`, it must not be routed through `Intermediate 2` or `Intermediate 3` on its way to the relay. This could potentially compromise TOR's anonymity, as an attacker could gain insight on more links of a circuit than allowed. Thus, this would violate the assumption of a non-global attacker. Without controlling the routing path, there is a risk that traffic might be routed through unintended intermediate nodes, which could break anonymity. One possible solution for this challenge is static routes. Static routes must be manually on routers acting as intermediate nodes. These static routes ensure that traffic is explicitly directed to the correct next hop, avoiding unintended intermediate nodes. We state, that these routes can not be tampered by an attacker in a way so that they can run an attack against the local TOR. Consider an adversary that has control over one family of relays and the corresponding intermediate node. The attacker can not change the static route of their own intermediate node, so that it is routed via their own node again. This is, because the attacker can only influence how outgoing traffic is routed and not how traffic is routed before it is intended to traverse them.

8.1.2 TYPES OF RELAYS IN A LOCAL TOR

In TOR, as it is deployed on the Internet, there are three types of relays: entry relays, middle relays and exit relays. As these relays are often hosted by volunteers, people can choose which kinds of relays they want to host. Operating exit relays can cause legal conflicts, which is the reason why many people choose to run entry and middle relays. However, in the context of a local TOR, these legal affairs are not an issue. Therefore, we do not deploy relays that only act as entry or middle relays. In the LAN, all relays can serve as entry, middle and exit relays. This makes relay selection much more flexible and therefore increases the number of possible circuits, which again increases the level of anonymity.

8.1.3 PHYSICAL DISTRIBUTION OF TOR RELAYS IN A LAN

In Section 7.1 we show how an existing LAN can be adapted to integrate TOR. In our simulation, three additional network nodes (switches or routers) are integrated into an existing LAN. Over these nodes, multiple TOR relays are deployed. We state that deploying TOR relays over multiple physically distributed network nodes makes sense, as an attacker needs access to all relays of a circuit to break the anonymity of this circuit. This is because of traffic correlation attacks, which are discussed in Section 9.2. A second reason for physical distribution of TOR relays is an organizational matter: If each network node with its connected TOR relays is assigned to a different administrator or administrative unit, multiple administrators have to aggregate their relays' data to lift a user's anonymity. This could be necessary in situations where a user committed criminal activities and prosecution requires breaking anonymity. Grouping all relays that are connected to a mutual network node can be achieved using the relays' `MyFamily` configuration option. Each relay has to list the identifiers of every other relay in the group. Using this configuration option, every circuit in the local TOR will visit relays from the three distinct groups.

8.1.4 TOR DIRECTORY SERVERS

Next to relays, which have to be physically distributed, there is also the need to protect the directory system from being manipulated. One option is to deploy multiple directory servers, e.g., one per family of relays. This ensures that no administrator can manipulate the consensus in such a way that a circuit is built of only their nodes. Another option is to host only one directory server at a trusted authority. Using this option, no attacker can attack TOR's anonymity by manipulating the directory system. In our simulations we choose the second option for the sake of simplicity.

8.2 IMPLEMENTATION OF SIMULATIONS

This section describes the implementation of simulations, which are introduced in Chapter 7.

8.2.1 SIMULATION FRAMEWORK

For the simulation of our approach, the *Shadow Network Simulator* (Shadow) [40, 41], a discrete event simulator implemented by Jansen et al., is used[1]. We chose to use this specific simulator, as it is also used by the TOR Project to perform simulations of the global TOR network. The developers of Shadow state that *client performance in Shadow closely matches live statistics gathered by the Tor Project, with download time quartiles within 15 percent of the live statistics for various download sizes* [40]. Thus, Shadow is a suitable tool for performing the simulations.

1 See `https://shadow.github.io/` for the documentation of Shadow.

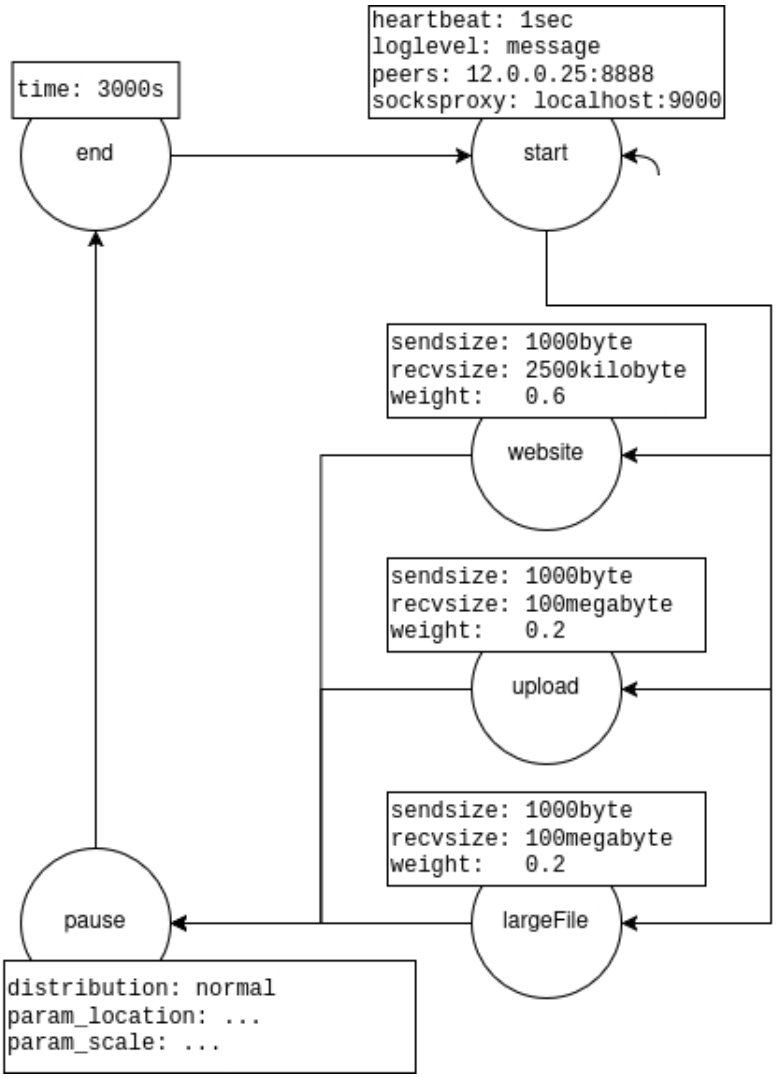


Figure 8.1: The TGen graph modeling our simulation. Clients choose between three different actions and run them until the simulation ends after 50 min.

8.2.2 TRAFFIC GENERATION

To measure the performance of our simulated network, traffic between clients and the server has to be generated. For this purpose, the tool *TGen*[2] is used. Traffic patterns from client to server can be modeled using a graph. Consider Figure 8.1, showing the TGen graph of our simulation. Each of the active clients chooses between three actions: Requesting a website (2.5 MB), downloading a large file (100 MB), and uploading a file (2 MB) were implemented. In every TGen graph, only *one* start vertex must exist. This node is the entry point when starting TGen on the client. Furthermore, connection information for the client to the server is defined on the start node. The 'heartbeat'-attribute defines the interval in which the server and client communicate status messages on successfully delivered packets, errors, etc. The attribute `loglevel` sets the loglevel of TGen. Using the attribute `peers`, the server's address and port are set. Additionally, the

2 See https://github.com/shadow/tgen for details.

attribute `socksproxy` defines, through which proxy TGen's communication is tunneled. From the start vertex, TGen travels to one of the three next nodes. The regarding node is chosen by the probability of its weight. The next node defines the actual communication from client to server. The attributes `sendsize` and `recvsize` are set to control the size of the sent and received packet (from the client's perspective). After performing request and response, TGen proceeds to the pause state. The pause between two states (in this case `webFlow` and `end`) is modeled using a stochastic model (Markov Model). The attribute `distribution` defines the type of distribution that models the pause time, after which TGen moves to the next state. In our case, a normal distribution is chosen. The next two attributes (`param_location` and `param_scale`) set the characteristics of the normal distribution. These values are different for each action and are provided in Table 8.1. After the pause, TGen transitions to the end state. At this state, it is decided whether TGen exits or moves to a next iteration of the graph. In our simulations, we choose a timeout as an exit condition. If this timeout of 3000 s is reached, TGen quits; otherwise, it moves on to the start state.

Consider Table 8.1, which provides the parameters of all three implemented actions. Running 6000 clients in the network at the same time results in a total

Action	**Req. [MB]**	**Resp. [MB]**	**Par. location [s]**	**Par. scale [s]**
Website	1	2.5	40	20
Large file	1	100	600	40
Upload file	2	1	40	20

Table 8.1: Configuration of the simulation for three different actions. Req. = request size, Resp. = response size, Par. = parameter.

data rate of 3360 Mbit/s in download traffic, which is routed from all clients towards the gateway. Additionally, a data rate of 480 Mbit/s in upload is reached. This corresponds to a peak rate of traffic in an ordinary university network.

We chose the Gaussian curve based on an estimation of users' behaviors. We assume that some users will pause longer after an action than others. For the web surfing action, we choose a distribution around 47 s (expectation) with a standard deviation of 20 s. We decide on these specific values, as users stay on a web page for around 47 s (average, as of 2023) [42]. For the other actions, we start with making educated guesses on the typical file sizes of the given actions. As we set a targeted network throughput, we calculate the required pause times. Additionally, for the large file download, we assume, based on values of experience, that a video file of 100 MB typically covers a video length of 10 min. Furthermore, a video is something that is typically downloaded in the context of a university.

9 RESULTS

This section presents the results we got by applying our methods, given in Chapter 7.

9.1 ASSESSMENT OF NETWORK PERFORMANCE

Results are based on simulations listed in Table 7.1. Simulations are referenced based on the introduced experiment numbers. Furthermore, we use metrics introduced in Subsection 7.2.1.

In our first simulations, we used TOR's version 0.4.6[1]. Running simulations using this particular version of TOR, we were surprised by the low performance results. Investigating the reasons for this behavior, it became clear that TOR deployed congestion control in its version 0.4.7 [67]. The implemented congestion functionality is called *TOR Vegas* and is revisited later in this section. Running the same simulations with TOR's most recent version (0.4.8[2]) resulted in a clear performance increase. We show the difference in performance, observed in TOR_{B_4}, between the two versions in Figure 9.1. One can clearly see that the TTFB decreases drastically with the congestion control mechanism being present. An additional preparative experiment we ran did *not* follow our timeline, presented in Chapter 8. In this experiment we tested if running experiments for a longer time (10 hours) results in a leveling behavior of TOR. More specifically, one hypothesis in this simulation was that TOR relays need additional time to adapt to load. Additionally, we tested if TOR needs the network to run for a specific time to evenly distribute the load over all available relays. These hypotheses were discarded.

We start presenting our results, based on the most recent TOR version, with an overview of how the network performance changes under an increasing load. Consider Figure 9.2, which shows data from TOR_A. Recall that in our locally deployed LAN, there are only relays that serve all three purposes: entry, middle, and exit relay. This figure shows the cumulative fraction of requests and their corresponding TTFB. We chose the amount of 144 relays because it is the number of relays at which there is only a small delay for the amount of 500 clients. With an increasing number of relays, the TTFB also increases. In this scenario this behavior already starts for 1000 clients. With a further increasing number of clients, the TTFB also increases. However, as the number of active clients in the network increases, the differences in TTFB seem to decrease. Note that for the shown figure, it makes no difference which action in the TGen graph is executed but shows the average TTFB over all executed actions. When viewing the figure, one also gets a first impression of the orders of latency in which a local deployment of TOR operates.

1 The release of TOR 0.4.6 can be found at https://gitlab.torproject.org/tpo/core/tor/-/tree/release-0.4.7?ref_type=heads

2 The release of TOR 0.4.8 can be found at https://gitlab.torproject.org/tpo/core/tor/-/tree/release-0.4.8?ref_type=heads

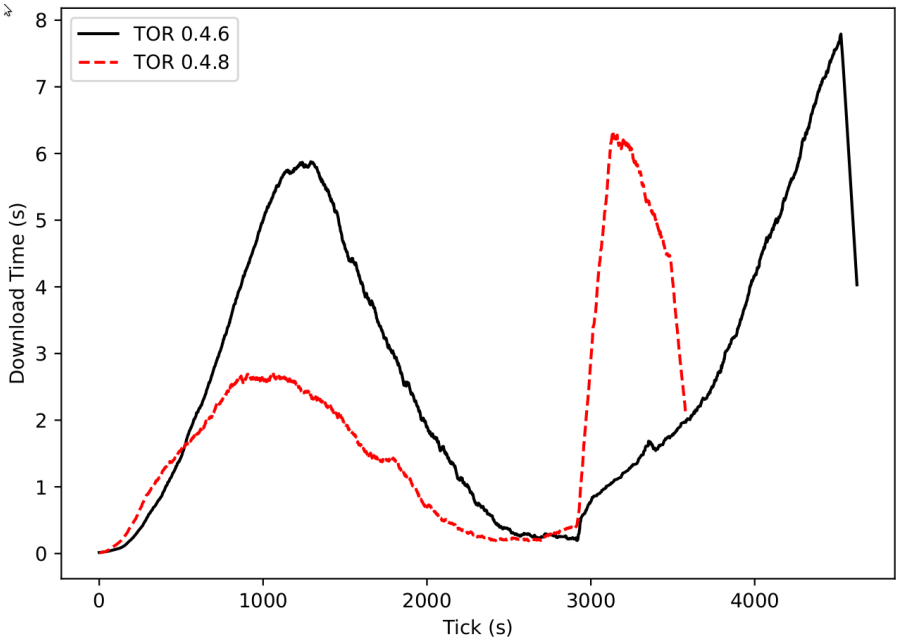


Figure 9.1: Moving average TTFB during simulation TOR_{B_4}. The comparison of two TOR versions is shown. In TOR 0.4.7 a congestion control algorithm was implemented (TOR Vegas).

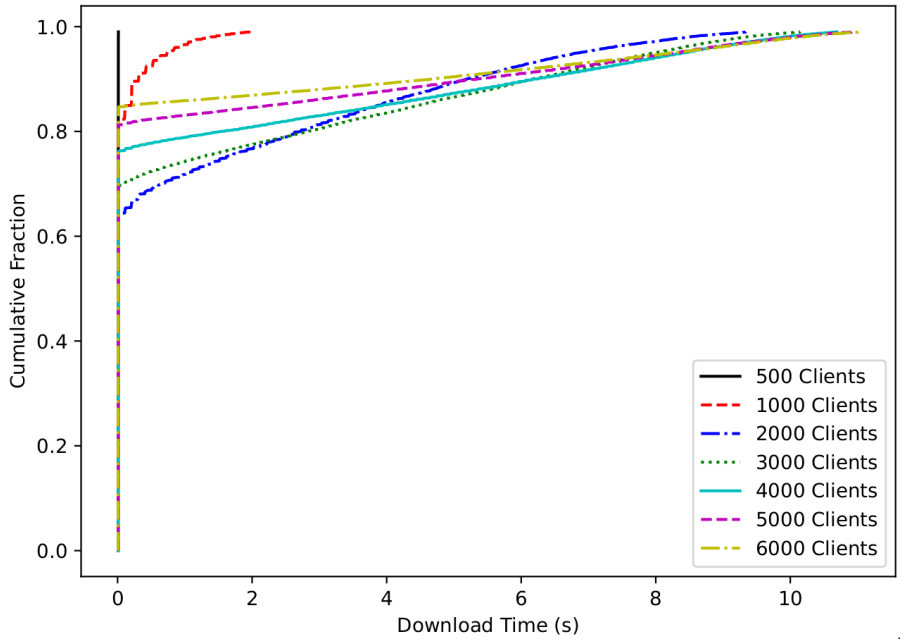


Figure 9.2: Average TTFB in TOR_A. The cumulative fraction of requests and their corresponding download time is shown.

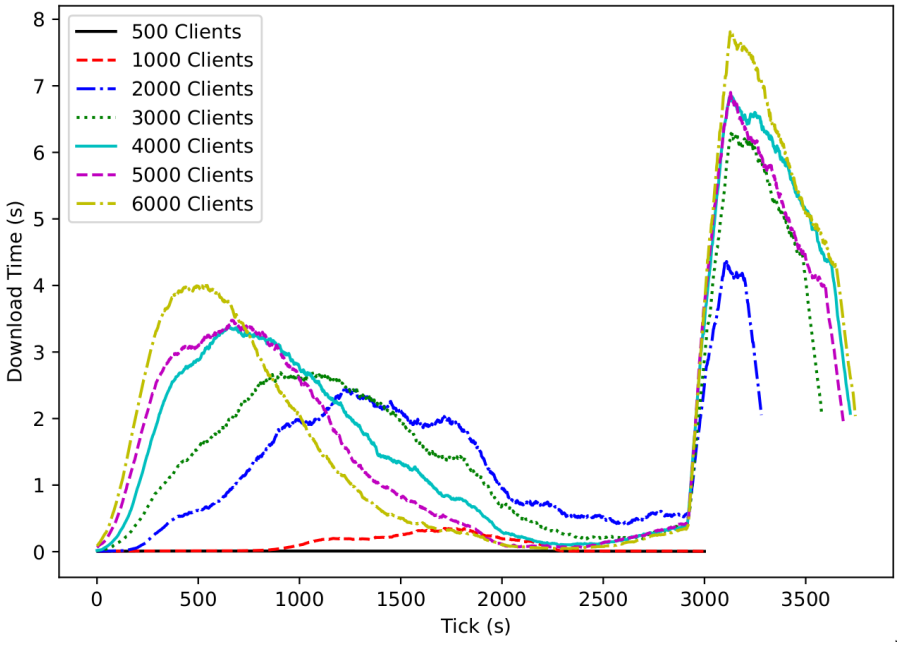


Figure 9.3: Moving average TTFB in TOR_A. The average TTFB of a transfer action during the process of simulation is shown.

Figure 9.3 shows the ongoing process of the simulation on the x-axis. The y-axis shows the average TTFB at the given point in the simulation. We observe that at the start of the simulation, the average TTFB increases. The more clients active in a simulation, the faster and further the TTFB increases in the beginning of the simulation. One can also see two peaks in the given graphs. The first of them appears at the point in the simulation when the load on the network first increases. Recall that until 30 min into the simulation (1800 s), the number of clients in the network and therefore also the load rises linearly. We suppose that the network 'adapts' to the load and, after a 'ramp-up' phase, relays cope with the load. The second peak in the TTFB starts at 3000 s (50 min). This is the point at which clients stop their activity. One question is why at this point the TTFB increases by this much. One reason is that each stream is included in the graph at this point where it successfully terminates. Hence, if a stream started at an earlier point in the simulation (e.g., at 2000 s) but finishes at 3000 s, its TTFB is part of the average at 3000 s and not at 2000 s. We further investigated the shown peak and found that it is caused by slow streams that only terminate at the end of the simulation.

We assumed that it is possible to fix this behavior by adding more relays to the network. Thus, we ran simulations with a higher number of relays in the network (simulation series TOR_B). We chose a mid-level amount of 3000 clients, and therefore load, and ran simulations with increasing numbers of relays. Again, we show the average TTFB for these simulations in Figure 9.4. One might expect that more relays result in higher performance. Our results show that this assumption does not hold. Even with many fewer relays (72), the average TTFB stays approximately equal.

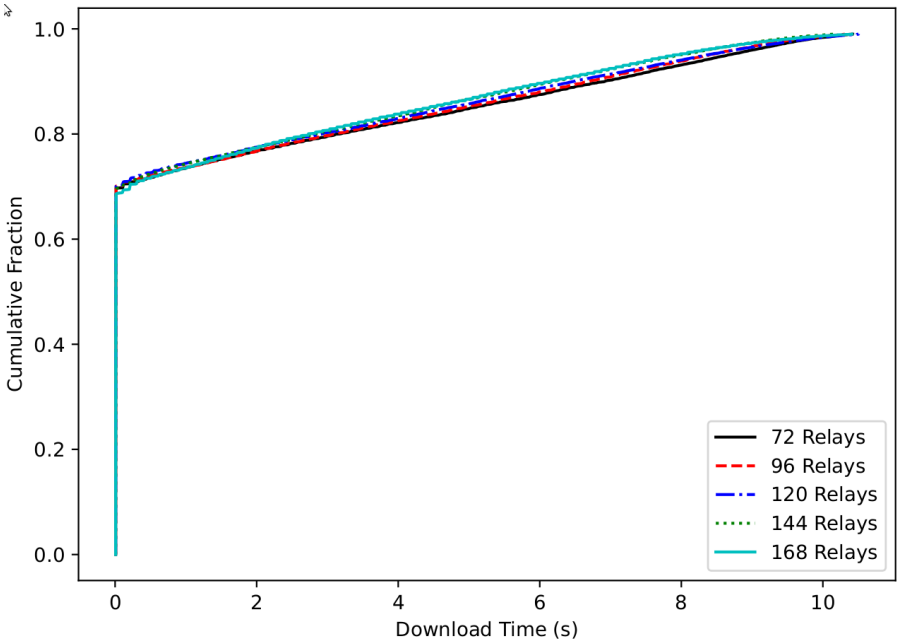


Figure 9.4: Moving average TTFB in a local TOR with different numbers of relays. Results stem from simulation series TOR_B. The average TTFB of a transfer action during the process of simulation is shown.

Consequently, we decided to investigate the cause of this behavior and took a closer look at active relays in the local TOR. For this case we analyzed data of TOR_{B_1} with a low number of relays (72) and of TOR_{B_4} with a high number of relays (144). We calculated the average throughput at each relay during the simulations. Data from relays of both simulations is shown in Figure 9.5. A first observation when looking at throughput rates of relays is that, compared to their configured maximum possible throughput of 560 Mbit/s, their average utilization is up to 10 times lower than possible. A second fact is that the total amount of transmitted data is equal for both plots. Therefore, with 72 relays, at each relay the throughput is twice the throughput of a relay from 144 relays. This matches our observation that an increasing number of relays does not bring a higher performance: More relays only lead to a lower throughput for each individual relay, regardless of the theoretically available bandwidth. Next to the average throughput at each relay, we also show the maximum throughput relays reach. We note that relays are able to route with a much higher throughput than they were configured to. As the maximum throughput is up to 10 times higher than the average throughput, we investigated if these values are outliers or if calculating average values distorts results. We were able to find out that the high maximum throughput values are outliers indeed.

In Figure 9.6 and Figure 9.7 we show the throughput rate at two relays over time. Note that these relays were active in a simulation with 3000 clients and 144 relays. Figure 9.6 shows the throughput for a relay that has the lowest average throughput during TOR_{B_4}. In contrast to that, Figure 9.7 shows the throughput of a relay that has the highest average throughput in TOR_{B_4}. When looking at these two plots, a first problem of TOR becomes visible: relays from the same network heavily differ in their throughput utilization. This can also already be seen at Figure 9.5.

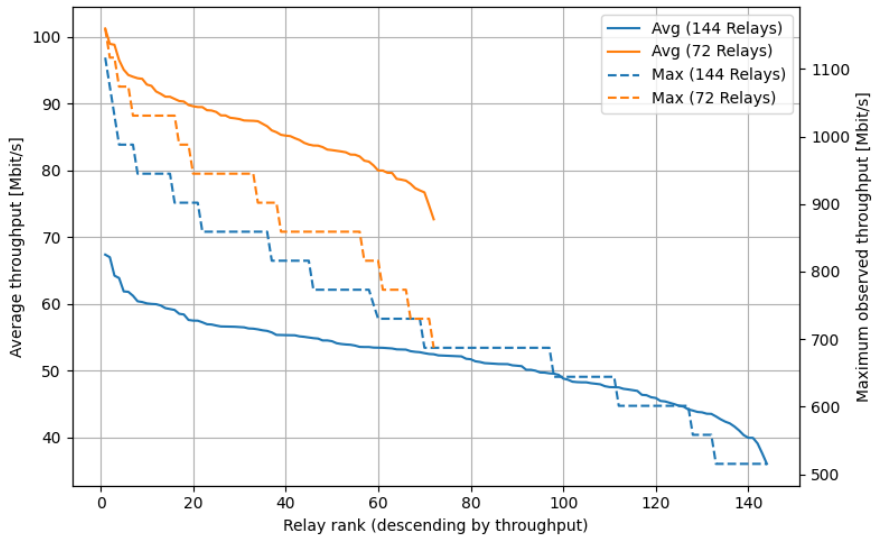


Figure 9.5: Ranking of observed average throughput values (left y-axis) and observed maximum throughput values (right y-axis). The plot shows values of relays from TOR_{B_1} and TOR_{B_4}. Additionally, the maximum throughput values, observed during simulations, are shown.

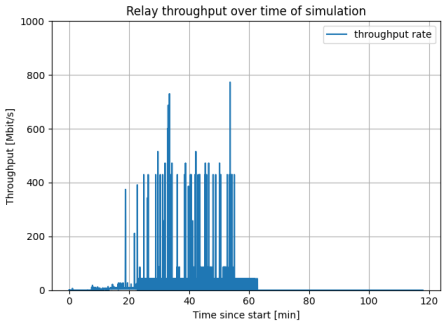


Figure 9.6

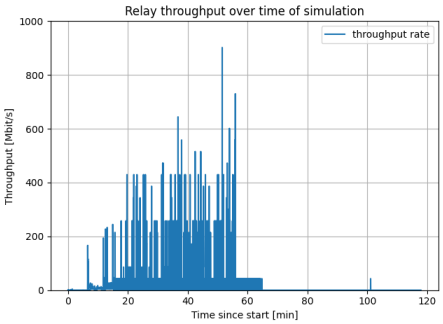


Figure 9.7

Throughput of TOR relays over the time of simulation TOR_{B_4}. Figure 9.6 is the relay with the lowest average throughput rate and Figure 9.7 is the relay with the highest average throughput rate.

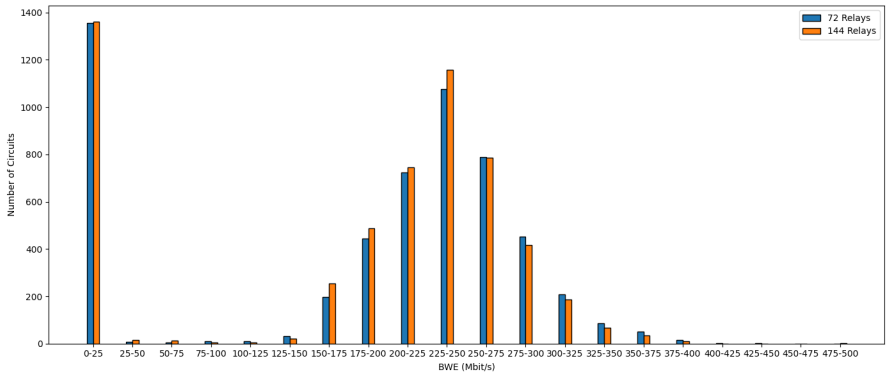


Figure 9.8: Average BWE values (Bandwidth Estimation) of client's active circuits during TOR_{B_1} and TOR_{B_4}. Bandwidths are estimated by TOR Vegas.

We state that a second problem is that none of the relays operates at its throughput limits (except for some rare outliers). The routed traffic stays far under the available bandwidth. Looking into details of TOR that could explain this, again we found one possible cause in its congestion algorithm, TOR Vegas. Each client periodically calculates the bandwidth of its active circuits during their lifetime. However, measurements are only approximations and based on a multitude of parameters. In TOR Vegas, BWE (Bandwidth Estimation) is the value that corresponds to the approximated bandwidth of a circuit. Figure 9.8 shows the average BWE values of circuits during TOR_{B_1} and TOR_{B_4}. When looking at these graphs, one can see that for both simulations, the distribution of values is almost equal. This implies that for both simulations the same number of circuits is built, which was to be expected, as both simulations have the same number of clients with the same traffic pattern. The graph also implies that the estimated throughputs of active circuits are approximately equal. Another observation is that the measured bandwidth estimations made by TOR Vegas match the average throughputs that are routed through relays. Hence, TOR clients adhere to the estimated bandwidth capacities and only send the amount of traffic at a time that is allowed by TOR Vegas. However, bandwidth estimations of circuits heavily underestimate the available bandwidth at relays in the network. Therefore, less traffic than possible is routed. The given result also raises the question of why there are over 1 300 circuits with an estimated BWE of less than 2 5 Mbit/s. Analysis showed that these circuits are responsible for low-throughput streams, shown in the beginning of this section. BWEs do not match the observed bandwidth utilization of active relays, which was previously discussed. If relays were at their bandwidth limit during simulations, low BWE values were reasonable, but this is not the case in the given situation. We propose two possible causes for the estimated low-throughput circuits. The first reason could be that TOR-Vegas measurements are faulty. A second reason could be that at the points at which measurements were performed, relays that are visited by these circuits were at their capacity limits, indeed. We found that low-throughput circuits did not start with a high throughput at the beginning of their lifetime, and throughput decreased over time. Rather, these circuits had a low estimated throughput immediately after their creation.

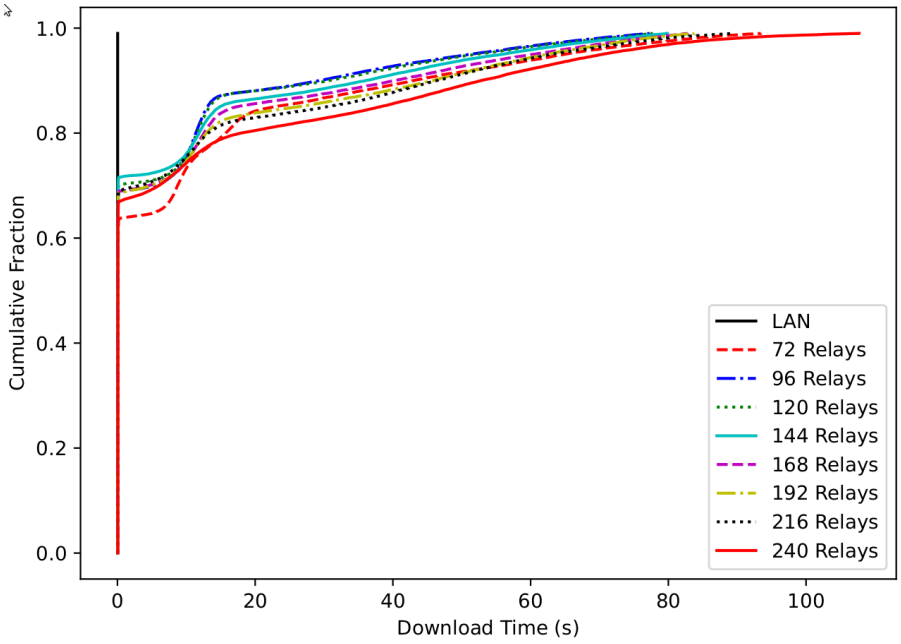


Figure 9.9: Time to download 2.5 MB during TOR_C and LAN_2

We now summarize the presumed causes for TOR's performance problems. One reason for a number of low-performance streams is that there are low-throughput circuits. We suppose that they should not be used if there are alternative circuits that offer a much higher throughput. A second cause for high TTFB values is that TOR's congestion algorithm Vegas underestimates the available bandwidth of TOR relays. We suppose that these first two problems fall into the same category. A third problem in TOR seems to be the uneven distribution of load over TOR relays.

Finally, to compare network performance in a LAN without TOR and in the local TOR, we present results from a LAN with 6000 active clients and the corresponding situation in a local TOR. Figure 9.9 shows this data from TOR_C and LAN_2. As one can see, 70% of the clients download a file of 2.5 MB in less than one second. As we were able to show with the last illustration, the rest of the streams, which take up to 100 s for download, are caused by low-throughput circuits. All in all, the local TOR deployment is clearly outperformed by the LAN. Reasons for the low performance of the rest of the streams are given earlier in this section.

9.2 ASSESSMENT OF ANONYMITY

This section analyzes the anonymity provided by the local deployment of TOR. Following our methodology, we argue that our local deployment of TOR does not violate TOR's assumptions. We now explain why we come to this conclusion.

In TOR's design paper, the authors assume an adversary who can *observe some fraction of network traffic; who can generate, modify, delete, or delay traffic; who can operate onion routers of his own; and who can compromise some fraction of the onion routers.* [24]. This is also called a *non-global adversary*. TOR's mentioned assumption is also equivalent to our assumed attacker model, presented in Sec-

tion 4.1. Furthermore, a local deployment of TOR satisfies our attacker model as we prevent a *global adversary*. This is possible because we deploy TOR in such a way that the network topology disables such an attacker. In Section 8.1 we describe how we ensure this. There are no other assumptions that are made by the designers of TOR. Hence, we argue that deploying TOR to a local network does not violate its original assumptions.

After reasoning why TOR's assumptions and threat model are also valid for deployment in a LAN, we proceed with analyzing the feasibility of existing attacks. For this purpose we first collect existing attacks on TOR from literature. For collection, we rely on a work that systematically presents existing attacks on TOR. This work is contributed by Karunanayake et al. [43]. They classify attacks into the following categories, based on the targeted infrastructure:

1. Attacks targeting both, Entry and exit TOR relay
2. Attacks targeting a single TOR component (relay, a user's TOR proxy, a destination server)
3. Attacks using a side-channel, e.g. not targeting the TOR components itself, but a network component traversed by TOR traffic (e.g. router)
4. Hybrid attacks: A mix of the previous three attacks, e.g. a TOR relay and a side-channel

In Table 9.1 we list all publications from the given categories. In the 'Target'-column we note the attack's target. For instance, if the attacker aims to de-anonymize TOR clients, we note down 'anonymity of TOR client'. In the 'Attacker's capabilities'-column we note the required capabilities for an attacker to run this attack, e.g., sniff access at a TOR entry relay or sniff TOR ingress traffic. Together, these two columns hold the formal assumptions that are relevant to deciding if we further analyze this attack. Hence, if one of these formal assumptions does not match our attacker model or our definition of anonymity, this assumption is highlighted red, and we do not take it into account for our further analysis. Consequently, all three attacker columns ('A1', 'A2', 'A3') are marked as infeasible by leaving them blank. Next to formal assumptions, the 'Attacker's capabilities' column also holds the secondary assumptions (such as the ability to host a website), which are marked with an asterisk*. Recall that secondary assumptions can not render an attack irrelevant in the pre-selection phase but only during analysis. For details on the used terminology, consider Subsection 7.2.2.

Attacks matching our attacker model and definition of anonymity are further discussed in the analysis phase and presented in the following. We discuss each of them and decide if they are feasible to be run by the three attackers ('A1', 'A2', 'A3'). If an attack falls into the scope of each of the three attackers, it is denoted in the regarding columns with ✓.

Publication	Target	Attacker's capabilities	A1	A2	A3
Attacks on Entry and Exit Relay					
[6, 5, 88, 1, 86, 26, 48, 75]	break anonymity of client	sniff traffic at entry relay + sniff traffic at exit relay			
Attacks on single TOR component					
[57, 91, 51]	break anonymity of Hidden Service (HS)	run TOR client*			
[45]	break anonymity of HS	sniff traffic at entry relay			
[47]	break anonymity of client	manipulate traffic at exit relay + run BitTorrent peer*	✓		
[89]	break anonymity of client	manipulate traffic at entry relay	✓		
Side-Channel Attacks					
[58, 81, 59, 62]	break anonymity of client	sniff TOR ingress traffic + sniff TOR egress traffic			
[34, 66, 11, 84, 85, 32, 65, 60, 92, 74, 78]	break anonymity of client	sniff TOR ingress traffic	✓	✓	
[64]	break anonymity of HS	sniff TOR ingress traffic			
[30, 71]	break anonymity of client	sniff TOR ingress traffic + access on Website Oracle*	✓	✓	
[28]	break anonymity of client	sniff TOR ingress traffic + run web server*	✓	✓	
[33]	break anonymity of client	manipulate TOR ingress traffic	✓	✓	
[2]	break anonymity of client	sniff TOR ingress traffic + run web server*	✓	✓	
[90]	break anonymity of client	run manipulated USB charger*			
Hybrid Attacks					
[61]	break anonymity of HS	run TOR client* + run middle relay			
[7]	identify circuit relays	run middle relay + run exit relay			
[54]	break anonymity of HS	sniff TOR egress traffic + run TOR client*			

Publication	Target	Attacker's capabilities	A1	A2	A3
[54]	identify circuit relays	sniff TOR egress traffic + run TOR client*			
[56]	identify circuit relays	run TOR relay + run web server*			
[36]	break anonymity of client	run TOR relay + run web server*			
[13]	break anonymity of client	run web server* + run network clients*			
[49]	break anonymity of HS	run TOR client* + run multiple entry relays + run web server*			
[27]	identify circuit relays	run exit relay + run multiple middle relays			
[80]	break anonymity of client	run web server* + run multiple entry relays + run multiple exit relays			
[14]	break anonymity of HS	run web server* + sniff TOR ingress traffic + sniff TOR egress traffic			
[14]	break anonymity of client	run web server* + sniff TOR ingress traffic + sniff TOR egress traffic			
[10]	break anonymity of HS	run TOR client* + run multiple entry relays			
[52]	break anonymity of client	ultrasound enabled device* + run exit relay or hidden service	✓		
[38]	break anonymity of HS	run TOR client* + run entry relay			
[37]	break anonymity of HS	run TOR client* + run multiple entry relays			

Table 9.1: Overview on de-anonymization attacks against global TOR, based on survey by Karunanayake et al. [43].

We argue that identification of all circuit relays is not a threat because it is not a secret which network the victim is a part of. We state that in order to identify the client the attacker additionally needs access to the entry relay of the victim's circuit. Therefore, attacks which have identification of a circuit's relays as their target [56, 27] are also not subject of further analysis.

All attacks marked as relevant in Table 9.1 are now further described. We also discuss for each of them if they are also a threat to our local deployment of TOR.

9.2.1 ATTACKS ON ENTRY AND EXIT TOR RELAY

Attacks from this category are not part of our attacker model. Therefore, we only give a brief and general overview of this category. Consider Figure 9.10 for an overview of the methodology of this attack type. There are two challenges coming with this attack type. The first challenge is to gain access to both the entry and exit relays of a victim's circuit. This challenge can be tackled using a variety of methods[3]. For instance, new relays can be deployed by an attacker. Additionally, in the early days of TOR, false (high) bandwidths could be advertised by relays. Consequently, these (allegedly) high-performance relays were favored in path selection. The second challenge to be solved is correlation of ingress and egress traffic. These two challenges can be addressed in different ways. Consider research presented in Table 9.1 for an overview of different approaches.

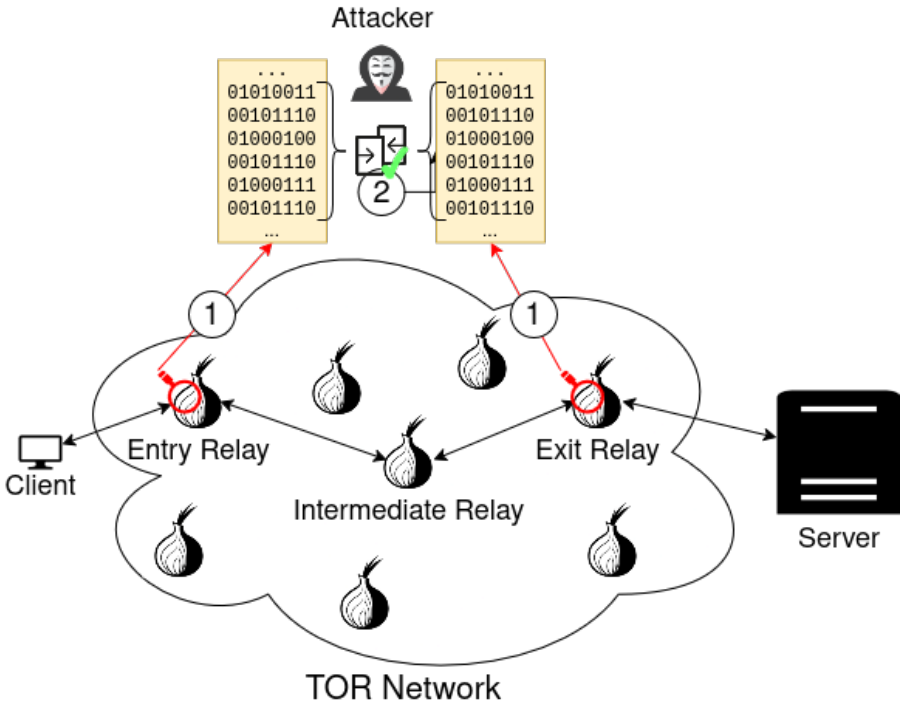


Figure 9.10: Overview of attacks using entry and exit relays.

3 In the early days of TOR, fewer relays than nowadays were deployed. Therefore, controlling a high fraction of them was a lot easier than it is today.

9.2.2 ATTACKS ON SINGLE TOR COMPONENT

Attacks from this category target a single TOR component (e.g., one of the entry, middle, and exit relays) to de-anonymize TOR users. In the following, we describe attacks that are marked as relevant in Section 9.2.

BitTorrent based attack

Le Blond et al. describe an attack that uses a malicious exit relay and the BitTorrent infrastructure to trace down TOR users [47]. The attack works as follows: When a client aims to download a file from a peer-to-peer network (like BitTorrent), they need to know which peers serve the fractions of the requested file. For this case, a BitTorrent tracker serves the information about which peers the requested file is distributed over. Requesting this list of peers is typically done in cleartext TCP, without encryption. Therefore, a malicious exit relay can manipulate this list and add a malicious peer to it. The attack also exploits the fact that, in many cases, only the request/response for the list of peers is communicated inside TOR. Hence, when a client requests a list of peers for a specific file and then connects to the listed peers (including the malicious peer), they leak their IP address to the attacker-controlled peer.

Discussion This attack falls into a category of attacks that do not exploit a flaw of TOR itself. A client rather uses TOR in a way that disables their anonymity. Hence, TOR is unable to protect against such an attack. This behavior generalizes to any situation in which a client tunnels their traffic through the TOR network but uses an application that leaks their anonymity. This attack is similar to a situation where a user leaks their identity to a website (e.g., by logging in with their credentials) that they visit via TOR. In this case, TOR still hides the client's identity from the destination. But if the client expose their identity to the destination themselves (in its communication content), TOR cannot prevent that. However, any entity observing traffic *inside* the TOR network will still be unable to link the source and destination of the communication, as this information is leaked via the application layer and encrypted by TOR.

We conclude that this attack will also be feasible against a local deployment of TOR. Our solution can not provide anonymity when a user leaks its identity via the application used, whether it is unintended, as in the setting of the described attack, or on purpose. More specifically, this attack can be executed by our defined attacker A1. This is possible as we consider A1 to have control over a set of exit relays. However, this specific attack depends on the encryption of traffic between the exit relay and the BitTorrent tracker. If traffic is encrypted, the attack is rendered infeasible, as encrypted traffic can not be altered.

Active Website Fingerprinting

Yang et al. present an attack that assumes an attacker that controls an entry relay [89]. This attack is an active website fingerprinting attack and similar to the attacks presented in Subsection 9.2.3. In fact, it is an improvement of the attack presented by He et al. [33]. Their improved methodology is able to distinguish all HTTP traffic from TOR control packets and therefore allows a higher classification accuracy. The reason why this methodology is not presented

together with the attacks in Subsection 9.2.3 is that it does not rely on a side-channel but needs to control an entry relay.

Discussion For discussion of this attack regarding its local feasibility, refer to Section 9.2.3. The only difference to attacks presented in this later section is the attacker who is able to execute this attack. As a TOR entry relay has to be controlled in order to run this attack, only A1 is able to do so.

9.2.3 SIDE-CHANNEL ATTACKS

This category includes all attacks that do not attack TOR components to break TOR's anonymity. Targets of attack could be network components such as routers or links that are used to route TOR traffic.

Passive Website Fingerprinting

The most prominent side-channel attack on TOR's anonymity is the website fingerprinting attack. An adversary performing this attack has access to the traffic between the TOR client and the entry relay. Consider Figure 9.11 for an overview of this attack type. The principle of this attack is that an adversary first creates fingerprints of a number of websites and later tries to find these fingerprints in a TOR client's traffic. Most websites nowadays do not consist of only one file but a multitude of scripts, media content, stylesheets, and more. When a user requests a website, these multiple website components result in a specific pattern of timing and request/response size for this website. These patterns are recorded as a fingerprint and stored in a database. For this case the attacker creates a fingerprint of the communication, via TOR, with a specific website.

In the next step, the adversary observes traffic between the TOR client and their entry relay. Subsequently, they compare patterns from the observed traffic to the fingerprints in their database. In this comparison/matching, the different attacks from this attack class differ. Early attacks often use basic comparison strategies, while more recent attacks make use of more advanced methods, for example, machine learning algorithms.

This attack type was initially introduced by Herrmann et al. [34]. They apply a *Multinomial Naïve Bayes Classifier* to match fingerprints. However, the achieved accuracy for website fingerprinting against TOR is only 2.96%. Furthermore, the attack is only conducted in a *closed world scenario*. This means that a victim is considered to visit websites from a set of 775 URLs. Experiments are run in this limited set of "available" websites.

In follow-up literature, the original approach is improved through more advanced classification algorithms and by considering additional features of the observed traffic, used for fingerprinting. Panchenko et al. are the first ones who extend website fingerprinting to an open-world scenario [66]. This means that in their experiments a victim is not limited to visiting the set of 775 URLs but is considered to visit arbitrary websites on the Internet. This extended attack scenario aims to identify if a victim visits a small set of "censored" URLs from a large set of randomly chosen websites. For this *open world scenario*, an accuracy of up to 73% is reached.

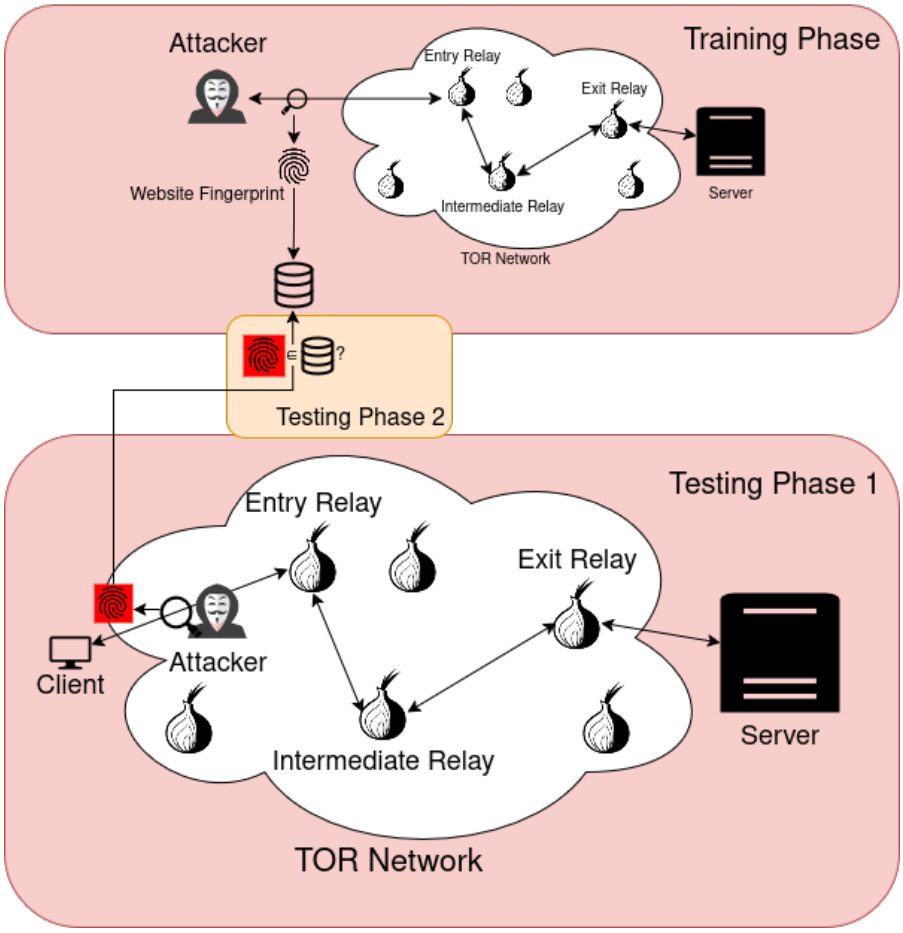


Figure 9.11: Overview of the website fingerprinting attack. Training phase: the attacker creates fingerprints of websites. Testing phase: the attacker observes network traffic between the client and entry relay and queries previously recorded fingerprints for observed traffic patterns.

Further work improves the success rate of these methods and applies them to open-world scenarios [85, 60] as well as to open-world and closed-world scenarios [11, 84, 32, 65, 92, 74, 78]. However, all these methodologies only vary in their fingerprinting algorithm and not in their general setup. Therefore, we detain from describing these specific attacks further.

In contrast to this group of similar attacks, Pulls et al. propose a novel methodology that makes use of *Website Oracles* (WO) [71]. Greschbach et al. also introduce an attack based on a WO [30]. A WO is defined as an additional information source that provides an answer (yes or no) to the question of whether a specific website was visited through TOR at a specific point in time. WOs mentioned by the authors are, for example, exit relay's DNS caches, OCSP responders, or CDNs. While some of these sources are only accessible by their providers, some offer publicly available data.

Discussion As all of these attacks follow a similar methodology and only differ in their implementation, we evaluate their feasibility against a local TOR jointly. This also holds for website fingerprinting attacks using a WO [30, 71]. This is because we argue that the same WOs can be used when performing

```
1: for r ∈ R do
2:     for i = 0 to 2 do
3:         Spoof IP packet with src = IP(W), dst = IP(r) and invalid sequence
   number
4:     end for
5: end for
```

Algorithm 9.1: Decreasing data rate between web server W and TOR exit relay.

fingerprinting in a LAN as for fingerprinting on the Internet. For evaluation, we follow the steps of the methodology of the attacks and argue for each step if and why running this step is possible/impossible in a local TOR.

The first step of running this family of attacks is the creation of fingerprints. We state that anyone who is able to participate in the local TOR as a client is also able to create fingerprints of all available websites. As our attacker model has the possibility of joining the network as a client, this step is possible for any of our three defined attackers. The next step is to observe the traffic between the client and their entry relay. Of our assumed attackers, only two, A1 and A2, are able to perform this step. Therefore, we consider attackers A1 and A2 to be able to run this attack. One must note that this attack is especially severe when performed by A2. This is because when A2 has access to the client node, they can run fingerprinting attacks against *any* of the clients.

Active Website Fingerprinting

Next to these passive attacks where an adversary only listens on communication, there also exists a number of active attacks. In these attacks an adversary actively interferes with components involved in communication.

Gilad et al.'s attack Gilad et al. present an attack that actively reduces the data rate between the exit relay and the server [28]. This reduction of traffic throughput can then be observed between the client and the entry relay. Consider a client C and a server providing a website W. The goal of an attacker is to find out whether C visits W or not. The adversary is able to sniff the client's connection to their TOR entry relay. Furthermore, they have a list R of all available TOR exit relays and their IP addresses. They now run the process presented in Algorithm 9.1. If an exit relay that has an active TCP connection with W receives an IP packet with an invalid sequence number from W, this is interpreted as a failed transmission. The exit relay acknowledges this with a double acknowledgement (ACK). After three apparently failed transmissions, the server hosting the website decreases the size of the TCP congestion window. This results in a decreased data rate. Thus, the data throughput between W and the exit relay also decreases. This reduction of data rate is also observable between the entry relay and the client. Hence, the attacker concludes that C communicates with W.

Discussion of Gilad et al.'s attack The first condition that an attacker in a local TOR has to satisfy, in order to run this attack, is to sniff the TOR ingress traffic. A second condition for an attacker is to possess a list of exit relays in the network. As an attacker can participate in the local TOR as a client, they also

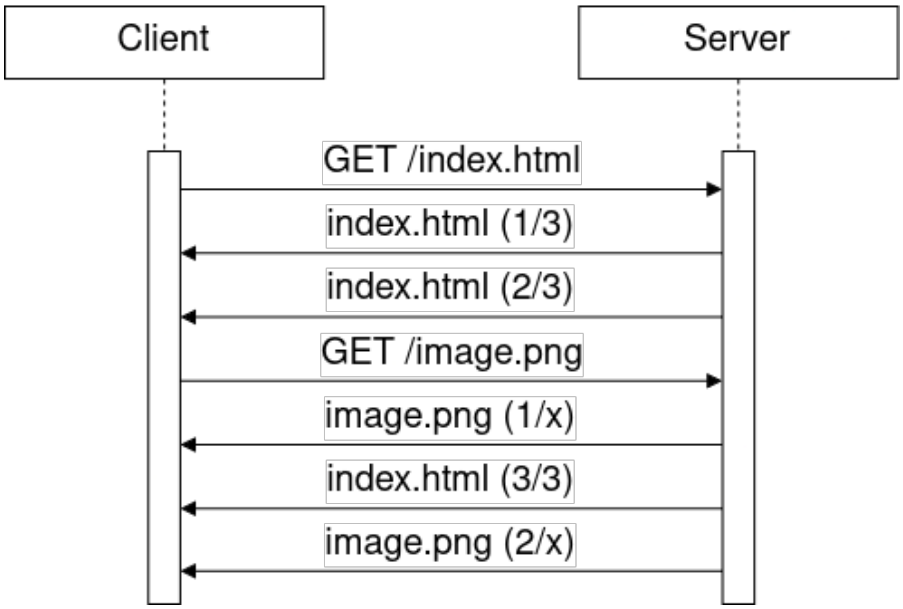


Figure 9.12: Overlapping requests/responses of different resources during client/server communication.

have access to such a list. The third condition for an adversary is to spoof TCP packets. We argue that an attacker in a local TOR can also fulfill this requirement. Therefore, we state that an attacker in the local TOR can run this attack. More specifically, A1 and A2 are able to run this kind of attack, as they have access to the TOR ingress traffic. Note that also when running this attack, A2 can attack any of the clients.

He et al.'s attack He et al. also propose an active website fingerprinting attack [33]. They state that one weakness of existing attacks is that some resources that are part of fingerprints are transferred overlapping. Consider a client requesting a website from a server. They first request the `index.html`. When parsing an embedded resource, for instance `image.png` in the `index.html`, they also request `image.png` from the server. As objects usually cannot be sent in a single IP packet, they get split for transfer. Hence, this results in a one-packet request and multiple packets as a response. This results in a non-sequential packet pattern between client and server. Consider Figure 9.12 for an example.

This overlapping behavior of requests/responses leads to a decreased performance of website fingerprinting attacks. He et al.'s methodology aims to delay requests to a web server so that each request and its related responses are executed sequentially. This packet delay is induced between the client and the entry relay. With their approach, they are able to outperform existing, passive website fingerprinting attacks by 16.5% in accuracy.

Discussion of He et al.'s attack The basic principle of this attack is similar to passive website fingerprinting attacks. Therefore, the only additional condition that has to be satisfied by an attacker is that they are not only able to sniff TOR ingress traffic but that they can also actively manipulate it. As our attacker types

do not make a difference between sniffing and manipulating traffic, we state that this attack can be performed by the same attacker as all previously discussed passive website fingerprinting attacks. Therefore, A1 and A2 are able to run this attack, with A2 being able to target any of the clients when having control over the client node.

Arp et al.'s attack Another active attack that also applies website fingerprinting is proposed by Arp et al. [2]. Their approach improves the detectability of websites by implanting *web page markers* in the victim's browser. In this attack, the attacker needs access to the victim's ingress traffic to TOR. Furthermore, they require a way to trigger the execution of JavaScript snippets in the victim's browser. As websites often embed content from third parties, this can be achieved by controlling such content. The injected code snippet can then issue unique patterns of HTTP requests towards an arbitrary destination. This destination does not have to be controlled by the attacker, as the pattern of requests will be visible to the attacker in the TOR ingress traffic anyway. Their approach is able to detect 91% of web pages, marked with web page markers, among a set of 60,000 web pages.

Discussion of Arp et al.'s attack We state that this attack is equivalent to passive website fingerprinting attacks. The only condition that has to be satisfied additionally is for the adversary to embed JavaScript code into a webpage that is visited by the victim. We do not discuss this challenge here, as it is part of the general attack and not an issue related to the local deployment of TOR. Therefore, we conclude that this attack itself has the same feasibility in a local TOR as in the global TOR and can be conducted by A1 and A2, with high impact when being run by A2. For information regarding its general effectiveness, refer to the original paper [2].

Attack based on manipulated USB charger

Next to fingerprinting attacks, there exists one attack that uses a side-channel from a totally different category: Yang et al. present an attack in which they instrument USB chargers (for charging phones) to find out which web pages a device visits using TOR [90]. In their setup, they access a set of 50 different web pages and 50 hidden services with a smartphone, using the TOR mobile browser *Orfox*[4]. While web pages are accessed, they collect power traces from the USB charging device connected to the smartphone. These traces of the smartphone's power consumption are used to train a random forest classifier. With this classifier they are able to re-identify web pages with an accuracy of 76.3% and hidden services with an accuracy of 87.3%. Due to the size of the datasets (only 50 web pages/hidden services), they consider their work as a proof of concept.

Discussion We include this attack in our results as, formally speaking, the required TOR components (which are none) are under the control of all our

4 At the time of the study, Orfox was used to access TOR with a smartphone. With the release of 'Tor Browser for Android', Orfox was retired. See https://support.torproject.org/tormobile/tormobile-5/.

defined attackers. However, it is questionable if the given attackers are able to run malicious USB chargers or are able to manipulate existing ones. Therefore, we argue that this attack is infeasible in a local TOR for all defined attackers.

9.2.4 HYBRID ATTACKS

This section presents attacks that are based on a mix of components from the last three categories. Again, only attacks that are considered relevant in Section 9.2 are introduced.

Attacks based on network latency measurements

Attacks presented in the following all build on performing measurements in the network.

Hopper et al.'s attack An early method that sets the basis for later attacks is proposed by Murdoch et al. [56]. They assume that an attacker has control over one arbitrary TOR relay and that the adversary is able to run a web server. Let v be the victim client and s a malicious web server. v must lie in the set of clients that access s, and v's entry relay must be under the control of the attacker. When a connection between v and s is established, the adversary embeds a unique traffic pattern in the traffic from s to v. Subsequently, this traffic pattern can then be observed at all relays that are part of the circuit between v and s. For this purpose, the authors use the circumstance that any TOR relay can measure shifts in the throughput rate of any other relay, even if it does not have access on its network data. This is done by building a circuit from the attacker-controlled TOR relay (r) through the relay that is to be probed. Through this circuit a constant *probe traffic* is sent. If the traffic pattern that is present in the traffic from s to v passes the targeted relay, the latency of the probe traffic (which is observable at r) reflects the implanted traffic pattern. By conducting this procedure against any TOR relay in the network, the nodes included in the circuit between v and s can be identified.

Hopper et al. build on this methodology of identifying relays in a circuit. Using a malicious web server (s), they measure the Round Trip Time (RTT) of a TOR circuit [36]. Based on the RTTs of multiple circuits, they are able to tell which circuits are equal and which are not. This enables them to give a statement if two traffic streams are originated by the same TOR client.

By applying their and the aforementioned methodology [56], they are able to learn the relays involved in a circuit as well as the circuit's RTT. Let these discovered relays be A, B, and C, and the victim's TOR client be v. Hence, the victim's circuit to the malicious server (s) is: $v \rightarrow A \rightarrow B \rightarrow C \rightarrow s$. As the circuit's relays are known to the attacker, they can build this circuit (with their own TOR client a) itself. The attacker's circuit is therefore: $a \rightarrow A \rightarrow B \rightarrow C \rightarrow s$. In the following, the RTT from a client x to a server/relay y is denoted as RTT_{xy}. Running measurements on TOR, the authors find out RTT_{vs}. They also find out RTT_{as} (as explained above). These RTTs resolve to:

$$RTT_{vs} = RTT_{vA} + RTT_{AB} + RTT_{BC} + RTT_{Cs}$$
$$RTT_{as} = RTT_{aA} + RTT_{AB} + RTT_{BC} + RTT_{Cs}$$

Furthermore, the attacker is able to measure RTT_{aA} as they control a. To calculate RTT_{vA} (the RTT from the victim to their entry relay), the adversary solves the following equation:

$$RTT_{vA} = RTT_{vs} - (RTT_{as} - RTT_{aA})$$

RTT_{vA} gives information on the network position of the victim. Using *network coordinate systems*[5], the adversary can drastically reduce the set of possible clients and might be able to find out the victim's identity.

Discussion of Hopper et al.'s attack The two conditions that have to be satisfied by an attacker in order to run this attack in a local TOR are 1) running a web server (which needs to be visited by the victim) and 2) controlling an arbitrary TOR relay, which is used for probing. We state that it is possible for any of the three attackers (A1, A2, A3) to run a web server. Using a TOR relay for probing is only possible for A1. This is because A1 is the only one of our considered attackers who has access to TOR relays. Therefore, the first step of this attack, which is to identify the circuit relays, is possible for A1. However, the step of identifying the victim is infeasible in a local setting. Recall the topology of our proposed deployment of TOR (Chapter 7). All clients are connected to the local TOR network via the same router, but they are connected to this router via multiple switches. Different clients might be connected to the router via different switches. Therefore, clients are also connected to the router via different routes, which can also result in individual RTTs for each client. However, there are multiple arguments that make this attack infeasible in a local context. First, we argue that switching in contrast to routing, is more dynamic, especially when multiple switches are deployed. Hence, the client's RTTs will not be static. Secondly, we argue that the attacker has no insight into the deployed infrastructure as there exists no network coordinate system in a LAN. Consequently, an attacker cannot use RTT as a metric to identify clients in a local setting. Therefore, none of the considered attackers (A1, A2, A3) is able to run this attack. Note that the feasibility of this attack highly depends on the given infrastructure. Thus, when transferring our conclusion to a different network, the situation should be re-evaluated.

Chakravarty et al.'s attack Chakravarty et al. propose an attack that employs '*single-end controlled* available bandwidth estimation tools' [13]. This means that an attacker induces a traffic pattern into TOR traffic and tries to observe it throughout the TOR network. For this case the adversary uses tools that are able to measure fluctuations in bandwidth from a single point in the network rather than between two specific points. In order to run this attack, an adversary needs to control a server that the victim connects to. This can, for instance, be a malicious web server. When the victim connects to the malicious server, the requested content is delivered as usual. During this process the server also embeds multiple small chunks of additional data or a single large one. These patterns are aimed to be observed in the network later.

Next to inducement of traffic patterns, the adversary needs multiple points on the Internet from which they can run measurement tools. The more spread

5 Network coordinate systems are used to predict network latencies between nodes in a network. Therefore, with one known node and a given RTT, a set of nodes that have similar RTTs to the given node can be identified.

these points are over the Internet, the more precise the measurements are. This is because links between different servers, ASes[6], and other Internet-level components can be measured better the closer the measurement point is to the link. For measurements, the authors mainly rely on their own tool called *LinkWidth* [12]. Running these steps, they are able to track traffic patterns down to the regarding circuit's entry relays. It remains the challenge of tracking circuits further to their initiating client's devices. For this case, ISP-level information is required. The authors try to tackle this goal by using ISP router maps. One example of such a resource is *Rocketfuel* [79]. Overall, using their methodology, the authors are able to identify all three relays in 11 out of 50 circuits that they aimed to track.

Discussion of Chakravarty et al.'s attack In the following, we argue why this attack is infeasible in a locally deployed TOR. The first challenge for an adversary is to run a web server that is connected to by a client. As we stated already for previous attacks, we assume that all our defined attackers are able to do so. This server can subsequently be used to induce traffic patterns towards the client. In principle, we assume that attackers can participate in the network. However, as outlined in Section 7.1, the ability to perform measurements in the network depends on the exact deployment of TOR. If clients are forced to use TOR in the LAN, measurements can no longer be performed as proposed in the original paper. Otherwise, an adversary can identify via which network nodes the marked traffic is routed. Hence, they can also identify the network node to which the entry relay is connected. We argue that from this point on the attack is not feasible anymore. This is for the same reason as discussed for the previous attack in Section 9.2.4: latency from a client device to the entry relay is not static. However, the attacker is still able to identify the entire circuit path of their victim. We reason that this is not a problem because it is not a secret in which network the victim is a part of. We state that in order to identify the client, the attacker additionally needs access to the entry relay of the victim's circuit. With this additional constraint satisfied, this attack would be feasible in a local TOR. Note that in this case the attack would be almost equal to Murdoch et al.'s attack [56]. In this case only the method of measuring the traffic pattern would differ. Without access to entry relays, this attack is infeasible for any of the three defined attackers.

Ultrasound-based attack

A last attack, which stems from a totally different field, is presented by Mavroudis et al. [52]. They use an *ultrasound ecosystem* as a side-channel to de-anonymize TOR clients. At the time of their study, the authors found that ultrasounds are used by ad providers to track users across different devices. This works because ultrasounds are inaudible for humans. Additionally, they can be played via normal device speakers and perceived by the device's microphones [52]. Consider a user surfing the Internet via TOR. A malicious website can now embed a so-called *ultrasound beacon* (a short ultrasound sequence encoding a unique string), which is then emitted by the user's browser. Chances are high that another user device is present that *does not use TOR for communication*. This device can capture the

6 An *Autonomous System* (AS) is a large-sized network on the Internet that is typically controlled by a single organization. It furthermore consists of large IP prefixes.

ultrasound beacon and send it to the adversary. As this second communication process is not performed via TOR, the adversary learns the user's real identity.

Discussion The idea behind this attack falls into the same category as the attack utilizing the BitTorrent network. The adversary does not exploit an insecurity in TOR itself but circumvents TOR by using a non-anonymized channel. As the feasibility of this attack does not depend on any TOR component, we infer that this attack also works in a local TOR. Its success rate and possible countermeasures are equivalent to the original attack in the global TOR. Therefore, all three attackers A1, A2, and A3 are able to run this attack.

General Observations

After discussing each of the attacks, we find that out of 29 listed attacks, 18 are eliminated during pre-selection. This is because they do not satisfy the formal assumptions. A1 (in control of one network node and the corresponding family of relays) is the strongest of our attackers, as they are able to run 8 out of 11 discussed attacks. A2 (with access to network traffic of one of the intermediate nodes or the `client_node`) has the capabilities to run 5 of the discussed attacks, whereas A3 (with access to network traffic of the `gateway`) is not able to perform a single one. It should also be highlighted that A2, when in control of the client node, has an extraordinarily high impact: they are able to run website fingerprinting against *any* of the clients in the network.

10 DISCUSSION

In the following we discuss results, presented in Chapter 9. We start with discussion of performance measurements and continue with results of the anonymity analysis.

10.1 PERFORMANCE OF LOCAL TOR

A first question that is raised when reviewing simulation results from Section 9.1 is whether measurements are realistic and reliable. We assume that the used simulation framework itself produces realistic data. This is shown by Jansen and Hopper for simulations of TOR on the Internet [40]. We argue that the only difference between the Internet and a LAN, as defined in Section 2.6, is the complexity of the network. This includes parameters such as the number of routers, subnetworks, and connections between networks, as well as performance properties (packet loss, data rates) of network infrastructure. All of these parameters were chosen based on information from experts in network administration. We argue that these parameters are therefore realistic. The last component of simulations that must be discussed is synthetically generated traffic. We used the tool TGen to generate traffic (see Chapter 8 for details). TGen itself is again used in simulations of TOR on the Internet. As these simulations are compared and validated against measurements on the Internet, we assume that TGen produces realistic packets and streams. We used traffic models that were also applied during simulations of TOR on the Internet. However, traffic patterns we generated using TGen are less diverse as we only used a single gateway node, which acted as a server. One could argue that this simplification, which we made to simplify our setup, differs from 'natural' traffic in the wild. Nevertheless, traffic throughput of synthetically generated traffic is equal to typical throughput in a LAN. All in all, we state that the generated traffic is not as diverse as it is in the wild. Still, it is realistic enough to produce results that are reliable and can be used to conclude on TOR's behavior in a LAN.

Considering our results, one can observe a trade-off between anonymity and performance when implementing TOR in a LAN. We expected that providing performant infrastructure in the form of TOR relays would fix TOR's known performance issues. We were able to show, that in the case of TOR, performance can not be reached by simply providing more and performant infrastructure. Nevertheless, there are a number of valuable insights we gained during simulations and evaluation. We claim that these findings are not only relevant for TOR in a local network but also for the TOR project on the Internet. We acknowledge that TOR uses more of its potential with a congestion algorithm being implemented than in the past, without congestion control. However, to use its full capacity, further steps have to be taken. These are, based on our experience, correcting the underestimation of available bandwidth of TOR Vegas and implementing an improved algorithm for building TOR circuits, which distributes the load more evenly over all available relays. The latter challenge has already been addressed

with mechanisms such as *sbws*[1]. TOR's sbws is a bandwidth scanner, which can provide bandwidth measurements from the TOR network. Bandwidth estimations can then be taken into consideration by directory authorities to weight the possibility of being chosen in a circuit for each relay. However, additional work in this direction shows that proposed algorithms do not solve the challenge sufficiently [23].

We are aware, that TOR is not the only system that can be used to address the challenge of anonymity in LANs. We also hypothesize that TOR is a 'heavy' solution, bringing an extensive additional infrastructure to a network. More lightweight approaches might still be able to provide a sufficient level of anonymity without a high overhead of new infrastructure and additional performance cost. Nevertheless, with TOR's discussed challenges tackled, we would expect a higher performance, rendering a local deployment of TOR a potential solution for anonymous communication in LANs.

10.2 ANONYMITY OF LOCAL TOR

One general observation regarding possible attacks on the local TOR is that there are three types of attacks that are possible to run. The first type is website fingerprinting. This includes passive attacks as well as active attacks, which induce patterns into traffic. A similar type of attack, which also induces patterns into traffic, aims to identify the circuit between the victim and their destination. The last type of attack exploits not TOR itself but influences the victim or their device so that they use a channel outside of TOR, which leaks their identity. It sticks out that, out of 29 listed attacks, only 11 remain after pre-selection. There are several reasons for that. A first reason is that a number of attacks do not target the client's anonymity and are therefore irrelevant. Additionally, through the deployment of relays in families, an entire group of attacks is eliminated: Attacks that need to access an entry and an exit relay are impossible to run. This is contrary to the situation in the global TOR. Especially in TOR's early days, attacks using entry and exit relays were common due to the small amount of available relays. Due to the organizational measures, this is not possible in the local deployment. Another great difference from the global TOR is the location of clients using TOR. On the Internet, clients are spread over a magnitude of different networks. Hence, identification of a client's entry relay already drastically decreases the group of candidates. Not so in a local deployment: As all clients are located in the same network, identification of a client's circuit relays does not allow any conclusions on the client's identity. Therefore, all attacks that aim to identify circuit relays are not critical in a local TOR. We state, based on our results, that deploying TOR to a LAN strongly limits the set of possible attacks. Nevertheless, the remaining attacks might be even more critical in a local context. This also leads to an observation that raises our attention. Attacks that can be run by A1 and A2, having access to one intermediate node or even directly to a family of relays, pose a severe risk. The reason for this is that there are only three families of relays deployed. As circuits always contain three relays, and relays *must* stem from separate families, each circuit contains relays from

1 The tool sbws is actively used by the TOR project to assess network health and enhance traffic distribution over relays. More information can be found at https://tpo.pages.torproject.net/network-health/sbws/.

the three families. Hence, we argue that each of the three intermediate network nodes observes a third of the traffic going into the local TOR network. This means that an attacker having access to one of the intermediate network nodes or the relays directly can run attacks against a third of the circuits. Controlling an intermediate relay or, as shown in our results, the connected relay family in the local TOR therefore allows running attacks against a much higher fraction of clients at the same time than in the global TOR network. This behavior gets even more severe if an attacker, in our case A2, gets control over the client node. This enables A2 to attack any of the present circuits, as all of them are routed via the client node. The given observation raises the question if the client node as a single point of control must be eliminated in order to provide an increased level of anonymity. This question should be subject of future work. In contrast to fingerprinting attacks, which highly benefit from the client node as a single control point, other attacks are limited by this simple network structure. Attacks that aim to de-anonymize TOR clients based on network measurements are rendered infeasible by our deployment. This must be considered when applying changes to the proposed infrastructure. All in all, we conclude that the provided anonymity of TOR is not limited through a local deployment. However, one must tackle a number of special challenges in a local context.

11 CONCLUSION, LIMITATIONS, AND FUTURE WORK

The following chapter concludes this thesis. We revisit the research questions given in Chapter 1 and provide conclusive thoughts on each of them. Subsequently, limitations of this work are shown. Finally, we show future research directions and open questions regarding our niche, which we were not able to address in the scope of this thesis.

11.1 CONCLUSION

In the following, we summarize our contributions, based on our research questions.

Which existing techniques can be adapted to enable anonymous communication in a LAN as well as help red teamers to stay undetected after infiltrating a network? Our first research question aimed to reveal possible linkages between two disjunct fields of research: red teaming and anonymity in networks. To answer this question, we investigated existing work and definitions. A first evidence in literature suggested that challenges from these fields fall into separate problem categories [39]: the challenge in red teaming is to hide the fact that communication is taking place. The goal of ACNs is to hide communication parties. Further research showed how these challenges can be classified into a hierarchical system of problems regarding information hiding [44]. Based on this hierarchy, it was shown that a system that provides anonymity in red teaming can also provide sufficient anonymity in LANs. However, such a system *can not be implemented* without violating additional constraints that are required in LANs. These are namely low latency and high throughput. Consequently, we came to the conclusion, that not both of the problems can be tackled jointly. Hence, we decided to focus on the problem of anonymity in LANs. In order to evaluate our solution later in this thesis, we defined the term anonymity and introduced our attacker model.

What can a prototype of such a LAN look like, and how can it be implemented? To answer the second research question, we investigated existing literature. We found that we are among the first ones to solve the problem of anonymous communication regarding the specific network architecture found in LANs, as defined in Section 2.6. There is a large amount of research aimed at providing anonymity on the Internet. In order to draft a first solution for local networks, we looked into existing literature. Subsequently, we presented multiple categories of implemented approaches. Based on these approaches, we presented two candidate concepts and further discussed their suitability for our purpose. We chose to implement a local ACN and selected TOR as our concept. Our primary reason for choosing TOR was that it is well tested and actively maintained. Furthermore, its design goals align with ours and seek to provide anonymity as well as low latency. We deployed TOR to a LAN, which we imple-

mented in a network simulator. Simulations were used to tackle our last research question.

Does this prototype solve the problem sufficiently? Our evaluation, which we conducted to test our approach and answer our last research question, was twofold: in a first step, we evaluated the cost in terms of performance at which anonymity can be provided in a LAN. Using an ACN certainly results in a loss of performance. We were able to show that TOR slows down communication in a network, even when a sufficient amount of resources is provided. Known performance issues of TOR were reproduced in multiple simulations. In a LAN with 6000 active clients and 240 relays, each providing a bandwidth of up to 600 Mbit/s, 65% of clients download a file of 2.5 MB in less than one second. However, the rest of the clients need up to 110 s to download the same file. We therefore conclude that, in its current state of development, TOR does not provide sufficient performance for productive operation in a LAN. Regardless of the unfavorable results in terms of performance, we are able to contribute novel insights: we identified TOR's congestion algorithm, TOR Vegas, as one source for the unused potential of performant infrastructure. TOR Vegas heavily underestimates the available bandwidth of circuits in TOR. Furthermore, an even distribution of load over all available relays in the network must be investigated further.

Next to performance results, we also evaluated the level of anonymity that is provided by TOR in a LAN. For this case, we first argued why deploying TOR in a LAN does not violate the original assumptions made by the authors of TOR. In the next step, we gave an overview of existing attacks on the global TOR. We systematically went through these attacks and argued, for each one, if and why a local TOR is vulnerable to each attack under our attacker model. Out of 29 listed attacks, only 11 are viable against TOR in a LAN. These stem from three attack categories. The first and largest group of feasible attacks is website fingerprinting. A second (single) attack induces a pattern into TOR traffic, which is then observed throughout the victim's circuit. A third class of attacks aims to trick a victim into leaking their identity to a second communication channel that is not anonymized by TOR. From our defined attackers, only A1, an adversarial administrator, is able to run all of them. Hence, de-anonymization and abuse by an administrator are the greatest threats to anonymity in our solution. Furthermore, an attacker observing a network node routing any of the client's (encapsulated) traffic to the next relay has great power, as they are able to run a subset of the feasible attacks against *any* of the clients. We still conclude that TOR provides a sufficient level of anonymity in a LAN, as attacks are strongly restricted in contrast to a situation without TOR deployed. Furthermore, feasible attacks require a high investment by the attacker and can only be executed to answer the question of whether a victim visits a certain website. Attacks that unravel all of a victim's communication parties are not feasible.

11.2 LIMITATIONS AND FUTURE WORK

We close this thesis with an overview of its limitations and further ideas that should be subjects of future research.

One topic we aimed to address in this thesis was a solution for red teamers to communicate without being detected in an infiltrated network. As this topic and our second domain differ more than we expected at the beginning of this thesis, we were not able to address both of them.

In Section 9.2 we find that our deployment of TOR has one node at which clients are connected to the network, which allows an adversary to run a specific set of attacks against any of the clients. We argue that when implementing TOR in a productive LAN, one can slightly adjust the topology to prevent this.

Further limitations regard TOR itself. One drawback of TOR is that it only runs single-threaded. To cope with that, one can run as many relays as cores are available, when using a multicore CPU. However, there is a re-implementation of TOR in progress which seeks to fix this behavior. This re-implementation is called *Arti*[1]. We suppose that further research could examine Arti regarding its suitability for a local deployment. Another limitation of TOR is that it only supports TCP as a communication protocol. However, with the emergence of HTTP/3[2] which builds on UDP, its adoption into TOR poses an important task. In fact, there already exists an accepted proposal for UDP support in TOR [50].

Future work should also investigate performance enhancements of TOR. A first starting point should be TOR Vegas and its problem of unused relay bandwidth. There exist various parameters in TOR Vegas that can be tuned to enhance performance. Furthermore, the second problem we inferred in Section 9.1, distribution of load, should be addressed. We suppose that by evenly distributing the load over all relays, the infrastructure can be more effectively utilized. We also suppose that in a local TOR, where guarantees on available infrastructure can be given, circuits that have low performance could be discarded and directly replaced with more performant circuits.

Next to research into TOR itself, other approaches for anonymity in LANs should also be investigated. The closest related work to this thesis is PriFi (see Chapter 5), which builds on DCNets. However, PriFi is limited to a specific amount of clients. Future work could aim to tackle this limitation. Additionally, one can use our overview of existing solutions for the Internet to draft new solutions for local networks.

1 See Arti's official website at https://tpo.pages.torproject.net/core/arti/ for more information.

2 In fact, in May 2025, already 30% of HTTP requests on the Internet use HTTP/3 [18].

BIBLIOGRAPHY

[1] Timothy G. Abbott et al. "Browser-Based Attacks on Tor". In: *Privacy Enhancing Technologies*. Springer Berlin Heidelberg, 2007, pp. 184–199. ISBN: 9783540755500. URL: http://dx.doi.org/10.1007/978-3-540-75551-7_12.

[2] Daniel Arp, Fabian Yamaguchi, and Konrad Rieck. "Torben: A Practical Side-Channel Attack for Deanonymizing Tor Communication". In: *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*. ASIA CCS '15. ACM, Apr. 2015. URL: http://dx.doi.org/10.1145/2714576.2714627.

[3] Michael Backes et al. "AnoA: A Framework for Analyzing Anonymous Communication Protocols". In: *2013 IEEE 26th Computer Security Foundations Symposium*. IEEE, June 2013, pp. 163–178. URL: http://dx.doi.org/10.1109/CSF.2013.18.

[4] Ludovic Barman et al. "PriFi: Low-Latency Anonymity for Organizational Networks". In: *Proceedings on Privacy Enhancing Technologies* 2020.4 (Aug. 2020), pp. 24–47. ISSN: 2299-0984. URL: http://dx.doi.org/10.2478/popets-2020-0061.

[5] Kevin Bauer, Dirk Grunwald, and Douglas Sicker. "Predicting Tor path compromise by exit port". In: *2009 IEEE 28th International Performance Computing and Communications Conference*. IEEE, Dec. 2009, pp. 384–387. URL: http://dx.doi.org/10.1109/PCCC.2009.5403852.

[6] Kevin Bauer et al. "Low-resource routing attacks against tor". In: *Proceedings of the 2007 ACM workshop on Privacy in electronic society*. CCS07. ACM, Oct. 2007. URL: http://dx.doi.org/10.1145/1314333.1314336.

[7] Kevin Bauer et al. "On the Optimal Path Length for Tor". In: *3rd Hot Topics in Privacy Enhancing Technologies*. July 2010. URL: https://nikita.ca/papers/tor-pathlen-hotpets10.pdf.

[8] Krista Bennett and Christian Grothoff. "gap – Practical Anonymous Networking". In: *Privacy Enhancing Technologies*. Springer Berlin Heidelberg, 2003, pp. 141–160. ISBN: 9783540409564. URL: http://dx.doi.org/10.1007/978-3-540-40956-4_10.

[9] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. "Web MIXes: A System for Anonymous and Unobservable Internet Access". In: *Designing Privacy Enhancing Technologies*. Springer Berlin Heidelberg, 2001, pp. 115–129. ISBN: 9783540447023. URL: http://dx.doi.org/10.1007/3-540-44702-4_7.

[10] A. Biryukov, I. Pustogarov, and R. Weinmann. "Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization". In: *2013 IEEE Symposium on Security and Privacy*. IEEE, May 2013, pp. 80–94. URL: http://dx.doi.org/10.1109/SP.2013.15.

[11] Xiang Cai et al. "Touching from a distance: website fingerprinting attacks and defenses". In: *Proceedings of the 2012 ACM conference on Computer and communications security*. CCS'12. ACM, Oct. 2012. URL: http://dx.doi.org/10.1145/2382196.2382260.

[12] Sambuddho Chakravarty, Angelos Stavrou, and Angelos Keromytis. *LinkWidth: A Method to Measure Link Capacity and Available Bandwidth using Single-End Probes*. Tech. rep. Jan. 2008. URL: https://www.academia.edu/45294099/LinkWidth_A_Method_to_Measure_Link_Capacity_and_Available_Bandwidth_using_Single_End_Probes.

[13] Sambuddho Chakravarty, Angelos Stavrou, and Angelos D. Keromytis. "Traffic Analysis against Low-Latency Anonymity Networks Using Available Bandwidth Estimation". In: *Computer Security – ESORICS 2010*. Springer Berlin Heidelberg, 2010, pp. 249–267. ISBN: 9783642154973. URL: http://dx.doi.org/10.1007/978-3-642-15497-3_16.

[14] Sambuddho Chakravarty et al. "On the Effectiveness of Traffic Analysis against Anonymity Networks Using Flow Records". In: *Passive and Active Measurement*. Springer International Publishing, 2014, pp. 247–257. ISBN: 9783319049182. URL: http://dx.doi.org/10.1007/978-3-319-04918-2_24.

[15] David Chaum. "The dining cryptographers problem: Unconditional sender and recipient untraceability". In: *Journal of Cryptology* 1.1 (Jan. 1988), pp. 65–75. ISSN: 1432-1378. URL: http://dx.doi.org/10.1007/BF00206326.

[16] David L. Chaum. "Untraceable electronic mail, return addresses, and digital pseudonyms". In: *Communications of the ACM* 24.2 (Feb. 1981), pp. 84–90. ISSN: 1557-7317. URL: http://dx.doi.org/10.1145/358549.358563.

[17] Chen Chen et al. "HORNET: High-speed Onion Routing at the Network Layer". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. CCS'15. ACM, Oct. 2015, pp. 1441–1454. URL: http://dx.doi.org/10.1145/2810103.2813628.

[18] *Cloudflare Radar*. URL: https://radar.cloudflare.com/adoption-and-usage (visited on 05/23/2025).

[19] Henry Corrigan-Gibbs, Dan Boneh, and David Mazieres. "Riposte: An Anonymous Messaging System Handling Millions of Users". In: *2015 IEEE Symposium on Security and Privacy*. IEEE, May 2015, pp. 321–338. URL: http://dx.doi.org/10.1109/SP.2015.27.

[20] Henry Corrigan-Gibbs and Bryan Ford. "Dissent: accountable anonymous group messaging". In: *Proceedings of the 17th ACM conference on Computer and communications security*. CCS '10. ACM, Oct. 2010, pp. 340–350. URL: http://dx.doi.org/10.1145/1866307.1866346.

[21] Henry Corrigan-Gibbs, David Isaac Wolinsky, and Bryan Ford. "Proactively accountable anonymous messaging in verdict". In: *Proceedings of the 22nd USENIX Conference on Security*. SEC'13. Washington, D.C.: USENIX Association, 2013, pp. 147–162. ISBN: 9781931971034. URL: https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/corrigan-gibbs.

[22] G. Danezis, R. Dingledine, and N. Mathewson. "Mixminion: design of a type III anonymous remailer protocol". In: *Proceedings of the 19th International Conference on Data Engineering*. SECPRI-03. IEEE Comput. Soc, pp. 2–15. URL: http://dx.doi.org/10.1109/SECPRI.2003.1199323.

[23] Hussein Darir et al. "MLEFlow: Learning from History to Improve Load Balancing in Tor". In: *Proceedings on Privacy Enhancing Technologies* 2022.1 (Nov. 2021), pp. 75–104. ISSN: 2299-0984. URL: http://dx.doi.org/10.2478/popets-2022-0005.

[24] Roger Dingledine, Nick Mathewson, and Paul Syverson. "Tor: the second-generation onion router". In: *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*. SSYM'04. San Diego, CA: USENIX Association, 2004, p. 21. URL: https://www.usenix.org/conference/13th-usenix-security-symposium/tor-second-generation-onion-router.

[25] Michael J. Freedman et al. "Introducing Tarzan, a Peer-to-Peer Anonymizing Network Layer". In: *Peer-to-Peer Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 121–129. ISBN: 978-3-540-45748-0. URL: https://link.springer.com/chapter/10.1007/3-540-45748-8_12.

[26] Xinwen Fu and Zhen Ling. "One Cell is Enough to Break Tor's Anonymity". In: *Black Hat DC 2009*. Feb. 2009. URL: https://www.blackhat.com/presentations/bh-dc-09/Fu/BlackHat-DC-09-Fu-Break-Tors-Anonymity.pdf.

[27] John Geddes, Rob Jansen, and Nicholas Hopper. "How Low Can You Go: Balancing Performance with Anonymity in Tor". In: *Privacy Enhancing Technologies*. Springer Berlin Heidelberg, 2013, pp. 164–184. ISBN: 9783642390777. URL: http://dx.doi.org/10.1007/978-3-642-39077-7_9.

[28] Yossi Gilad and Amir Herzberg. "Spying in the Dark: TCP and Tor Traffic Analysis". In: *Privacy Enhancing Technologies*. Springer Berlin Heidelberg, 2012, pp. 100–119. ISBN: 9783642316807. URL: http://dx.doi.org/10.1007/978-3-642-31680-7_6.

[29] Sharad Goel et al. *Herbivore: A scalable and efficient protocol for anonymous communication*. Tech. rep. Cornell University, 2003. URL: https://hdl.handle.net/1813/5606.

[30] Benjamin Greschbach et al. "The Effect of DNS on Tor's Anonymity". In: *Proceedings 2017 Network and Distributed System Security Symposium*. NDSS 2017. Internet Society, 2017. URL: http://dx.doi.org/10.14722/ndss.2017.23311.

[31] Christian Grothoff et al. *GNET*. Tech. rep. Version 0.5.2. GNUnet e.V., June 2002. URL: https://git.gnunet.org/bibliography.git/plain/docs/main.pdf.

[32] Jamie Hayes and George Danezis. "k-fingerprinting: A Robust Scalable Website Fingerprinting Technique". In: *Proceedings of the 25th USENIX Conference on Security Symposium*. SEC'16. Austin, TX, USA: USENIX Association, 2016, pp. 1187–1203. ISBN: 9781931971324. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/hayes.

[33] Gaofeng He et al. "A novel active website fingerprinting attack against Tor anonymous system". In: *Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, May 2014. URL: http://dx.doi.org/10.1109/CSCWD.2014.6846826.

[34] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. "Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier". In: *Proceedings of the 2009 ACM workshop on Cloud computing security*. CCS '09. ACM, Nov. 2009. URL: http://dx.doi.org/10.1145/1655008.1655013.

[35] Jelle van den Hooff et al. "Vuvuzela: scalable private messaging resistant to traffic analysis". In: *Proceedings of the 25th Symposium on Operating Systems Principles*. SOSP '15. ACM, Oct. 2015, pp. 137–152. URL: http://dx.doi.org/10.1145/2815400.2815417.

[36] Nicholas Hopper, Eugene Y. Vasserman, and Eric Chan-Tin. "How much anonymity does network latency leak?" In: *Proceedings of the 14th ACM conference on Computer and communications security*. CCS07. ACM, Oct. 2007, pp. 82–91. URL: http://dx.doi.org/10.1145/1315245.1315257.

[37] Alfonso Iacovazzi, Daniel Frassinelli, and Yuval Elovici. "The DUSTER Attack: Tor Onion Service Attribution Based on Flow Watermarking with Track Hiding". In: *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*. Chaoyang District, Beijing: USENIX Association, Sept. 2019, pp. 213–225. ISBN: 978-1-939133-07-6. URL: https://www.usenix.org/conference/raid2019/presentation/iacovazzi.

[38] Alfonso Iacovazzi, Sanat Sarda, and Yuval Elovici. "Inflow: Inverse Network Flow Watermarking for Detecting Hidden Servers". In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. IEEE, Apr. 2018, pp. 747–755. URL: http://dx.doi.org/10.1109/INFOCOM.2018.8486375.

[39] *Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and Countermeasures*. Wiley, Feb. 2016. ISBN: 978-1-119-08171-5. URL: http://dx.doi.org/10.1002/9781119081715.

[40] Rob Jansen and Nicholas Hopper. "Shadow: Running Tor in a Box for Accurate and Efficient Experimentation". In: *Proceedings 2012 Network and Distributed System Security Symposium*. Internet Society, Feb. 2012. URL: https://www.ndss-symposium.org/ndss2012/ndss-2012-programme/shadow-running-tor-box-accurate-and-efficient-experimentation/.

[41] Rob Jansen, Jim Newsome, and Ryan Wails. "Co-opting Linux Processes for High-Performance Network Simulation". In: *2022 USENIX Annual Technical Conference (USENIX ATC 22)*. Carlsbad, CA: USENIX Association, July 2022, pp. 327–350. ISBN: 978-1-939133-29-52. URL: https://www.usenix.org/conference/atc22/presentation/jansen.

[42] Marie Jehanne. *Durchschnittliche Verweildauer auf Websites: Bedeutung und Optimierung*. 2023. URL: https://contentsquare.com/de-de/blog/durchschnittliche-verweildauerwebsite/ (visited on 05/21/2025).

[43] Ishan Karunanayake et al. "De-Anonymisation Attacks on Tor: A Survey". In: *IEEE Communications Surveys & Tutorials* 23.4 (2021), pp. 2324–2350. ISSN: 2373-745X. URL: http://dx.doi.org/10.1109/COMST.2021.3093615.

[44] Christiane Kuhn et al. "On Privacy Notions in Anonymous Communication". In: *Proceedings on Privacy Enhancing Technologies* 2019.2 (Apr. 2019), pp. 105–125. ISSN: 2299-0984. URL: http://dx.doi.org/10.2478/popets-2019-0022.

[45] Albert Kwon et al. "Circuit fingerprinting attacks: passive deanonymization of tor hidden services". In: *Proceedings of the 24th USENIX Conference on Security Symposium*. SEC'15. Washington, D.C.: USENIX Association, 2015, pp. 287–302. ISBN: 9781931971232. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/kwon.

[46] Stevens Le Blond et al. "Herd: A Scalable, Traffic Analysis Resistant Anonymity Network for VoIP Systems". In: *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. SIGCOMM '15. ACM, Aug. 2015, pp. 639–652. URL: http://dx.doi.org/10.1145/2785956.2787491.

[47] Stevens Le Blond et al. "One bad apple spoils the bunch: exploiting P2P applications to trace and profile Tor users". In: *Proceedings of the 4th USENIX Conference on Large-Scale Exploits and Emergent Threats*. LEET'11. Boston, MA: USENIX Association, 2011, p. 2. URL: https://www.usenix.org/conference/leet11/one-bad-apple-spoils-bunch-exploiting-p2p-applications-trace-and-profile-tor-users.

[48] Zhen Ling et al. "A new cell counter based attack against tor". In: *Proceedings of the 16th ACM conference on Computer and communications security*. CCS '09. ACM, Nov. 2009. URL: http://dx.doi.org/10.1145/1653662.1653732.

[49] Zhen Ling et al. "Protocol-level hidden server discovery". In: *2013 Proceedings IEEE INFOCOM*. IEEE, Apr. 2013. URL: http://dx.doi.org/10.1109/INFCOM.2013.6566894.

[50] Nick Mathewson. *UDP traffic over Tor*. The Tor Project GitLab. 2020. URL: https://gitlab.torproject.org/tpo/core/torspec/-/blob/main/proposals/339-udp-over-tor.md (visited on 05/24/2025).

[51] Srdjan Matic, Platon Kotzias, and Juan Caballero. "CARONTE: Detecting Location Leaks for Deanonymizing Tor Hidden Services". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. CCS'15. ACM, Oct. 2015. URL: http://dx.doi.org/10.1145/2810103.2813667.

[52] Vasilios Mavroudis et al. "On the Privacy and Security of the Ultrasound Ecosystem". In: *Proceedings on Privacy Enhancing Technologies* 2017.2 (Apr. 2017), pp. 95–112. ISSN: 2299-0984. URL: http://dx.doi.org/10.1515/popets-2017-0018.

[53] Jon McLachlan et al. "Scalable onion routing with torsk". In: *Proceedings of the 16th ACM conference on Computer and communications security*. CCS '09. ACM, Nov. 2009. URL: http://dx.doi.org/10.1145/1653662.1653733.

[54] Prateek Mittal et al. "Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting". In: *Proceedings of the 18th ACM conference on Computer and communications security*. CCS'11. ACM, Oct. 2011. URL: http://dx.doi.org/10.1145/2046707.2046732.

[55] Ulf Moeller et al. *Mixmaster Protocol Version 2*. Internet-Draft. 2003. URL: https://www.freehaven.net/anonbib/cache/mixmaster-spec.txt (visited on 05/27/2025).

[56] S.J. Murdoch and G. Danezis. "Low-Cost Traffic Analysis of Tor". In: *2005 IEEE Symposium on Security and Privacy (S&P'05)*. IEEE, pp. 183–195. URL: http://dx.doi.org/10.1109/SP.2005.12.

[57] Steven J. Murdoch. "Hot or not: Revealing hidden services by their clock skew". In: *Proceedings of the 13th ACM conference on Computer and communications security*. CCS06. ACM, Oct. 2006. URL: http://dx.doi.org/10.1145/1180405.1180410.

[58] Steven J. Murdoch and Piotr Zieliński. "Sampled Traffic Analysis by Internet-Exchange-Level Adversaries". In: *Privacy Enhancing Technologies*. Springer Berlin Heidelberg, pp. 167–183. ISBN: 9783540755500. URL: http://dx.doi.org/10.1007/978-3-540-75551-7_11.

[59] Milad Nasr, Alireza Bahramali, and Amir Houmansadr. "DeepCorr: Strong Flow Correlation Attacks on Tor Using Deep Learning". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS '18. ACM, Oct. 2018, pp. 1962–1976. URL: http://dx.doi.org/10.1145/3243734.3243824.

[60] Milad Nasr, Amir Houmansadr, and Arya Mazumdar. "Compressive Traffic Analysis: A New Paradigm for Scalable Traffic Analysis". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS '17. ACM, Oct. 2017, pp. 2053–2069. URL: http://dx.doi.org/10.1145/3133956.3134074.

[61] L. Overlier and P. Syverson. "Locating hidden servers". In: *2006 IEEE Symposium on Security and Privacy (S&P'06)*. IEEE, 2006, 15 pp. –114. URL: http://dx.doi.org/10.1109/SP.2006.24.

[62] Francesco Palmieri. "A Distributed Flow Correlation Attack to Anonymizing Overlay Networks Based on Wavelet Multi-resolution Analysis". In: *IEEE Transactions on Dependable and Secure Computing* (2019), pp. 1–1. ISSN: 2160-9209. URL: http://dx.doi.org/10.1109/TDSC.2019.2947666.

[63] Andriy Panchenko, Stefan Richter, and Arne Rache. "NISAN: network information service for anonymization networks". In: *Proceedings of the 16th ACM conference on Computer and communications security*. CCS '09. ACM, Nov. 2009, pp. 141–150. URL: http://dx.doi.org/10.1145/1653662.1653681.

[64] Andriy Panchenko et al. "Analysis of Fingerprinting Techniques for Tor Hidden Services". In: *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society*. CCS '17. ACM, Oct. 2017, pp. 165–175. URL: http://dx.doi.org/10.1145/3139550.3139564.

[65] Andriy Panchenko et al. "Website Fingerprinting at Internet Scale". In: *Proceedings 2016 Network and Distributed System Security Symposium*. NDSS 2016. Internet Society, 2016. URL: http://dx.doi.org/10.14722/ndss.2016.23477.

[66] Andriy Panchenko et al. "Website fingerprinting in onion routing based anonymization networks". In: *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*. CCS'11. ACM, Oct. 2011. URL: http://dx.doi.org/10.1145/2046556.2046570.

[67] Mike Perry. *Congestion Control Arrives in Tor 0.4.7-stable!* Tor Blog. 2022. URL: https://blog.torproject.org/congestion-contrl-047/ (visited on 05/27/2025).

[68] Mike Perry. *The move to two guard nodes*. TOR Specification. 2018. URL: https://github.com/torproject/torspec/blob/main/proposals/291-two-guard-nodes.txt (visited on 05/07/2025).

[69] Ania M. Piotrowska et al. "The loopix anonymity system". In: *Proceedings of the 26th USENIX Conference on Security Symposium*. SEC'17. Vancouver, BC, Canada: USENIX Association, 2017, pp. 1199–1216. ISBN: 9781931971409. URL: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/piotrowska.

[70] The Invisible Internet Project. *I2P: The Invisible Internet Project*. 2025. URL: https://geti2p.net (visited on 01/08/2025).

[71] Tobias Pulls and Rasmus Dahlberg. "Website Fingerprinting with Website Oracles". In: *Proceedings on Privacy Enhancing Technologies* 2020.1 (Jan. 2020), pp. 235–255. ISSN: 2299-0984. URL: http://dx.doi.org/10.2478/popets-2020-0013.

[72] M.G. Reed, P.F. Syverson, and D.M. Goldschlag. "Anonymous connections and onion routing". In: *IEEE Journal on Selected Areas in Communications* 16.4 (May 1998), pp. 482–494. ISSN: 0733-8716. URL: http://dx.doi.org/10.1109/49.668972.

[73] *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*. Official Journal of the European Union. 2016. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679 (visited on 12/11/2024).

[74] Vera Rimmer et al. "Automated Website Fingerprinting through Deep Learning". In: *Proceedings 2018 Network and Distributed System Security Symposium*. NDSS 2018. Internet Society, 2018. URL: http://dx.doi.org/10.14722/ndss.2018.23105.

[75] Florentin Rochet and Olivier Pereira. "Dropping on the Edge: Flexibility and Traffic Confirmation in Onion Routing Protocols". In: *Proceedings on Privacy Enhancing Technologies* 2018.2 (Feb. 2018), pp. 27–46. ISSN: 2299-0984. URL: http://dx.doi.org/10.1515/popets-2018-0011.

[76] Fatemeh Shirazi et al. "A Survey on Routing in Anonymous Communication Protocols". In: *ACM Computing Surveys* 51.3 (June 2018), pp. 1–39. ISSN: 1557-7341. URL: http://dx.doi.org/10.1145/3182658.

[77] Emin Gün Sirer et al. "Eluding carnivores: file sharing with strong anonymity". In: *Proceedings of the 11th workshop on ACM SIGOPS European workshop*. EW04. ACM, Sept. 2004, p. 19. URL: http://dx.doi.org/10.1145/1133572.1133611.

[78] Payap Sirinam et al. "Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS '18. ACM, Oct. 2018, pp. 1928–1943. URL: http://dx.doi.org/10.1145/3243734.3243768.

[79] Neil Spring, Ratul Mahajan, and David Wetherall. "Measuring ISP topologies with rocketfuel". In: *ACM SIGCOMM Computer Communication Review* 32.4 (Aug. 2002), pp. 133–145. ISSN: 0146-4833. URL: http://dx.doi.org/10.1145/964725.633039.

[80] Muhammad Aliyu Sulaiman and Sami Zhioua. "Attacking Tor through Unpopular Ports". In: *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*. IEEE, July 2013, pp. 33–38. URL: http://dx.doi.org/10.1109/ICDCSW.2013.29.

[81] Yixin Sun et al. "RAPTOR: routing attacks on privacy in tor". In: *Proceedings of the 24th USENIX Conference on Security Symposium*. SEC'15. Washington, D.C.: USENIX Association, 2015, pp. 271–286. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/sun.

[82] Paul Syverson. "Why I'm Not an Entropist: (Transcript of Discussion)". In: *Security Protocols XVII*. Springer Berlin Heidelberg, 2013, pp. 231–239. ISBN: 9783642362132. URL: http://dx.doi.org/10.1007/978-3-642-36213-2_26.

[83] Qiyan Wang and Nikita Borisov. "Octopus: A Secure and Anonymous DHT Lookup". In: *2012 IEEE 32nd International Conference on Distributed Computing Systems*. IEEE, June 2012, pp. 325–334. URL: http://dx.doi.org/10.1109/ICDCS.2012.78.

[84] Tao Wang and Ian Goldberg. "Improved website fingerprinting on Tor". In: *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*. CCS'13. ACM, Nov. 2013. URL: http://dx.doi.org/10.1145/2517840.2517851.

[85] Tao Wang et al. "Effective attacks and provable defenses for website fingerprinting". In: *Proceedings of the 23rd USENIX Conference on Security Symposium*. SEC'14. San Diego, CA: USENIX Association, 2014, pp. 143–157. ISBN: 9781931971157. URL: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/wang_tao.

[86] Xiaogang Wang et al. "A potential HTTP-based application-level attack against Tor". In: *Future Generation Computer Systems* 27.1 (Jan. 2011), pp. 67–77. ISSN: 0167-739X. URL: http://dx.doi.org/10.1016/j.future.2010.04.007.

[87] David Isaac Wolinsky et al. "Dissent in numbers: making strong anonymity scale". In: OSDI'12. Hollywood, CA, USA: USENIX Association, Oct. 2012, pp. 179–192. ISBN: 9781931971966. URL: https://www.usenix.org/conference/osdi12/technical-sessions/presentation/wolinsky.

[88] Matthew K. Wright et al. "The predecessor attack: An analysis of a threat to anonymous communications systems". In: *ACM Transactions on Information and System Security* 7.4 (Nov. 2004), pp. 489–522. ISSN: 1557-7406. URL: http://dx.doi.org/10.1145/1042031.1042032.

[89] Ming Yang et al. "An active de-anonymizing attack against tor web traffic". In: *Tsinghua Science and Technology* 22.6 (Dec. 2017), pp. 702–713. ISSN: 1007-0214. URL: http://dx.doi.org/10.23919/TST.2017.8195352.

[90] Qing Yang et al. "USB side-channel attack on Tor". In: *Computer Networks* 141 (Aug. 2018), pp. 57–66. ISSN: 1389-1286. URL: http://dx.doi.org/10.1016/j.comnet.2018.05.018.

[91] Sebastian Zander and Steven J. Murdoch. "An improved clock-skew measurement technique for revealing hidden services". In: *Proceedings of the 17th Conference on Security Symposium*. SS'08. San Jose, CA: USENIX Association, 2008, pp. 211–225. URL: https://www.usenix.org/conference/17th-usenix-security-symposium/improved-clock-skew-measurement-technique-revealing-hidden.

[92] Zhongliu Zhuo et al. "Website Fingerprinting Attack on Anonymity Networks Based on Profile Hidden Markov Model". In: *IEEE Transactions on Information Forensics and Security* 13.5 (May 2018), pp. 1081–1095. ISSN: 1556-6021. URL: http://dx.doi.org/10.1109/TIFS.2017.2762825.