Defending End-to-End Confirmation Attacks against the Tor Network

Konstantin Müller

Master's Thesis
Master of Science in Information Security
30 ECTS
Department of Computer Science and MTDMT Technology
Gjøvik University College, 2015

Avdeling for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Computer Science
and MTDMT Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Abstract

Tor is an anonymity network designed for interactive applications such as Web browsing or instant messaging. The network consists of voluntarily operated nodes distributed around the world and routes user traffic over three randomly chosen nodes in order to conceal which destinations a user is accessing. If an attacker can control or observe the nodes where traffic is entering and leaving the network, one can correlate traffic and confirm that a user connected to a particular destination, for instance a Web site. Currently, Tor does not directly defend against such end-to-end confirmation attacks, because proposed defences put to much load onto the network. Instead Tor makes it harder for an attacker to come into the position to execute such an attack.

Successful end-to-end confirmation attacks were demonstrated in the past and researchers assume that such attacks are generally possible against the Tor network. However, it is not known how effective they actually can be against the current size Tor network. What is the real threat today for users by end-to-end confirmation attacks? In addition, current research shows that Tor's approach to defend such attacks has its own limitations. For this reason it is necessary to not rely on one protection mechanism alone in order to keep Tor users safe and provide the anonymity they expect from Tor.

This thesis investigates end-to-end confirmation attacks against the current size Tor network with the goal to better understand the threat such attacks pose to users. It confirms by experiments on the live Tor network that end-to-end confirmation attacks are still a valid and serious threat against Tor. This builds the necessary foundation to better protect against them. In a second step the thesis develops a defence against end-to-end confirmation attacks based on dummy traffic and examines the level of protection this can provide to users. This also studies the costs associated with the defence in order to better understand if it is worthwhile to deploy the defence to the Tor network. Experiments on the live Tor network and large-scale simulations of Tor show that the proposed defence can protect against end-to-end confirmation attacks. At the same time Tor is not slowed down by the defence from a user's perspective. Instead the defence requires higher bandwidth from Tor nodes.

The ultimate goal of the thesis is to better protect Tor users against end-to-end confirmation attacks. To the author's best knowledge this work presents for the first time a general defence directly defending end-to-end confirmation attacks against the Tor network. The proposed defence can protect against such attacks, is usable, easy to implement and easy to deploy. Utilising this defence improves the security and anonymity of Tor users, but at the same time it does not impose unacceptable high costs on the Tor network.

Keywords: Anonymity, Privacy, Tor, Onion routing, Link padding, Traffic correlation, End-to-end confirmation attack

Preface

Completing this thesis would have been impossible without the guidance from my supervisor Lasse Øverlier. Thank you for constantly giving me new ideas and feedback on my work, but also for letting me follow my own ideas and directions! Without you this thesis would not have the quality it has now! I also want to thank Erik Hjelmås for borrowing me two of his servers for several months, which I used for my experiments. Thanks to Christoffer Hallstensen for helping me with the administration of my Tor nodes and especially for taking care of the paperwork my nodes generated.

Thanks to everyone in the Tor community who cares about Tor, either by developing Tor, conducting research on Tor, spreading the word about Tor or by contributing to Tor in any other way. You are too many to name, but finishing this thesis would have been impossible without you! Your work made it much easier for me to carry out my own research. Most importantly I want to thank all the anonymous people around the world who use Tor to make the world a better place! Every day you gave me the motivation to continue working on this thesis. I hope I could contribute with my work to keep you safe in the future and that you will be able to continue using Tor for good!

Contents

List of Figures

List of Tables

1 Introduction

Anonymity is fundamental for democracy, for the people's right to exercise freedom of expression and the possibility to uncover wrongdoing unharmed. As [1] writes: *"Anonymity is necessary for the conduct of democratic politics. Not only must we be able to choose with whom we discuss politics, we must also be able to protect ourselves against retaliation for our expressions of political ideas."* Because of that, it is especially necessary to preserve anonymity while communicating over the Internet as more and more communication is today carried out digital. In information security anonymity is the desirable property to conceal the communication partners, i.e. to hide who is communicating with whom.

One very popular tool for achieving anonymity on the Internet is Tor [2], which is a low-latency anonymity network designed for interactive applications such as Web browsing or instant messaging. Tor consists of a network of voluntarily operated relays, which transport traffic from users through the network to their final destinations. Users install the Tor client software on their computers and configure their applications, for example a Web browser, to send the application traffic through the Tor network. One relay connects to the final destination, for instance a Web site, and sends the user data to it. Responses are transported back through the network to the user. By routing all traffic through the Tor network it is concealed that the user accessed the Web site.

Tor randomly selects three relays inside the network for transportation of data. The client connects to an *entry node* forwarding data to a *middle node* and an *exit node*. The exit node establishes the connection with the user's destination. The client encrypts the data three times and every relay removes one layer of encryption. Thus, the entry node knows the source of the communication, i.e. the user, but not the destination or the communication content. In contrast, the exit node can see the content and knows the destination but not the source. With this technique users gain anonymity, because no single node alone can link users to their destinations.

However, if an attacker is able to control or observe both the entry node and the exit node of a communication at the same time, one can correlate the traffic from the user with the traffic to the destination. By using statistics on the traffic such as timing or volume information an attacker can link the user with the destination. This is called an *end-to-end confirmation attack*. In general, there are two possible directions to defend against such an attack. Make it harder for an attacker to correlate traffic in order to find a match between traffic at the entry node and traffic at the exit node or prevent that an attacker can come into the position to carry out such a confirmation attack. If an attacker cannot control or observe the entry and exit nodes at the same time, it is impossible to link traffic. Currently, Tor focuses on the second method, whereas this thesis investigates the first.

1.1 Problem Description

In contrast to confidentiality, which hides the communication content to any unauthorised person, anonymity cannot be achieved by the communication partners alone. If two persons want to communicate confidential, they use encryption, but in order to be

anonymous the two need cover traffic to hide their own traffic in. Thus, an anonymity system needs as much users as possible in order to provide anonymity [3, 4]. Because Tor supports interactive applications users do not accept large delays and a poor performance. But proposed defences, for instance as described in [5–7], against end-to-end confirmation attacks relying on dummy traffic or delaying of messages in order to destroy traffic patterns are expensive and for this reason are currently not implemented in Tor [2]. Instead Tor uses an *entry guard* design as suggested in [8], where clients always select their entry node from a small set of the same entry nodes in order to make it harder for an attacker to come into the position to observe both ends of the communication.

But this does not solve the problem of end-to-end confirmation attacks itself. It only slows down attacks and makes them more expensive, because the entry guards are rotated every few months in order to better balance the load on the network [9, 10]. Eventually a connection through the Tor network is picked, which can be compromised by an attacker, who is able observe a larger fraction of the network [11]. Furthermore, most research papers, for example [11], assume that attackers have won, when they control the entry and exit nodes. But is that actually true? As of the time of writing the Tor network transports roughly 7000 MB/s of data and has about 2.5 million clients connecting to it everyday[1]. The performance of the Tor network improved a lot over the last years and all the users provide much more cover traffic as well. Are under those conditions end-to-end confirmation attacks still possible? Can they be made more difficult by deploying defences? What are the costs associated with these defences? What is the trade-off between the gained anonymity and the costs? Answering these questions is the goal of this thesis.

The focus is placed on passive attackers, i.e. attackers who only observe data in order to link traffic. Attackers may run, compromise or observe Tor nodes, they may eavesdrop on the connections between a user and the entry node as well as between the exit node and the destination or collect all traffic on a connection, for instance as an Internet service provider (ISP). Active attackers, which are able to manipulate data with the goal to enhance the attack by deleting, modifying, adding or replaying of data, are not directly studied, because passive attacks are generally easier to carry out. An attacker does not need to manipulate traffic and anyone who can intercept traffic entering and leaving the Tor network can conduct end-to-end confirmation attacks. Active attacks could also cause protocol errors, which could lead to the detection of the attack.

1.2 Justification and Motivation

End-to-end confirmation attacks are a serious problem. They *"deserve more investigation"* as pointed out by a recent survey of Tor research [12]. The paper further states that a *"solution for this problem will provide a huge improvement for the anonymity of Tor and may help thwart serious threats [...]"*. Indeed, end-to-end confirmation attacks can be utilised to deanonymise people using Tor. This can have grave consequences for them, especially in countries where Tor is used by dissidents, human rights activist or ordinary citizen to avoid repression from the government, for example in Iran or China, where Tor also helps to circumvent censorship. In addition, current research [11] shows that the entry guard design has limitations against ISP-level or state-level adversaries. Relying only on entry guards to avoid compromised connections may not be enough.

[1] All statistics about the Tor network can be found at `https://metrics.torproject.org/`.

Furthermore, the Tor designers decided in 2004 not to directly defend against end-to-end confirmation attacks because of the high costs associated with it [2]. But it is unknown if this decision still holds a decade later. Additionally, most end-to-end confirmation attacks were validated either through simulations or experiments with only small laboratory networks, for example [13–15]. It is not known if such attacks are in the current size Tor network as effective as in laboratory environments or simulations.

Better understanding end-to-end confirmation attacks and the threat they pose to the Tor network helps to better protect against them. Better defending against them helps to provide users with better anonymity and protection. In some cases this could be the difference between living in freedom and communicating without surveillance and being put in jail or worse, because a government could identify people who posted undesired information on the Internet.

1.3 Research Questions

This thesis answers the following four research questions:

Research Question 1 Can end-to-end confirmation attacks be successfully applied against the current size Tor network? How good is such an attack, i.e. what is the false-positive, what the false-negative rate of it?

Research Question 2 How can such attacks be defended? How effective is the defence? Can it actually defend the attack? In opposite to research question 1 the false-positive and false-negative rates should be high, such that an attacker is unable to correctly link traffic.

Research Question 3 What are the costs associated with the defence? What is the decrease of performance by the usage of the defence, for a single user and for the Tor network as a whole? What additional load would be placed onto the network in terms of bandwidth usage?

Research Question 4 What is the trade-off between the gained anonymity and the costs? Is the defence worth the effort or does it just slow down the network and drives users away from using Tor?

1.4 Contributions

Previous research about end-to-end confirmation attacks, for instance in [13–15], shows that such attacks can be very effective in deanonymising users and that no usable defence against them is known yet [2]. This thesis helps to understand the realistic threat of such attacks against the current Tor network and how they can be applied against Tor. In fact, this work confirms by performing a controlled experiment on the live Tor network that end-to-end confirmation attacks are still a valid threat against the current size Tor network and that even a relatively simple attack can be carried out with high accuracy. However, this work does not present a previously unknown attack against Tor. *Tor is as secure as it was before!*

These results build the necessary foundation to better defend against such attacks. In order to accomplish better protection this thesis develops and evaluates a defence against end-to-end confirmation attacks based on dummy traffic. Verifying the effectiveness of the defence by repeating the attack experiment shows that the defence can protect

against the attack. In addition, evaluating the costs associated with the defence through experiments and large-scale simulations reveals that the implemented defence does not slow down Tor from a user's perspective. The costs are higher load on the network with higher bandwidth requirements for Tor nodes. In conclusion, the main contribution of this thesis is a defence which can protect against end-to-end confirmation attacks, is usable from a user's perspective and can easily be deployed to the Tor network.

1.5 Outline

The thesis is organised in the following way. First, in the next chapter the reader is introduced to the necessary background about anonymity systems, in particular Tor, and end-to-end confirmation attacks. The goal is to provide the reader with the knowledge to follow the remainder of the thesis. In addition, related work in the area of end-to-end confirmation attacks is reviewed to form the foundation for the following chapters. Chapter 3 describes the methods employed to answer the research questions and how the research including the experiments is designed. In Chapter 4 an end-to-end confirmation attack is implemented and is effectiveness against the current size Tor network tested. Based on this, Chapter 5 develops a defence against this attack and studies how good the defence can prevent the attack. Chapter 6 determines the costs associated with the defence for both a single user and the Tor network as a whole. The results of the experiments are analysed and discussed in Chapter 7 in order to finally answer the research questions. This includes further questions not yet answered by this thesis. The thesis is concluded with Chapter 8 summarising important findings.

2 Background

This chapter contains an introduction to the concept of anonymity and anonymity research as well as an introduction to Tor. This consists of an excerpt from a specialisation course report written by the same author as part of his degree [16]. The text is taken from the report, shortened and adapted to the purpose of this thesis. The goal is to provide the reader with the necessary information to understand the remainder of the thesis. For the full details about how Tor works the reader is referred to the original publication [16]. In addition to this, previous research in the context of end-to-end confirmation attacks is reviewed in order to identify work related to this thesis.

2.1 Introduction to Anonymity

The Internet was not designed and built with security in mind. Instead secure communication protocols were developed on top of this insecure foundation. Traditionally, the three considered aspects of information security are *confidentiality*, *integrity* and *availability*. Confidentiality prevents the "unauthorized disclosure", integrity the "unauthorized modification" and availability the "unauthorized withholding" of information [17, p. 34]. Different application areas add further aspects to this basic CIA-triplet, with anonymity being one of them. This thesis uses the definition of anonymity as given by Dieter Gollmann:

> **Anonymity** – A subject (user) is *anonymous* if it cannot be identified within a given *anonymity set* of subjects. [17, p. 35]

Confidentiality can be achieved with the encryption of information, such that only the users in possession of the correct key can decrypt and read the encrypted information. This hides the information itself, but does not hide who is communicating with whom. The latter is the goal of anonymity. People communicate with each other without revealing the existence of this communication and the communication partners. But there exists a fundamental difference between confidentiality and anonymity. Confidentiality can be achieved by the communication partners alone, anonymity not. For reaching anonymity users must blend in with a group of other users, the anonymity set from the definition above. Different users communicate over the same anonymity network and provide *cover traffic* for the other users. If the users in the anonymity set and their communications are undistinguishable, an attacker cannot figure out, who is communicating with whom, thus, the users are anonymous inside the given anonymity set.

Anonymity research dates back to Chaum's design of *mixes* [18]. A mix is a computer which receives messages – originally e-mails –, delays, reorders and outputs them in batches of messages. The idea is that a single mix hides the correspondence between incoming and outgoing messages. Mixes can be combined to *cascades*, where mixes send batches of messages from one mix to another in order to further conceal who is the originator of a particular message. In Chaum's design messages are encrypted by layers of public-key encryption and each mix removes one layer of encryption. This basic design of cascades of mixes is fundamental to every anonymity system.

Anonymity systems are generally divided into *high-latency* and *low-latency* anonymity systems [4]. High-latency systems assume a *global adversary* which is able observe the whole network, i.e. every message entering, transmitted through and leaving the network. In order to defend against this strong attacker such systems introduce large random delays before messages are forwarded and for this reason are only suitable for applications tolerating a high latency such as e-mail. Such a system offers higher security, but is inappropriate for interactive applications, because users expect a quick response from these applications. In contrast, low-latency systems like Tor have a weaker threat model and generally do not try to defend against a global adversary and do not allow large delays with the goal to support interactive applications such as Web browsing or instant messaging [2]. This can in fact lead to a higher degree of anonymity compared to high-latency systems as shown in the following subsection. Today Tor is the most widely-used implementation of a low-latency anonymity system.

2.1.1 Anonymity Loves Company

The previous subsection explains that an anonymity system needs users, who provide cover traffic to other users. In general, a larger anonymity set can provide better anonymity, because it will be harder for an attacker to link network activities to specific users. Thus, an anonymity system needs to attract as much users as possible. But every user has different requirements with regard to both security and anonymity, but also in the way one uses an anonymity system. Therefore, [3] distinguishes between *low-sensitivity* users and *high-sensitivity* users. High-sensitivity users have higher requirements of security and anonymity as low-sensitivity users. They may choose to use a system A, which is highly secure but has a lower performance and is only useful for a handful of applications. But this system has a small anonymity set consisting only of high-sensitivity users, thus, it cannot provide very much anonymity. In contrast, an anonymity system B with a lot of low-sensitivity users can offer stronger anonymity and resistance against attacks even if its design is theoretically not as secure as the design of system A.

The above highlights that sometimes it may be better for anonymity to have more users and a weaker anonymity system than a stronger system and only a few users [4]. Additionally, it shows that usability and performance of an anonymity system are equally important than security and anonymity properties. An anonymity systems needs to attract low-sensitivity users in order to provide cover traffic for high-sensitivity users [3]. If these users turn away because of bad usability, they will not have anonymity at all. At the same time high-sensitivity users are put at risk of deanonymisation, because they lost their cover traffic to hide in. When designing an anonymity system this insight must always be kept in mind. A secure system alone is not enough, usability is a key factor to attract enough users [4]. The difficulty is to build an anonymity system, which is secure on one hand but also usable on the other hand.

But the size of the anonymity set alone is not enough. For example, if a company uses its own anonymity system in order to hide its communications from its competitors, this system does not provide any anonymity at all. Every communication will be by definition originate from this company. Because of that, user diversity is important as well. An anonymity system needs to attract users with different backgrounds, objectives, motivations and reasons for using the system in order to make it harder for an attacker to deanonymise users based on these properties. In this context the authors of the Crowds

anonymity network [19] coined the fitting phrase “anonymity loves company”.

But another obstacle needs to be solved in order to use an anonymity system in a secure way. The behaviour of users must be indistinguishable, i.e. users must act in the same way. For instance, a user who behaves completely different than the other users stands out from them and is therefore easily recognisable. The same also holds for the client software utilised to access the anonymity network. If the client software behaves differently compared to other users' software, this can be used by an attacker to identify the user. The reason could be misconfiguration or unusual settings. Users changing the settings with the goal to gain more security can actually hurt their anonymity, because the software behaves different compared to the users without changed settings. Thus, different users may make other security-anonymity trade-offs based on their personal requirements. This demonstrates again that anonymity set size and user diversity is important, because it is more difficult with a larger set and more diversity to find users who stand out. On the other hand it may be possible to partition users in various groups based on their behaviour or client configuration, which makes attacks to identify users easier, because an attacker can concentrate on only one smaller group of users.

In summary, this subsection shows that designing an anonymity system is a very difficult task. Trade-offs between anonymity, security and usability must carefully be examined and balanced. Focusing to close on only one property could have a negative impact on the other properties. It is important to understand this set of problems, because the design of Tor is motivated by them and it helps to comprehend it. It also helps to lead the development of improvements to Tor.

2.2 Introduction to Tor

Over the years different designs for anonymity systems were proposed. One of them is *onion routing* and Tor is the “second-generation onion router” [2], which improves the original onion routing design. Messages are wrapped into multiple layers of encryption and sent through the network until they reach the final destination. The Tor network consists of relays or nodes run by volunteers around the world, which in the Tor terminology are called *onion router* (OR). Each user runs a client software, named *onion proxy* (OP), in order to connect to the network. The OP exposes a SOCKS proxy [20], which provides a common interface to applications in order to utilise Tor for anonymous communications. With this approach Tor is able to transport arbitrary TCP streams without having to deal with application- or protocol-specific properties. Additionally, users do not need to modify their applications, for instance a browser can easily be configured to use Tor as a proxy server.

2.2.1 Bird's Eye View

Tor's design is based on a distributed trust model, see Figure 1 on the next page for a graphical overview of the basic mode of operation of the Tor network. A user does not need to trust a single entity in the network. Instead three different nodes are chosen to build a path through the network: an *entry node*, a *middle node* and an *exit node*. This path is called a *circuit*. The circuit is constructed one hop at a time. First Tor connects to the entry node, then extends the circuit to the middle node and further to the exit node. The Tor client establishes with each node secret session keys during this procedure. The client encrypts the data, which is forwarded through the network via the constructed circuit,

with each session key. Thereby the data is encrypted three times and every node on the circuit removes one layer of encryption, such that the exit node receives the plaintext. This node connects to the final destination and sends the plaintext to it, for instance connecting to a Web site and requesting a specific page. The response is transmitted back through the circuit with the nodes in reverse order. Every node encrypts the data with its own session key, such that only clients can decrypt responses, because only they know all three session keys. This procedure of adding and removing different layers of encryption is the reason for the "onion" analogy.

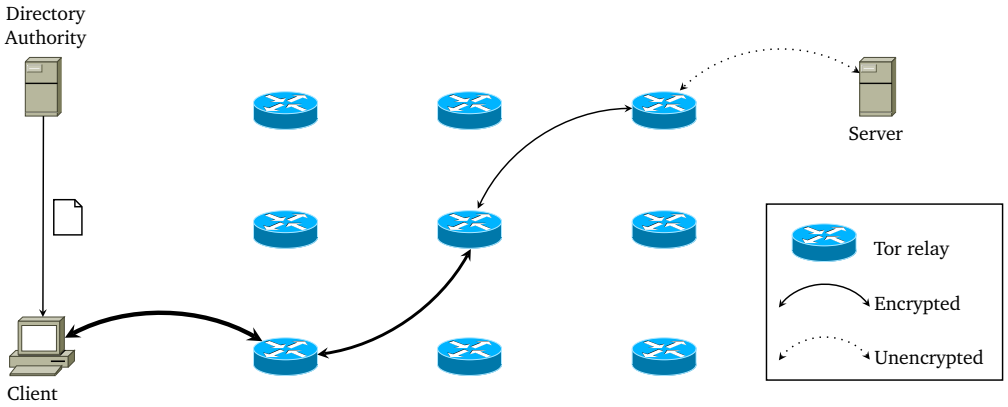


Figure 1: Overview of the Tor Network

The first node on the circuit knows the user who is accessing Tor, but does not know the content of the data sent over the network. The exit node knows the destination of the communication and can see the data in plaintext, if no end-to-end encryption protocol like TLS for encrypted Web traffic is used. But the exit node does not know, who sent the messages. With this approach no single node alone on the circuit can link together the user with the destination of the communication. Every node only knows its predecessor and successor on the circuit. Because of that, only if an attacker controls both the entry and the exit nodes, one can deanonymise users by performing an end-to-end confirmation attack. New circuits are built in regular intervals of at least ten minutes and old circuits are destroyed in order to avoid that attackers can make long lasting profiles of users by either observing or compromising the exit node.

But how do clients know the nodes inside the Tor network? Every node publishes in regular intervals a signed *relay descriptor* to a set of semi-trusted *directory authorities*, see [21]. This descriptor contains all necessary information needed for establishing a connection with a node (IP address, port, public keys, etc.). Thus, every directory authority knows all or most nodes inside the network. They conduct a vote with the information and agree on a *consensus* document, which contains the current view of the network of all directory authorities, i.e. the nodes currently participating in the network. The final network status document together with the relay descriptors is downloaded by clients from the authorities and is used in constructing circuits.

2.2.2 Selecting Nodes and Transporting Data

Tor clients choose nodes for circuits based on the flags assigned to them by the directory authorities during a vote, the nodes' bandwidth and some additional constraints, see [22]

for details. Thereby for a relay having a high bandwidth it is more likely to be picked as a node for a circuit. The goals of the selection algorithm are circuit performance, load balancing and security. Clients should use circuits, which offer them a high performance, but at the same time the load needs to be distributed over all nodes in the network in order to avoid that a few fast nodes are overloaded and must handle most of the traffic. In addition, it should be difficult for an attacker to come into the position to control both the entry and exit nodes of a circuit. From a security perspective the following rules are most important during the selection of nodes:

- No relay is picked twice.
- Not more than one relay from the same *family* is picked. Relay operators are encouraged to specify during relay configuration, that relays controlled by the same operator belong to the same family. Thus, every relay on a circuit should be controlled by a different operator.
- Only pick one relay from the same /16 IP subnet. This ensures that relays are not closely located and makes it less likely that two relays can be observed by an attacker at the same time.
- The entry node must have the *guard* flag as explained in the following subsection.

After the Tor client selected three nodes a circuit through these nodes is built. Thereby direct link connections between two adjacent relays respectively a client and a relay is secured with TLS [2]. The client establishes a TLS connection with the entry node and every node on the circuit does the same with its successor. These TLS connections are used to transport the Tor protocol, for instance creating circuits and transferring data over those circuits. Multiple circuits are multiplexed over a single TLS connection. The utilisation of TLS prevents that an outside attacker can modify the data sent over the connection or can look into the connection.

Subsequently, these TLS connections are used to transport the Tor protocol for all further communications. For this purpose Tor utilises fixed-size *cells* with two basic types of cells: *control cells* and *relay cells* [23]. Control cells are exchanged between two adjacent nodes and are always interpreted by the node receiving them, for example the creation of circuits uses control cells. In contrast, relay cells are end-to-end communication between the client and an arbitrary node on a circuit and are utilised for end-to-end stream management and end-to-end transportation of data. Relay cells are encrypted with the session keys such that intermediary nodes on a circuit cannot see the content of relay cells and just forward them to the subsequent node on the circuit. Because of the layered encryption only the rightful receiver can finally decrypt the cell.

2.2.3 Guard Nodes

In the context of end-to-end confirmation attacks the concept of *guard nodes* is important for the security of Tor. Recall that with such an attack the goal of an attacker is to control both the entry and the exit nodes on a circuit in order to correlate traffic. In general, if an attacker controls C out of N nodes and nodes are selected uniformly at random, the probability of choosing a compromised circuit with the attacker controlling both entry and exit nodes is $\left(\frac{C}{N}\right)^2$. Over time this probability goes to 1, when a client keeps building circuits at random. The attacker only needs to wait until the client creates a compromised circuit. The concept of guard nodes tries to mitigate this attack [22].

The directory authorities assign the *guard* flag to nodes which should be used by

clients as guard nodes. Then upon the first start-up each Tor client chooses a small set of guard nodes, currently three, and always uses one of the guard nodes as the entry node for a circuit. When selecting guard nodes, there are two possible outcomes. First, the guard is controlled by the attacker and all circuits can be compromised, or second, the guard is not controlled by the attacker and the attacker never has a chance to compromise a circuit. If the number of attacker controlled guards compared to all relays in the network is small, then the probability of picking a bad node is small as well. This increases the costs for attackers, because they need to run more nodes in order to increase the likelihood of a successful attack.

But this method also has a drawback. Because clients do not change their guards and new clients choose the same guard nodes, these nodes accumulate more and more clients and traffic they need to process. For this reason clients rotate their guard nodes currently every 8 to 12 weeks [9]. This helps to distribute the traffic over all guard nodes. But this is always a trade-off between performance, load balancing and security and is subject to current research in order to choose the best parameters for this trade-off [10]. The Tor developers work towards the goal of a single guard node and a rotation period of 9 to 10 months [24, 25], because recent research shows that the current implementation of guard nodes has limits [11]. The goal is to further slow down attacks, because with these parameters attackers need to wait even longer before clients pick their controlled nodes. It is important to note that the design of guard nodes does not prevent end-to-end confirmation attacks themselves, instead it is designed to slow down attacks and to make attacks unattractive due to the required time and/or costs for running nodes.

2.3 Related Work

This section reviews the literature for previous research related to this thesis in order to identify research which can help to answer the research questions. First, generic end-to-end confirmation attacks and defences against them are described. This is followed by a subsection about Tor-specific attacks and defences. The section is finished with a short summary of the results from this literature review.

2.3.1 End-to-End Confirmation Attacks

End-to-end confirmation attacks are studied both in the context of high-latency and low-latency anonymity systems. In [26] two passive attacks against low-latency systems are presented. The main idea is to observe the incoming and outgoing connections of a node in the network. The first attack counts the number of messages on the incoming and the outgoing links of a single node. Links with a similar number of messages correspond to each other. The second attack tracks the start of a connection. When on an incoming connection a lot of traffic is detected and shortly after an increase in traffic on an outgoing connection as well, both connections belong to the same traffic flow. This work only considers the observation of one single node, but the same techniques could be used to track traffic from one node to another through the whole network by a global adversary. Furthermore, if the anonymity system is viewed as a black box only observing traffic at the edges of the network, this attack can be applied as well in order to conduct end-to-end confirmation attacks.

Another attack, the *statistical disclosure attack*, is introduced in [27]. The basic assumption is that a user communicates with a fixed number of recipients and that the

background traffic of other users is uniformly distributed over all recipients. An attacker wants to confirm the recipients of a specific user within a single mix system. To do this one needs to observe every message entering and leaving the mix. In each round messages are sent to the mix, which outputs them in fixed-size batches. The attacker remembers the recipients of messages in every round as candidates for the user's recipients. By observing many rounds the most likely candidates are revealed, because they will receive significantly more messages compared to the uniformly distributed background traffic.

In [13] the statistical disclosure attack is extended in order to relax the strong assumptions from the original attack. Especially, the attack is broadened to more complex traffic patterns, both for users and the background traffic, to variable-size batches, to networks with mix cascades and to possible dummy traffic inside the network. But most important the requirement of a global adversary is relaxed. This makes the attack suitable for an attacker only able to observe a part of the network, for example an ISP, which may then be able to correlate traffic entering and leaving the anonymity network. Furthermore, the attack can be extended to anonymity systems such as Tor not operating in rounds with mixes forwarding messages in batches. An attacker can artificially introduce rounds by fixed-size time windows, which would constitute different rounds, and using variable-size batches from the extended statistical disclosure attack.

The statistical disclosure attack studies mixes which forward messages in batches. In contrast, [28] analyses mixes forwarding each message individually, but messages may be delay by a randomly chosen amount of time. The described attack extracts the traffic pattern of an incoming link and determines with a statistical hypothesis test if this pattern corresponds to a second observed pattern on an outgoing link. By comparing the input pattern with every link pattern in the network the traffic can be traced through the network. The links with the highest similarity are the links most likely transporting the messages belonging to the input pattern. Additionally, the same technique can be used to correlate traffic at the edges of the network, which makes the attack suitable for end-to-end confirmation attacks as well.

The attacks described so far assume an attacker able to observe partly or entirely the anonymity network, but do not further define the attacker. In contrast, [14] investigates the possibility that an Internet exchange point conducts confirmation attacks by just using the traffic statistics already stored for the purpose of network management. By using a probabilistic Bayesian approach it is possible to correctly correlate traffic going through the same Internet exchange point. This method has the interesting property that it is independent of the exact timing of messages and therefore not affected by randomly introduced delays of messages.

During the above attacks the attacker only passively observes the traffic from the anonymity network under attack. The attacker does not actively interfere with the traffic. For example, an attacker could be in the position to manipulate or artificially delay packets going through the network. This could be used to enhance an end-to-end confirmation attack. In [29] the attacker introduces a recognisable pattern into traffic by delaying packets. This pattern encodes some bits of information. Subsequently, these bits can be recovered by recognising the pattern in traffic. If this pattern is modulated onto traffic entering an anonymity network and recovered from traffic leaving the network, the attack can be used as an end-to-end confirmation attack by linking incoming and outgoing traffic together.

2.3.2 Defences against End-to-End Confirmation Attacks

An own class of end-to-end confirmation attacks against low-latency anonymity networks are *timing attacks* [5, 6], which exploit timing information to correlate observed messages. For example, observed traffic is divided into adjacent intervals and each interval in turn in fixed-size windows. For each window the number of transported packets is counted. Different intervals can be statistically correlated in order to find matches between observed traffic. One way to defend against this kind of attack is constant cover traffic sent by the user with the goal to make traffic patterns indistinguishable. Another defence called *defensive dropping* is proposed in [5]. The idea is to send cover traffic which is dropped at intermediary nodes between the user and the destination. This should confuse attackers, because the traffic entering and leaving the network exhibits different patterns. Yet another approach is *adaptive padding* [6]. With adaptive padding nodes inside the network inject with some probability dummy messages between user messages in order to confuse the attacker by destroying the traffic pattern. The advantage of adaptive padding is that it does not artificially delay messages from users.

Defensive dropping and adaptive padding consider traffic flows from individual users independently. In contrast, [7] proposes an algorithm called *dependent link padding*, which takes at a single mix all incoming traffic flows from different users and generates outgoing traffic flows with the same timing characteristics. All outgoing flows transmit packets at the same points in time. If a flow does not have an incoming packet to forward, a dummy packet is sent instead. Because of that, all flows transmit packets, either real or dummy, at the same time with the same timing characteristics, such that an attacker cannot match incoming flows to outgoing flows. This algorithm can be extended to a cascade of mixes with each mix following the dependent link padding strategy. In addition, a maximum delay between receiving a packet and forwarding it can be specified in the algorithm in order to make it suitable for interactive applications. But a smaller delay increases the bandwidth requirements, because more dummy packets need to be sent compared to real packets as the time frame between two transmitting events for receiving real packets is smaller as well.

Another defence specifically designed to protect against active timing attacks, where an attacker drops or delays packets, is proposed in [30]. The basic idea is to reuse a working defence against passive timing attacks and to turn it into a defence against active timing attacks. In this proposal packets are sent over multiple paths through the anonymity network according to the utilised defence and are forwarded at each node at a specified point in time. If the attacker does not control the majority of the network, active attacks can be defended, because packets not manipulated always reach the destination in time. Thus, if an effective defence against passive timing attacks can be found, the same defence can be utilised to counter active timing attacks as well.

2.3.3 Tor-specific Attacks and Defences

The attacks and defences presented so far are not specific to Tor or any other anonymity system. They describe general attacks against anonymity systems. The results are validated through system-independent simulations and theoretical analysis. In addition, [31] implements the packet counting, the connection start tracking and the described timing attack within a custom Tor simulator in order to test the effectiveness of the attacks by a global passive adversary. The results are that the attacks can be very effective, when

only traffic with low volume is transported over a shared connection, but become less effective, if the traffic volume on the connection increases.

Furthermore, [15] develops an end-to-end timing attack against the Tor network. By analysing the traffic Tor generates during the set-up of a connection through the network an entry node and an exit node can confirm that a particular connection goes through both nodes. Thus, this attack deanonymises users even before any data is transmitted. The attack is validated by an experiment executed in a laboratory environment in order to avoid negative effects on the real Tor network.

An attack against Tor's *location-hidden services* is presented in [8]. Hidden services are a technique to provide anonymity to a service by concealing its IP address. The attack employs an extended version of the packet counting attack by incorporating timing information to reveal the real IP address of a hidden service. This attack is demonstrated on the live Tor network by attacking own controlled hidden services.

The attacks presented in [8, 15] show that passive end-to-end confirmation attacks can be successfully applied against the Tor network. In addition, [32] demonstrates an active end-to-end confirmation attack. A malicious entry node manipulates (add, delete, modify or replay) data sent through the network and a colluding exit node detects the manipulation, because it causes an error during the decryption of data at the exit node. Thus, it can be confirmed that traffic entering the network at the entry node leaves the network at the exit node, which reveals both the user and the corresponding destination. The original Tor design paper calls this a *tagging attack* [2].

In [33] a further active end-to-end confirmation against Tor is demonstrated. In this attack a malicious entry node marks traffic by inserting a specific signal into traffic. The signal is encoded by generating bursts in the forwarded traffic and can be recognised by a colluding exit node in order to link the marked traffic together. A similar active attack is carried out in [34] in order to identify hidden services. An attacker controlled node introduces a pattern in the traffic to the hidden service by generating specific dummy traffic, which can be detected by the hidden service's entry node. If the attacker also controls the entry node, one knows the real IP address of the hidden service. All three described active end-to-end confirmation attacks are demonstrated on the live Tor network and it is shown that they can be very effective.

The same basic idea as in [14] is followed in [35] as well in order to implement an active end-to-end confirmation attack against Tor by correlating network traffic statistics. When a user connects to a server under the control of the attacker, the server sends a response to the user. The response is transmitted in a way that it exhibits a distinguishable traffic pattern. Using standard traffic management software at routers between the user and the entry node as well as the exit node and the server traffic statistics about connections are gathered and correlated afterwards in order to confirm that the user accessed the malicious server. Additionally, a defence against this particular attack is proposed. The defence relies on dummy packets with a very small time-to-live (TTL) value in the IP header sent from the entry node towards the user. The dummy packets distort the traffic statistics with the goal to prevent the correlation of traffic. Because the TTL value is very small the dummy packets are dropped at intermediary routers very quickly and never reach the user. For this reason the defence does not require very much additional bandwidth on the connection between the user and the entry node.

2.3.4 Summary

The attacks described in this chapter demonstrate possible end-to-end confirmation attacks against Tor, but not all of them are directly targeted at Tor. Nevertheless, [8, 15] and [32–35] show that both passive and active end-to-end confirmation attacks against the Tor network are possible. The results indicate that such attacks can be very effective, however, it is unknown if all of them work against the current size Tor network. Defences against such attacks can be delay of traffic, cover traffic or in particular defensive dropping, adaptive padding as well as dependent link padding. But the literature only examines how effective those mechanisms can protect against attacks and does not quantitatively analyse the costs associated with them in terms of network performance or network load. But such an analysis is very important, because deploying costly defences can drive away users and can actually decrease the anonymity of the remaining users as highlighted in Subsection 2.1.1 of this chapter. In addition, it increases the costs for volunteers to run Tor nodes. Furthermore, most defences, except [35], have not been implemented and tested on the Tor network.

In conclusion, the literature as presented in this section can provide a starting point for answering the research questions as stated in the introduction, especially about end-to-end confirmation attacks. But the effectiveness against the current size Tor network as well as defences and the trade-off between anonymity and the costs associated with defences need to be further studied.

3 Methods

In order to answer the research questions three different methods could be used: simulations, laboratory experiments with a test Tor network or experiments on the live Tor network. This chapter describes first the different methods with their advantages and disadvantages and analyses to what extend they are suitable to answer the research questions. Second, the choice of methods is justified and it is explained how the thesis approaches to answer the research questions with the selected methods. For answering them several experiments are carried out as described in the following chapters and the precise set-up of those experiments is illustrated as well.

3.1 Description

The Tor Project makes it very easy and comfortable to conduct research about Tor. All data and statistics about the Tor network from the last ten years are publicly available for analysis[1]. Furthermore, the open nature of the network – everyone can operate Tor nodes – makes it possible to experiment with the live Tor network. Researchers can set up own Tor nodes, which are integrated into the live network, and can use those nodes for carrying out their experiments. This has the great advantage that the results from the experiments are more valid, because they are obtained from the live network with real traffic and real user activity. The problem is that the experiments may not be reliable or repeatable, because user activity is never the same and can change over time. Users and Tor nodes are constantly joining and leaving the network. Thus, the Tor network is always subject to changes.

However, not all experiments are possible to execute on the live network. For example, if experiments require modifications to a lot of Tor nodes, such experiments could be to costly due to the fact that many Tor nodes need to be operated by the researchers. In addition, experiments on the live network could cause negative effects on the network itself and often it is impossible to anticipate effects caused by the experiment beforehand, see for instance [34]. Due to the nature of anonymity research users must never be identified during experiments, because people rely on Tor to communicate anonymously. Therefore, experiments on the live Tor network must be carefully designed such that they do not harm users, see [36] for a negative example. Nevertheless, experiments on the live Tor network were successfully carried out [8, 32, 37].

3.1.1 Experiments with Laboratory Networks

If experiments with the live Tor network are impossible, an alternative solution are private laboratory networks. Researchers set up a private Tor network and carry out their experiments on this network. With this method the experiments do not interfere with the live network and the experiments are performed in a controlled environment, which makes them more reliable and repeatable. But private Tor networks are much smaller than the real Tor network and it is very difficult to simulate real user activity and traffic.

[1] Available at `https://collector.torproject.org/`.

Because of that, it could be possible that the results obtained with such experiments cannot be transferred to the real Tor network. Furthermore, setting up private Tor networks may be expensive and cumbersome due to the resources such as computers needed. This can be mitigated by using a shared experimental network such as PlanetLab[2], which allows researchers to perform their experiments on the PlanetLab computers distributed around the world. But a research institution needs to be a member of PlanetLab in order to be able to use it.

Experiments on private Tor networks are a good alternative if experiments with the live Tor network are impossible. Different researchers successfully used this approach, either with PlanetLab [15] or own private networks [38].

3.1.2 Simulations

A third alternative method is simulation. Instead of running real experiments in a Tor network, either the live network or a private network, a Tor network and the experiments are simulated. This approach is very reliable, because with the same parameters repeated simulations should produce the same results. Single parameters can be manipulated and their effect studied in order to uncover cause-and-effect relationships. Simulations have the advantage that they are cheap and can be executed automatically, because not very much physical hardware is needed (except the computer running the simulation). Albeit providing a controlled environment for experiments simulations have the same drawback as private research networks, that their results may not be transferable to the real Tor network. Nevertheless, simulations can be successfully used for research about Tor, for instance in [11, 31]. In fact, most end-to-end confirmation attacks were validated by simulations [5, 6, 13, 14, 26, 28], however, these simulations were independent of Tor.

In order to make experiments with Tor easier, comparable and more reliable two different tools have been developed: *ExperimenTor* and *Shadow*. ExperimenTor [39] allows to run realistic Tor experiments by providing an emulated virtual network with realistic network properties such as latency, bandwidth or drop rates of packets. ExperimenTor emulates an entire Tor network with the same characteristics as the real Tor network like the bandwidth each Tor node provides. Real applications like Tor itself or a Web browser are executed in a controlled manner and connected to the virtual network. This allows researchers to create realistic experiments, which makes it possible to study effects on the *whole* Tor network by deploying a modified Tor network. For example, a new defence against end-to-end confirmation attacks could be implemented and the effects such as performance or network load on the whole network studied, when *all* clients use the defence. ExperimenTor uses one node which emulates the virtual network and several *edge nodes*, which connect to the virtual network and run the applications.

The same goals as ExperimenTor but a completely different approach is followed by Shadow [40]. Instead of emulation Shadow uses simulation in order to provide efficient, accurate and controlled experiments. Shadow is a single program, which executes real and unmodified applications like Tor during the simulation. Applications are integrated into Shadow via plug-ins. Furthermore, it simulates a virtual network with latency and bandwidth, a realistic Tor network as well as cryptography and CPU operations. Shadow has the same goals as ExperimenTor and can be utilised for the same purposes and experiments, but has one great advantage. It is a single application, which can be executed

[2] Visit `https://www.planet-lab.org/` for more information.

on a single computer and does not require the set up of different nodes like ExperimenTor. In addition, Shadow can be easily deployed to Amazon's EC2 cloud, which makes experiments with Shadow convenient and cost-effective.

Experiments conducted with both Shadow and ExperimenTor require a realistic model of the Tor network. Only if the experiments model the network accurately, the results of the experiments can be transferred to the real Tor network. In [41] such a model is developed for both tools based on the Tor network as it existed in January 2012. This model could be the basis for Tor experiments with Shadow or ExperimenTor.

3.1.3 Summary

All three methods are suitable to answer the research questions. But in order to study the effectiveness of end-to-end confirmation attacks against the current size Tor network experiments on the live network are necessary, because only such experiments can give accurate answers regarding the real Tor network. Private research networks and simulations cannot model the current size of the Tor network accurately enough. Furthermore, end-to-end confirmation attacks can be studied by just deploying two Tor nodes, an entry and an exit node, and routing own client traffic over both nodes. In this case other users only provide cover traffic and this attack must not try to identify other users, only the self-controlled client. The effectiveness of a defence against end-to-end confirmation attacks can be studied with the same set-up as well, if the defence does not require the modification of many Tor nodes. In addition, the performance penalty of the defence for a *single* client can be measured as well.

Defences which require the cooperation of many Tor nodes need to be studied with a private network or through simulation, because the defence cannot be deployed to the whole live network. Furthermore, the performance effects of a defence against the *whole* Tor network when *all* clients use the defence can only be analysed with a private network or simulation as well.

3.2 Research Design

The research questions are answered by a combination of experiments on the live Tor network and through simulations. The following five steps are carried out:

1. Deploying an entry and an exit node to the live Tor network and implementing an end-to-end confirmation attack on those two nodes by correlating the traffic of a controlled client going through both nodes. This answers research question 1 by assessing how effective the attack can be against the real Tor network.
2. Implementing a defence against the confirmation attack and repeating step 1 in order to study the effectiveness of the defence compared to the success of the attacker without the defence. This assesses the effectiveness of the defence for a single client using the defence in isolation.
3. Studying the effectiveness of the defence in the case all clients employ the same defence by carrying out a simulation with Shadow. Together steps 2 and 3 answer research question 2.
4. Measuring the performance penalty imposed by the defence for a single client. The modified client with and without enabled defence is used in this step as well.
5. Measuring the performance penalty imposed by the defence for the whole Tor network by performing a simulation with Shadow and all clients utilising the defence.

Steps 4 and 5 together answer research question 3.

Research question 4 is answered by analysing the results obtained from all five steps.

The effectiveness of the end-to-end confirmation attack is measured in terms of false-positive and false-negative rates. A false-positive occurred when the attack decides that two traffic patterns correspond to each other, when in fact they do not. A false-negative is the opposite case when the attack could not match two corresponding traffic patterns. The equal error rate (ERR) gives the rate at which both the false-positive and the false-negative rates are the same. An attacker tries to minimise the ERR. In contrast, an effective defence should maximise the EER for the attacker.

In order to assess the costs associated with a defences both the network performance and the network load needs to be measured. Performance can be evaluated by clients. They download the same files over the Tor network and measure the time needed for the download with the defence enabled and disabled. The tool *Torperf* is utilised for this purpose [42]. Torperf downloads files via Tor and measures the time needed. Additionally, every Tor node publishes statistics about how much data it transported [42]. Aggregated statistic about the transported data can be used to measure the load on the Tor network. When a simulation is performed with Shadow, Shadow stores data about network performance and network load, which can automatically be analysed.

3.3 Experimental Set-Up

The set-up of the experiments used for validating both the implemented end-to-end confirmation attack and the developed defence against the attack is depicted in Figure 2 below. The set-up for the experiments is the same in both cases with the only difference that in one case the defence is enabled in order to study how effective the defence can protect against the attack. For a detailed description of the attack and the defence see Chapter 4 and Chapter 5, respectively.

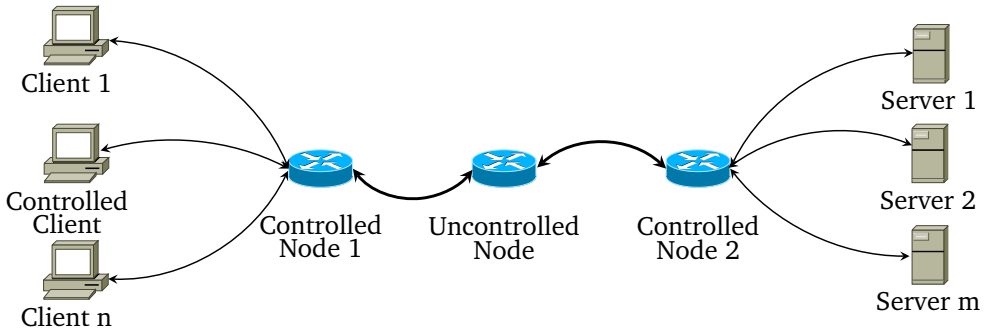


Figure 2: Set-up of the experiments on the live Tor network

Two Tor nodes are deployed to the Tor network and act as ordinary nodes in the network, i.e. other Tor clients use them to transport data through the network. A controlled client is modified to always choose one controlled node as its entry node and the other controlled node as the exit node. The middle node can be any other Tor node in the network. The client generates traffic by accessing a server and the traffic is sent over both controlled nodes. During the experiment the controlled nodes store for every connection going through them the traffic patterns exhibited by the connection. After the experi-

ments those patterns are correlated with the goal to match traffic at the entry node with traffic at the exit node. The goal of the attacker is to identify the traffic generated by the controlled client with high accuracy.

Other Tor clients generate cover traffic at the two controlled nodes. Therefore, every pattern at the entry node must be compared to every pattern at the exit node. In this process the following cases can occur:

- Traffic from the controlled client is correctly matched at both nodes.
- Traffic from the controlled client is not matched at both nodes. This is a false-negative.
- Traffic from the controlled client at one node is incorrectly matched with cover traffic at the other node. This is a false-positive.
- Cover traffic is matched at both nodes. This is also a false-positive, because the two controlled nodes are configured to belong to the same *family*, which means that clients never pick both nodes for a circuit at the same time [22][3].

This set-up allows to determine the effectiveness of both the end-to-end confirmation attack and the defence against the attack by counting and comparing the correct matches as well as the false-negatives and the false-positives in both cases.

The two controlled nodes are named *Tor1HiG* and *Tor2HiG* and all the information about the nodes, for instance usage statistics and network status, can be accessed through Tor's Globe portal[4,5]. Both nodes operate on the Tor version 0.2.5.10. All experiments and modifications made to the Tor software are based on this version as well.

3.3.1 Threat Model

It is important to understand the adversary able to conduct end-to-end confirmation attacks. This is needed to accurately estimate the threat of end-to-end confirmation attacks against users and to interpret the results of the experiments correctly. The adversary assumed in this thesis is able to observe or control a fraction of the Tor network. In the simplest case, as with the experimental set-up described above, the attacker just needs to control two Tor nodes. This is sufficient for the experimental validation of an end-to-end confirmation attack, because the controlled client can be explicitly instructed to use the attacker's nodes. But a real world attacker would need to control more nodes in order to increase the likelihood that a user chooses malicious nodes for circuits through the network. In this case the attacker has to correlate traffic between all controlled nodes. Because of that, the attack is only limited by the adversary's resources, i.e. the number of controlled nodes and analysis capabilities.

An attacker controlling nodes inside the Tor network is a valid threat, because everyone can operate Tor nodes without any central instance approving those nodes. Such nodes are very difficult to detect, because by carrying out a *passive* end-to-end confirmation attack the adversary does not tamper with traffic, instead one just observes traffic going through nodes. Therefore, a client cannot detect if traffic is routed over a malicious node. In addition, it must be noted that for such an attack it is not necessary to operate own Tor nodes. If attackers are just able to observe traffic going in and out Tor nodes, they can perform exactly the same attack. This is even possible if the attacker

[3] This configuration also ensures that no other users will be deanonymised during the experiments.
[4] Tor1HiG: `https://globe.torproject.org/#/relay/B912D67FCB5A968590C9BDCA37B7D482A994B868`
[5] Tor2HiG: `https://globe.torproject.org/#/relay/D0C1FAF148A434977C554E1E0CB9CABEA7F619F5`

only observes the connection between a user and an entry node as well as the connection between an exit node and the destination of the communication. Internet service providers, Internet exchange points as demonstrated in [14] or a state-level adversary can realisticly carry out this kind of network-level attack.

The end-to-end confirmation attack validated in the experiments answers the following question: How accurately can an attacker identify traffic going through the two controlled Tor nodes? Basically, the controlled client sends traffic over both controlled nodes and the attacker tries to identify this traffic. This attack can be used to confirm a previous suspicion, that a particular user accesses a specific destination, for example a Web site. Accessed user A Web site B? In addition, an adversary might want to find out which *set* of Web sites user A visited. If not a particular user is targeted, the attacker could try to identify the set of destinations for a set of users. The latter would be an untargeted deanonymisation of many users.

Because of the probabilistic nature of end-to-end confirmation attacks false-negatives and false-positives are always possible. A false-negative means that the attacker was not able to correctly correlate traffic, i.e. the attacker could not link users to their destinations. On the opposite a false-positive gives a false match, i.e. the attacker assumes that a user accessed a destination when in fact one did not. A high false-negative rate basically implies that attackers failed with their attack, because they could not link user traffic. A high false-positive rate implies that they cannot draw relevant conclusions from their attack, because they made to many mistakes during the attack and they cannot assert with certainty that a user accessed a specific destination.

3.3.2 User Traffic Model

In order to be able to draw relevant conclusions from the experiments and to transfer the results to the real Tor network it is important to accurately model user traffic during the experiments. The controlled client generates traffic in a similar way than the Tor client model utilised in [41]. Traffic is divided into two categories: Web surfing and bulk download. For Web surfing the client generates the following traffic. For five minutes the client repeatedly downloads 2000 KiB of data. According to the HTTP Archive[6] and its scan of 10.000 Web pages from the 01. January 2015 the average size of a Web page is 1931 kB. Between two downloads the client waits for a randomly chosen interval from 1 to 60 seconds. This simulates user access to a Web page and the time someone needs to consume the page before surfing to the next page. In [43] it is shown that 80% of all users do not stay longer than 70 seconds on a single page. For bulk download the client repeatedly downloads a 5 MiB file for five minutes without a delay between two successive downloads. This models file sharing or watching an online video. Each experiment is repeated ten times at different days and times and each attempt consists of generating five minutes of Web surfing and five minutes of bulk download traffic. During the experiments traffic patterns at both controlled nodes are recorded and analysed afterwards.

The files to download via HTTP are `http://torperf.torproject.org/.50kbfile` and `http://torperf.torproject.org/.5mbfile`. The 5 MiB file is used for the bulk download case and the 50 KiB file for the Web surfing case. In the latter case the file is downloaded 40 times in a row without a delay in order to simulate the access to a

[6] Available at `http://www.httparchive.org`.

Web page which is composed of different files (HTML, pictures, etc.). The above files are originally intended to be used by Torperf, but are reused for the experiments, because they do not change in opposite to real Web pages. For this reason the accuracy of the end-to-end confirmation attack does not depend on the downloaded file. Furthermore, the results from two different attempts are better comparable. This would not be the case, if the client would access a live Web page, which could change between two attempts.

The set-up of an experiment is always a trade-off between validity and reliability, or between accuracy and repeatability. The described experiment is repeatable and comparable between multiple runs, if the cover traffic and the network conditions do not change significantly between the runs, because these parameters are the only parameters not controllable during the experiment. This is compensated by repeating each experiment ten times. It is also accurate due to the fact that the experiment is carried out on the live Tor network simulating a realistic adversary and due to the employed traffic model.

4 End-to-End Confirmation Attack

In order to answer research question 1 an end-to-end confirmation attack against Tor is implemented and its effectiveness on the live Tor network studied. The implemented attack is a timing attack as first described in [5]. The same attack is employed in [6] as well to evaluate another defence against timing attacks. This attack was chosen because it is easy to implement in Tor with the experimental set-up described in the previous chapter. Furthermore, to the best knowledge of the author this attack was until today never directly employed against the Tor network. Previously the attack was only validated through simulations. Because of that, it is actually not known how effective this specific attack can be against Tor.

4.1 Description

Independently of Tor, the attack assumes the following adversary [5]. The attacker controls or observes two nodes inside an anonymity network. A user creates a path through the network with one of the two nodes being the first hop of the path and the other node the last hop. The attacker can determine that traffic at the first hop originates from the user and that traffic at the last hop leaves the network to the final destination. If the attacker can correctly match the traffic at the two nodes, one can link the user with the destination. Additionally, it is assumed that the attacker can distinguish between traffic flows from different users at both nodes.

Formally, user A chooses a path P_A through the network with the hops $H_1^A \dots H_n^A$ on the path, where the attacker controls both H_1 as the first node and H_n as the last node. For this attack the number of hops is irrelevant, only the first and last nodes matter. In addition, another user B creates a path P_B through the network. The attacker records traffic flows at both nodes. When traffic patterns at H_1^A and H_n^B are observed, the goal is to determine if $A = B$. Effectively, the attacker has to match all traffic at H_n with the traffic from user A at H_1. Therefore, other users produce traffic acting as cover traffic for user A, if the attacker cannot correctly correlate A's traffic.

The concrete attack works in the following way, see also the graphical illustration in Figure 3 on the next page [6]. The time of observation, i.e. the time while the attacker records traffic, is divided into adjacent time windows W. [6] suggests $W = 1s$ as producing the best results. For each window the attacker counts the number of packets received for each traffic flow. x_k and x'_k denote the number of packets during the kth window at the first node and the last node, respectively. For every possible entry and exit traffic flow combination the attacker calculates the cross-correlation r of the two flows as

$$r = \frac{\sum_k \left((x_k - \mu)\,(x'_k - \mu') \right)}{\sqrt{\sum_k (x_k - \mu)^2}\sqrt{\sum_k (x'_k - \mu')^2}}, \tag{4.1}$$

where μ and μ' are the means of the packet counts of the two traffic flows compared [6] and r being in the interval $[-1; 1]$. If r is greater than some threshold t the attacker decides that both flows carry the same traffic. Furthermore, assume the two traffic flows

A and B. If $r > t$ but $A \neq B$, this is a false-positive. If $r < t$ but $A = B$, this is a false-negative. When the attacker chooses t such that both the false-positive and the false-negative error rates are equal, this gives the equal error rate of the attack.

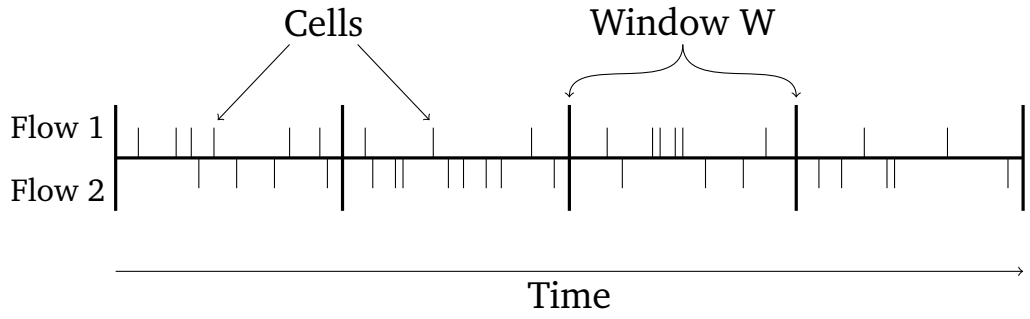


Figure 3: Graphical illustration of the implemented end-to-end confirmation attack

4.2 Implementation

The implementation of the attack consists of two parts. The first part is modifying the two Tor nodes to record traffic patterns. The second part is the development of a tool to analyse the recorded traffic patterns, to calculate the cross-correlation between patterns, to decide on traffic matches and to compute false-positive and false-negative rates.

This approach has the advantage that most of the attack logic is implemented in the analysis tool. Therefore, the attack can be tweaked without the need to repeat recording of data. Once traffic data is gathered, the actual correlation of traffic takes place in the analysis tool. On the downside of this approach data is not preprocessed, thus, producing large log files in the order of hundred megabytes. But this is not an issue, because the actual analysis of traffic is computed offline with the analysis tool and most time is spent while correlating traffic, which is a $n \times m$ comparison of n entry flows with m exit flows, and not for reading the log files.

4.2.1 Record Traffic Patterns

In order to record traffic patterns the controlled Tor nodes are modified in the way that they intercept each received cell in the forward direction, i.e. cells travelling from the client to the destination, and record the time when the cell was received, the circuit ID from the circuit the cell belongs to, the IP address from the sender of the cell and the IP address the cell is forwarded to. For each received cell these values are written to a log file. In Tor's configuration file at both controlled nodes it must be defined when to start and when to stop recording data.

Cells are only stored once a circuit is fully established between sender and receiver. This is needed in order to reliably distinguish between cells travelling in the forward and in the backward direction. But this also means that the cells needed to set up the circuit are not recorded. However, this is not a problem, because only a handful of cells are exchanged during circuit establishment [23]. For this reason these cells do not have a great influence on the cross-correlation computation, which counts the number of cells, because the number is negligible small compared to the number of cells sent for real user traffic.

The circuit ID is saved in order to be able to distinguish between traffic flows from different users. For example, the exit node can receive traffic from the same middle node, which could belong to different users. The circuit ID is used to differentiate between these flows. In the case of the entry node the sending IP address belongs to the user, whereas in the case of the exit node this IP address belongs to the previous Tor middle node. The forward IP address is the next Tor middle node for the entry node case and the final destination for the exit node case. In the latter case multiple forward IP addresses could be saved, because multiple exit connections are sent over the same circuit by the user's Tor client [2]. These exit connections are not distinguished, because they come from the same user. In the case of a traffic match all exit connections are correctly attributed to the same user.

In essence, both nodes save every cell on all circuits received in the forward direction. On a traffic match the sender IP address at the entry node gives the user and the forward IP address(es) at the exit node the destination(s).

4.2.2 Sanitisation of IP addresses

Because the attack is carried out on the live Tor network with actual users, who use Tor to preserve their privacy and anonymity online, no IP addresses are directly stored in the traffic logs. Instead IP addresses are concealed by only saving a Base64 [44] encoded HMAC of the IP addresses, i.e. using a keyed-hash function [45]. The key used for the calculation of the HMAC is fixed, such that equal IP addresses produce the same HMAC. For the analysis of the recorded traffic patterns it is unnecessary to know real IP addresses as long as the same IP address is represented by the same HMAC, such that only HMAC values are compared during analysis. Because of this sanitisation it is impossible to retrieve the IP address by just knowing the HMAC, thus, users' privacy and anonymity is guaranteed.

But in order to be able to compute false-positive and false-negative rates it is necessary to identify the traffic patterns belonging to the traffic generated by the controlled client. To achieve this the modified two Tor nodes can be configured to specifically mark the traffic from the controlled client in the traffic logs. This is done by specifying in Tor's configuration file the IP address of the controlled client at the entry node and the IP address of the destination of the generated traffic at the exit node. This procedure is not a problem for preserving user's privacy, because this marking only affects the traffic of the own controlled client and not traffic of other users.

4.2.3 Analysis Tool

The analysis tool is implemented in Python. It takes two file names of traffic logs as input, reads both files and parses them. Each unique circuit ID represents a unique traffic flow with sender and forward IP addresses. If an IP address is marked as coming from the controlled client in the traffic log, the flow gets a "marked" flag as well. For each second during the recording the cells belonging to a flow are counted as previously described. After both files are parsed, the tool computes the $n \times m$ correlations of all entry traffic flows with every exit traffic flow according to Formula 4.1. Then for each correlation the tool determines, if the correlation is a true-positive (i.e. a correct match), a true-negative (i.e. a correct non-match), a false-positive (i.e. an incorrect match) or a false-negative (i.e. an incorrect non-match):

- True-positive, if $r \geq t$ and both entry and exit flows are marked.
- False-positive, if $r \geq t$ and one or both flows are not marked.
- True-negative, if $r < t$ and one or both flows are not marked.
- False-negative, if $r < t$ and both entry and exit flows are marked.

The threshold is after preliminary tests set to $t = 0.7$ and the true-positives (TP), true-negatives (TN), false-positives (FP) and false-negatives (FN) are counted in order to compute the false-positive rate as $FPR = \frac{FP}{FP+TN}$ and the false-negative rate as $FNR = \frac{FN}{TP+FN}$ [46], which are the two relevant measures for the effectiveness of the attack.

4.2.4 Controlled Client

The controlled client generates Web surfing and bulk downloading traffic according to the experimental set-up as described in Section 3.3. A Python script using the Stem library[1] to control a Tor process starts a Tor client with a custom configuration needed for the experiments and performs download requests via the Tor client in order to simulate user Web surfing or bulk downloads. The Tor client is configured to always use Tor1HiG has its entry node and Tor2HiG as its exit node by specifying the options `EntryNodes` and `ExitNodes`, respectively, in the Tor configuration file. In addition, `EnforceDistinctSubnets` is set to zero. This ensures that Tor does not verify, if nodes on a path through the Tor network are on different subnets. Otherwise the client could not create any circuits, because Tor1HiG and Tor2HiG have consecutive IP addresses. Furthermore, the number of entry guards is set to one with `NumEntryGuards` and `UseEntryGuardsAsDirGuards` is set to zero. As a consequence of the latter the client does not use the entry guard while fetching directory information from the directory authorities in order to not produce additional traffic from the controlled client at the entry node. Only the explicitly generated traffic should be sent over the entry node during the experiments.

In addition to those configurations, two small modifications of the actual Tor client are needed for the experiment to function properly. First, the checks for the family of Tor nodes during path selection needs to be deactivated similar to the `EnforceDistinctSubnets` configuration option. Unfortunately, Tor does not provide the same configuration option for the family check, such that the source code must be modified directly. Second, at the time of the experiments the directory authorities did not assign guard bandwidth to both Tor1HiG and Tor2HiG in order to instruct clients to prefer the nodes for exit traffic. Because of that, the Tor client would not choose Tor1HiG as its entry node for a circuit and no circuit could be created as the client is not allowed to use any other node besides the controlled nodes. For this reason the client is modified to override the guard bandwidth assignment from the directory authorities for those two specific nodes. But these two minimal changes are the only modifications made to the Tor client. Otherwise the client works as any other Tor client.

4.3 Experiment

The experiment to study the effectiveness of the implemented end-to-end confirmation attack was carried out during the 7th calender week 2015. Data was collected between the 10. and 14. February 2015. Two data collections consisting each of one time Web surfing and one time bulk downloading traffic were conducted every day. In total 20 sessions of five minutes traffic were recorded and the stored traffic logs analysed afterwards.

[1] See `https://stem.torproject.org/`.

The experiments were executed in the following way. Both controlled nodes were configured to record five minutes of traffic and the controlled client was started one minute before the scheduled recording. The client immediately began to send traffic over both nodes. This ensured that the two nodes saw traffic from the client constantly during the whole five minutes period. The one minute lead time secured a stable client, which already created all necessary circuits for the traffic, such that only one single connection transporting the actual traffic was recorded at each node. After data collection the attack goal was to identify this connection during the analysis.

During the week of data collection Tor1HiG on average received 3.73 MB/s and sent 3.6 MB/s of traffic according to Tor Globe from the 16. February 2015. Tor2HiG received 3.89 MB/s and sent 3.76 MB/s. In comparison the average received and sent data for the top ten Tor nodes[2] was 35.126 MB/s and 34.435 MB/s, respectively. Based on data from Tor Metrics the average for *all* relays was 1.0 MB/s and 1.03 MB/s during this week[3].

4.4 Results

The results of the experiment are listed in Table 1 on the next page. For each session the false-positive rate, false-negative rate and the number of connections are given. The number of connections equals the number of connections at the entry node plus the number of connections at the exit node. The minimum number was 2433 and the maximum number 5358 depending on the amount of traffic transported by the two nodes during a session. The average number of connections was 3626. For all sessions the mean of false-positives was 0.000562. This implies that on average out of 3626 connections two connections were misclassified as being a match. In the table a value of 0.0 in the false-negative column means that the traffic from the controlled client was correctly identified by the attack, whereas 1.0 means that the traffic could not be detected. Only the two values 0.0 and 1.0 are possible, because during a single session there was only one connection from the controlled client to find, thus, this connection is either identified or not. In 19 out of 20 cases the connection was successfully recognised, i.e. the user could be linked to the destination. Only in one case this was not possible.

The results show no significant difference between the Web surfing and bulk downloading sessions regarding the false-positive rates, because the false-positives are predominantly caused by the cover traffic. Therefore, the single connection from the controlled client, either Web surfing or bulk downloading, does not make a significant difference. However, the experiment exhibited other important differences between the two cases. For Web surfing the number of recorded cells sent from the client to the exit node was on average 982, whereas in the bulk download case this number was 8450. This difference is as expected, because during Web surfing an up-to 60 seconds pause between two subsequent fetches of files is possible. In turn during bulk downloading traffic is generated for the whole five minutes period resulting in more sent cells. In addition, the correlation coefficient r is significantly different between Web surfing and bulk downloading, in the mean $r = 0.827$ and $r = 0.933$, respectively. If the one false-negative is not taken into account, $r = 0.981$ for bulk downloading, which nearly reaches the maximum value of 1. Unexpectedly, for this false-negative r was as low as 0.497. Why this outlier could occur is explained in detail in Subsection 4.4.2 below.

[2]For the list of the top ten Tor nodes in the network see `https://globe.torproject.org/#/top10`.
[3]The raw data used to calculate these numbers is available at `https://metrics.torproject.org/stats/`.

		False-positive rate	False-negative rate	#Connections
2015-02-10	Web 1	0.000731	0.0	3023
	Web 2	0.000641	0.0	3736
	Bulk 1	0.000551	0.0	3493
	Bulk 2	0.000571	0.0	4969
2015-02-11	Web 1	0.000589	0.0	3158
	Web 2	0.000395	0.0	4106
	Bulk 1	0.000696	0.0	4468
	Bulk 2	0.000667	0.0	3018
2015-02-12	Web 1	0.000606	0.0	2578
	Web 2	0.000416	0.0	3769
	Bulk 1	0.000594	0.0	2433
	Bulk 2	0.000638	0.0	2792
2015-02-13	Web 1	0.000455	0.0	4155
	Web 2	0.000504	0.0	4506
	Bulk 1	0.000518	0.0	3465
	Bulk 2	0.000421	0.0	4434
2015-02-14	Web 1	0.000618	0.0	2874
	Web 2	0.000629	0.0	3307
	Bulk 1	0.000627	0.0	2882
	Bulk 2	0.000378	1.0	5358
Mean		0.000562	0.05	3626

Table 1: False-positive and false-negative rates of the end-to-end confirmation attack

Not considering the outlier, bulk download traffic can be correlated very accurately. In addition to the nearly optimal mean of the correlation coefficient, the variance was as low as 0.00017. In contrast, the variance for Web surfing was 0.00521. But Web surfing traffic could still be accurately correlated. The higher variance in the latter case can be explained by the random delays introduced in the Web surfing traffic, such that the traffic patterns could vary between different sessions significantly. Additionally, the difference between the two types of traffic shows that as more user traffic is available for an attacker it is easier to match traffic with high accuracy.

4.4.1 Analysis

The experiment shows that an attacker can correlate traffic with very high accuracy and a low false-positive rate, especially if much traffic is sent over the nodes controlled by the attacker. As little as five minutes of traffic is enough. But what is the time window an attacker has to correlate traffic? Tor normally creates new circuits every ten minutes, if there are no open streams, i.e. open exit connections, on a circuit [47] in order to prevent an attacker from making user profiles. However, for long lasting connections such as instant messaging or large file downloads circuits are only rotated after the connection is closed. For those reasons the implemented and demonstrated end-to-end confirmation attack is a realistic threat to users' anonymity. In addition, by tweaking the threshold

value t the attacker can trade off the false-positive rate with the false-negative rate. In general, increasing the threshold yields a smaller false-positive rate, whereas the false-negative rate grows at the same time, and vice versa, because a higher threshold implies that traffic must be more similar in order to count as a match. Thus, an attacker, who wants be sure that a match is correct, chooses a high threshold. But this risks to not identify traffic. On the opposite with a low threshold more traffic is detected with the drawback of more false-positives.

During this experiment the attacker experienced approximately two false-positives during each session with an amount of traffic which is three to four times larger than the average for all Tor nodes. But what if one has more connections to correlate? Both controlled nodes together just saw approximately 0.1% of all traffic per second going through the Tor network. An attacker can either control nodes transporting more traffic, record traffic for a longer period of time or observe more Tor nodes. The number of false-positives grows to around six with 10 000 connections, to 56 with 100 000 connections and to 562 with 1 000 000 connections assuming that the false-positive rate stays constant. The false-negative rate is not very meaningful in this context due to the small sample size of only 20 user connections and the fact that the relatively high false-negative rate was caused by one single outlier. Because of that, it is impossible to make a definite statement about the false-negative rate with a higher number of connections.

In the experiment the attacker always knew, when the controlled client was correctly deanonymised, but in a real world attack one can never be that certain. What is the probability $P(A = B|A \sim B)$, i.e. the probability that flow A actually corresponds to flow B when A and B are correlated? In order to answer this question Equation 4.2 below can be employed. For the derivation of this formula see [5]. Assume that $P(A = B) = \frac{1}{n}$, where n is the number of connections, i.e. the attacker has no a priori suspicion about the connections from a user. The probability that two connections transport the same data is equal for all connections.

$$P(A = B|A \sim B) = \frac{(1 - FNR) \cdot P(A = B)}{(1 - FNR - FPR) \cdot P(A = B) + FPR} \tag{4.2}$$

With the false-negative rate $FNR = 0.05$, the false-positive rate $FPR = 0.000562$ and $n = 3626$ from the experiment the attacker has only a chance of 31.8% that two traffic flows were matched correctly. If n increases to 10 000, 100 000 and 1 000 000, this probability drops to 14.5%, 1.7% and 0.17%, respectively. Even if the false-negative rate is set to zero ignoring the outlier, these values change only very little. These numbers show a general problem for an attacker. On one hand the attacker wants to observe as much traffic as possible in order to increase the chances that one can see both the entry and exit traffic of users. But on the other hand this decreases the ability to draw relevant conclusions from the observations, because with increased traffic the chance of a correct match drops at the same time. However, if the attacker has an a priori suspicion about who is communicating with whom, i.e. $P(A = B) > \frac{1}{n}$, the equation changes to the attacker's favour [5].

The implemented end-to-end confirmation attack only considered cells transmitted in the forward direction, i.e. cells travelling in the direction from the client to the destination. As demonstrated this is already enough to accurately correlate traffic at the entry and exit nodes. However, the attack can be made even more accurate by incorporating

data transported in the backward direction, too, i.e. in the direction from the destination to the client. The reason beyond this is the fact that the generated HTTP traffic is asymmetric. The client sends a relatively small HTTP request to the destination, which answers with a much larger HTTP response, for example consider a HTTP request of some bytes or kilobytes at most to a 5 MiB response in the bulk download case. Therefore, in the backward direction much more data is transported than in the forward direction. As already seen with the difference in transmitted cells between Web surfing and bulk downloading more cells, i.e. more data, makes the correlation more accurate. Because of that, it is anticipated that the attack can be significantly enhanced by also correlating traffic in the backward direction. However, this was not experimentally verified in this thesis and is left for future work.

4.4.2 Explaining the Outlier

The outlier from the second bulk download session at the 14. February 2015 with a correlation coefficient of only 0.497 occurred very unexpected, because in the nine previous bulk download sessions the correlation coefficient was always nearly optimal, i.e. a near perfect match.

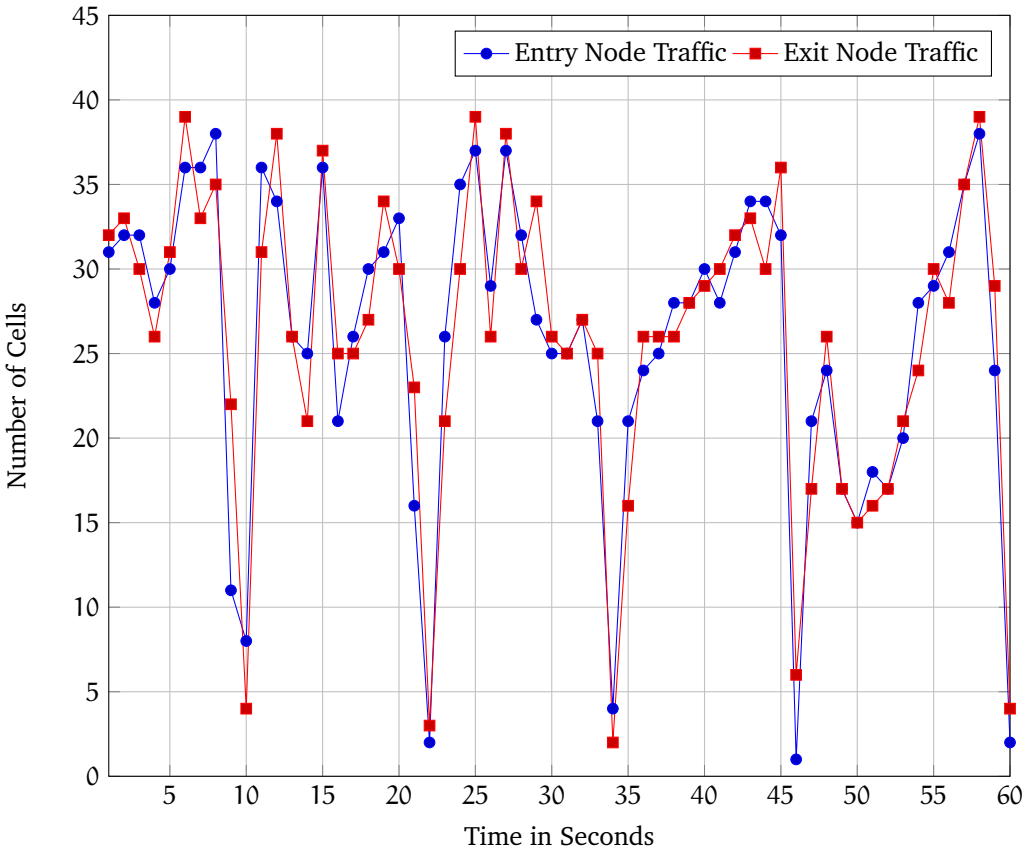


Figure 4: Session 2015-02-14 Bulk 1 – 60s plot of the number of received cells

In order to explain this outlier the received number of cells in one second intervals over the duration of the traffic recording was plotted for the outlier and “normal” cases.

Figure 4 above shows a 60 second extract for the first bulk download session at the 14. February 2015. In addition, Figure 5 below presents a similar plot for the outlier. Both plots illustrate the number of received cells at both the entry node and the exit node for a 60 second period.

By looking at Figure 4 both curves for the exit and entry traffic have to a great extend the same shape, i.e. both curves exhibit the same traffic pattern. If traffic drops at the entry node, traffic drops at the exit node as well. If the entry traffic raises again, the exit traffic also raises. From the figure it is evident that both flows follow the same traffic pattern, i.e. both flows are highly correlated resulting in a very high cross-correlation. In contrast, the curves for the outlier in Figure 5 do not exhibit this kind of behaviour. Whereas the number of received cells at the entry node is almost constant most of the time, the exit node curve shows an oscillating behaviour, i.e. traffic drops and raises in an alternating way frequently. Cells arrive at the exit node in bursts. This explains why the algorithm thinks that both flows are not correlated. They clearly do not exhibit the same traffic pattern, although the *total* number of cells is the same.

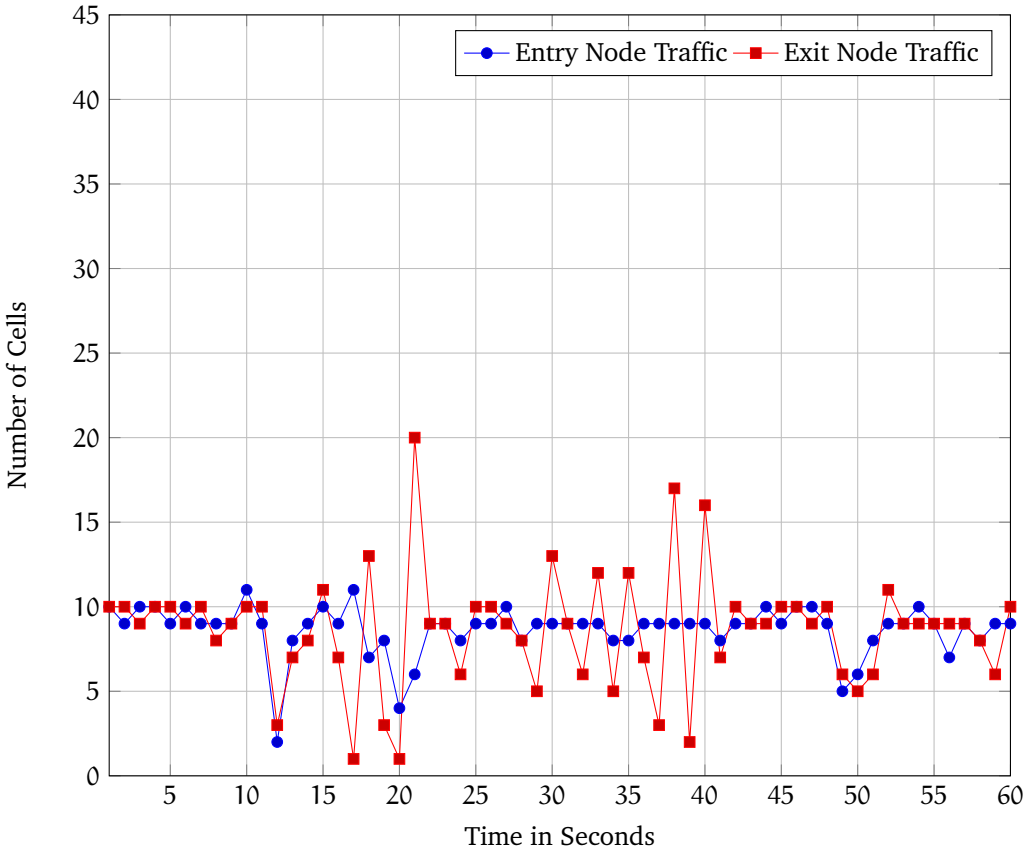


Figure 5: Session 2015-02-14 Bulk 2 – 60s plot of the number of received cells

But why does the traffic arrive in bursts at the exit node? To the author's knowledge this is best explained with Tor's scheduling algorithm [48]. When there exists more than one circuit between the same two Tor nodes, the circuits are multiplexed over one single TLS connection between the two nodes [2]. In the case one node has cells from multiple

such circuits to send to the next node, the node must decide what cells are transmitted first. Tor keeps a record of the number of cells transported in the past for each circuit, which is weighted in the way that recent cells count more than older cells, and sends the cells from the circuit with the least number of cells transmitted in the past. This effectively prioritises circuits with Web surfing traffic over bulk download traffic, because the latter requires much more cells to be transported through the network [48]. When multiple Web surfing circuits and one bulk downloading circuit compete who is scheduled next for sending cells, the Web surfing circuits are prioritised over the bulk downloading circuit. Thus, the latter circuit queues up cells and transmits cells in a burst, when it is the circuit's turn to send cells. Such a competing situation at the middle node explains why the cells arrive in bursts at the exit node in the outlier case. But it is unknown, whether this outlier was a unique occurrence or whether this could happen more often. More experiments would be needed to make a definite statement.

4.4.3 Summary

The validation of the implemented end-to-end confirmation attack with an experiment on the live Tor network demonstrates that such attacks are a realistic threat against Tor. User traffic can be accurately correlated through timing analysis with a very low false-positive rate. But at the same time the analysis of the results show that an attacker's success decreases significantly even with such a low false-positive rate when the number of observed connections increases.

These results answer research question 1: "Can end-to-end confirmation attacks be successfully applied against the current size Tor network? How good is such an attack, i.e. what is the false-positive, what the false-negative rate of it?" – Yes, end-to-end confirmation attacks can be successfully applied against the current size Tor network. And yes, they can be very accurate with a very low false-positive rate. But at the same time, the results can guide the development of a defence against end-to-end confirmation attacks as described in the next chapter.

5 Defence against End-to-End Confirmation Attacks

The results from the previous chapter have a good and a bad implication for the Tor network. On the bad side, they demonstrate that end-to-end confirmation attacks are a real threat against Tor, because they can be carried out with high accuracy. But on the good side, the results also show that attackers may not be able to draw relevant conclusions from their attacks, if they observe a lot of connections. This becomes an increasingly bigger problem for attackers, because the Tor network is growing constantly over time. More people are using Tor everyday and more and more traffic is transported through the network. Because of that, it is not necessary to find a *perfect* defence, which protects under all circumstances against end-to-end confirmation attacks. Such a defence is most likely not possible without either completely redesigning Tor or introducing a prohibitive large amount of cover traffic, which just makes the network unusable. On the other hand, the results suggest that it could be enough to only increase the false-positive and false-negative rates a little bit, such that an attack may not yield any relevant results.

In this chapter a defence against end-to-end confirmation attacks is developed and it is studied how effective the defence can protect against such attacks in the Tor network. First, previously proposed defences are reviewed and it is analysed, whether and how they could be implemented in Tor. Based on this a new defence is proposed and its implementation in Tor is described. Furthermore, the effectiveness of the defence is analysed by repeating the attack experiment on the live Tor network plus a large-scale simulation of Tor with Shadow. Together this answers research question 2.

5.1 Review

In the context of Tor two design criteria besides stronger anonymity for a concrete defence are important: *deployability* and *usability*. A defence needs to be easily deployable. If a defence relies on the cooperation of Tor nodes, it is unrealistic to assume that it is possible to deploy a new Tor version to 7000 nodes simultaneously. Therefore, the defence still needs to work, if not all nodes taking part in a circuit operate on a Tor version implementing the defence. Because of that, a client-side defence is preferable, i.e. the defence is implemented in most parts inside the Tor client not the Tor nodes. In this case the defence should additionally work stand-alone without the need of other clients implementing and using the same defence. This means that a user immediately achieves better anonymity once the defence is activated.

Equally important is usability as highlighted in Subsection 2.1.1. A defence should not decrease the usability of Tor too much. This includes added latency or bandwidth requirements. Latency is especially an issue with Tor, because Tor's goal is to support interactive applications, where latency is a crucial problem. Higher bandwidth requirements increase the costs for operators to run Tor nodes. But diversity in Tor nodes is important in order to defeat attackers, which can observe parts of the network, for example a single country or an Internet exchange point, by increasing the number of nodes they cannot observe [11]. In addition, more nodes mean more bandwidth and better performance for users. In this context it is important to recall that usability and anonymity

needs to be balanced carefully in an anonymity system, because more users mean more cover traffic and stronger anonymity [3]. If a defence decreases the performance of the network, it can restrain users from using it, which in turn results in *less* anonymity for the remaining users [4]. Because of that, a usable defence increases the user's anonymity, but at the same time does not decrease the network's performance too much. Thus, the gained anonymity must outweigh the loss of usability.

In order to defend end-to-end confirmation attacks two basic measures can be utilised in an anonymity system. The introduction of *packet delays* or *dummy traffic*. With packet delays incoming packets are delayed for some time, before they are forwarded by nodes. Thus, the timing of an outgoing packet is independent of the timing of an incoming packet. Dummy traffic, sometimes also called *link padding*, creates and sends dummy packets, i.e. packets without real user traffic, in order to make different connections indistinguishable. In the context of Tor a packet is a Tor cell. With both techniques the goal is to either decrease the similarity between traffic coming from the same user or increase the similarity between traffic coming from different users. The former increases the false-negative rate of an attack, because it decreases the correlation of traffic from the same user captured at the entry and the exit nodes. The latter increases the false-positive rate, because traffic from independent users have a higher correlation. Packet delays work at the expense of latency, whereas dummy traffic requires more bandwidth. But additionally, dummy traffic can also negatively impact latency due to network or node congestion caused by the added dummy packets.

5.1.1 Packet Delays

In [28] a single mix system for low-latency anonymous traffic is analysed, which delays packets independently from each other before they are forwarded by the mix. Each packet is delayed according to a random variable following the exponential distribution. The mix does not adapt its delaying strategy, for instance depending on the arrival rate of packets. It is shown that using the exponential distribution provides the theoretical best anonymity compared to other statistical distributions. However, despite this theoretical optimality the paper presents an attack against this delaying strategy based on a hypothesis test, which decides if two traffic patterns are the same with regards to some specified confidence interval. This shows that even a theoretical optimal delaying strategy can be defeated, even if traffic is routed over multiple mixes using the same delaying strategy.

In addition to the results above, delaying of packets is not a feasible defence against end-to-end confirmation attacks in the context of Tor, because Tor's low-latency requirement is especially sensitive to packet delays. Users do not accept even longer delays than already experienced and Tor tries to decrease latency [48]. Thus, adding additional latency would thwart these efforts. Furthermore, consider the timing attack implemented in the previous chapter. It uses time windows of one second. For this reason a packet received in the beginning of one window must be delayed for at least one second to be counted in the following time window. But one single packet does not disturb the correlation very much. In addition, packets must be spread over a longer period of time and more consecutive time windows in order to decreased the correlation significantly. But if the delay strategy is known the attacker can adapt the attack. Assume that packets are delayed between the entry and the exit node at random according to a statistical distribution with a mean of ten seconds and a variance of three seconds. The attacker

just increases the time windows to three seconds and compares traffic at the entry node with traffic at the exit node shifted by ten seconds. This will decrease the correlation, because some packets will be delayed longer than three seconds and counted in a wrong time window, but the attacker can compensate this by longer observations. Note that the spread is the critical factor here. In the best case traffic would be spread in such a way that traffic bursts, which constitute the distinctive traffic patterns of interactive applications, are removed from the traffic. But this introduces an unacceptable large latency for interactive applications.

For the described reasons the author argues that packet delays are not a feasible countermeasure against end-to-end confirmation attacks in Tor.

5.1.2 Dummy Traffic

In an anonymity network two types of dummy traffic or link padding can be used. The first type exchanges dummy packets between adjacent nodes in the network, between the client and the first node, or between the last node and the final destination. The last option is only possible, if the destination is part of the anonymity network. The second variant of dummy traffic is *long-range padding*, where dummy packets are not sent to adjacent nodes but to nodes later on a path. In many cases the client creates dummy packets, which traverse multiple nodes in the network until the packet is dropped by one node. Depending on the protocol intermediary nodes may or may not recognise those dummy packets. Malicious nodes receiving dummy packets can always filter out those packets in order to enhance an attack by ignoring the dummies. The same does not apply to network-level adversaries, who just observe encrypted traffic inside an anonymity network. Both adversaries must be taken into account while a protection mechanism using dummy traffic is designed.

Tor already supports both variants of dummy traffic [23]. The first can be achieved with `padding` or `vpadding` control cells transmitted to adjacent nodes on a circuit, the second with `drop` relay cells. Note that only the receiver of a `drop` relay cell can recognise the dummy cell, but no other node on the circuit. If a Tor node receives a dummy cell, the cell is ignored. Despite the fact that both types of dummy traffic are already supported by Tor, no concrete padding scheme is currently implemented [23].

Defensive Dropping

A variant of client-side long-range padding is proposed in [5] and called *defensive dropping*. The client sends dummy packets to intermediary nodes on a path, which drop them. Thereby the receiving node is chosen randomly. Both real packets and dummy packets together are transmitted at a constant sending rate. Because of that, the attacker sees at the entry node only a constant flow of packets. At the exit node traffic exhibits a different pattern, because dummy cells are dropped at intermediary nodes. This decreases the correlation between traffic at the entry node and traffic at the exit node originating from the same user. In addition, the correlation between traffic from different users is increased, if they utilise defensive dropping as well. However, they must transmit packets at the *same* constant sending rate in order to make the traffic looking similar [6]. The original paper demonstrates that defensive dropping can be effective against the exact same timing attack also employed in this thesis, with an equal error rate of 0.1 to 0.3. On the downside this defence adds extra latency due to the constant sending rate in addition to the bandwidth needed for the padding. If the application wants to send data with a

higher sending rate than the fixed constant sending rate, packets need to be delayed in order to fulfil the constant sending rate requirement.

In the forward direction defensive dropping can easily be implemented in Tor. Long-range `drop` relay cells already exist, such that only the Tor client needs to be modified to use the defence. In contrast, it is more difficult to implement the defence in the backward direction, because the final destination is not aware of Tor and therefore cannot generate dummy traffic. A way to implement defensive dropping in the backward direction is that the nodes on a circuit send `drop` relay cells to the client [5]. This requires a modification to Tor nodes. Moreover, nodes need to be honest and must participate in the scheme. For example, in the experimental set-up of the attack in this thesis both the malicious entry and exit nodes could just not send dummy packets. Thus, only the middle node would generate dummy traffic. In addition, without the cooperation of the nodes it is impossible to guarantee a constant packet rate between the entry node and the client. But such a coordination between nodes is currently not intended in Tor and could be exploited by malicious nodes.

Defensive dropping could be gradually deployed to Tor, because clients and nodes can independently start using the defence, assuming there is no coordination protocol between nodes to guarantee a constant packet rate in the backward direction. Even if only a single user uses the defence, one should gain additional protection. Nevertheless, the full protection of defensive dropping is only achieved if all clients and nodes utilise the defence.

Adaptive Padding

Adaptive padding as suggested in [6] is fundamentally different to defensive dropping, because it is a defence implemented at the nodes of an anonymity system. It assumes that a node knows the statistical distribution of intervals between two consecutive packets, i.e. the time between two received packets. After each forwarded packet a value is randomly draw from this distribution. This gives a time until the node expects the next packet. If no packet is received during this time frame, the node sends a long-range dummy packet to the final destination. If a packet is received, the packet is forwarded, a new random value chosen and the process repeated. As illustrated in Figure 6 below, adaptive padding basically injects dummy packets (red strokes) between real packets (blue strokes), if the delay between two consecutive packets becomes to large. For a detailed description of the adaptive padding algorithm see [6].

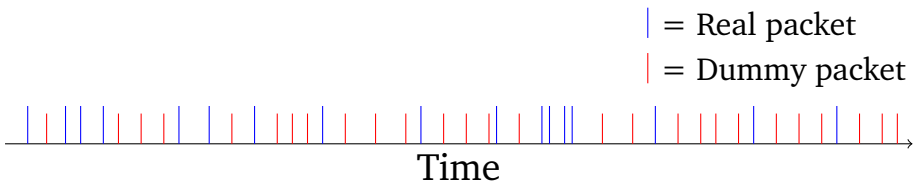


Figure 6: Graphical illustration of Adaptive Padding

The defence works by reducing the correlation between traffic from the same user at the two attacking nodes, because distinctive gaps between packets are filled by dummy packets. Therefore, the defence is functional in isolation, even if only one user uses it [6]. The paper claims that adaptive padding performs better than defensive dropping against

the same timing attack with an equal error rate of up to 0.48. Furthermore, adaptive padding itself does not add any extra latency, because real user traffic is not artificially delayed. It only requires additional bandwidth for the dummy packets. One difference between adaptive padding and defensive dropping is that the former pads the connection between the exit node and the final destination, whereas the latter pads the connection between the client and the entry node. In contrast to defensive dropping, adaptive padding can also protect against active attacks, where an attacker drops packets at malicious nodes or introduces artificial bursts by delaying user packets in order to increase the correlation of traffic. Packet drops are thwarted, because introduced gaps are repaired again by subsequent honest nodes on a path. Traffic bursts caused by an attacker can be reduced by delaying bursty packets for a short period of time in order to flatten out and remove the burst [6]. However, this works at the costs of extra latency.

In the proposed form adaptive padding is impossible to implement in Tor, because no dummy traffic can be sent to the final destination. In addition, nodes cannot create `drop` relay cells in the forward direction due to the reason that they do not share session keys with the subsequent nodes on a circuit. Nodes only share session keys with clients. Therefore, they can only transmit `padding` or `vpadding` control cells to adjacent nodes, which can be filtered out by the receiving node. Because of that, long-range padding by Tor nodes is only possible in the forward direction, if such cells are provided by the client [6]. But this would need a protocol change to Tor and it would break Tor's integrity protection. The running digest used for integrity protection must be synchronised between both the client and each node [23]. But the client does not know, when a node would insert a dummy cell, thus, it is impossible to supply dummy cells in advance, which do not break the synchronisation. Nevertheless, adaptive padding can be implemented in the backward direction in the same way defensive dropping could be implemented.

Dependent Link Padding

A completely different approach is followed by the *dependent link padding* algorithm as proposed in [7]. The main idea is to make at one single node for all incoming flows all corresponding outgoing flows indistinguishable as depicted in Figure 7 on the next page. In fact, the algorithm creates a sending schedule, which is exactly the same for all flows forwarded by a node. On all outgoing flows packets are sent at fixed points in time. If a flow has no real user packet to forward at the next sending point, a dummy packet is sent instead. Thereby the packet delay is bound by a fixed value, i.e. it is guaranteed that packets are *never* delayed longer than the specified delay bound. Dependent link padding can provide full anonymity [7], because all outgoing flows have the exact same sending schedule and are therefore indistinguishable by an attacker. In addition, flows can be grouped in order to provide anonymity inside the same group. This can improve efficiency, because the outgoing sending rate must be higher than the highest incoming receiving rate to satisfy the delay bound. If flows are grouped by their sending rate, groups with a lower sending rate do not need to generate much dummy traffic. In the case flows with high and low sending rates would be combined, all flows must transmit at the same rate, thus, flows with low sending rate would need to create a lot of dummy packets to achieve the higher sending rate. Moreover, traffic could be grouped by the type of application traffic they are transporting, for example Web surfing and bulk downloading traffic. The dependent link padding algorithm can also be extended to anonymity

networks, where all nodes would use the algorithm. But then the overall delay bound needs to be increased.

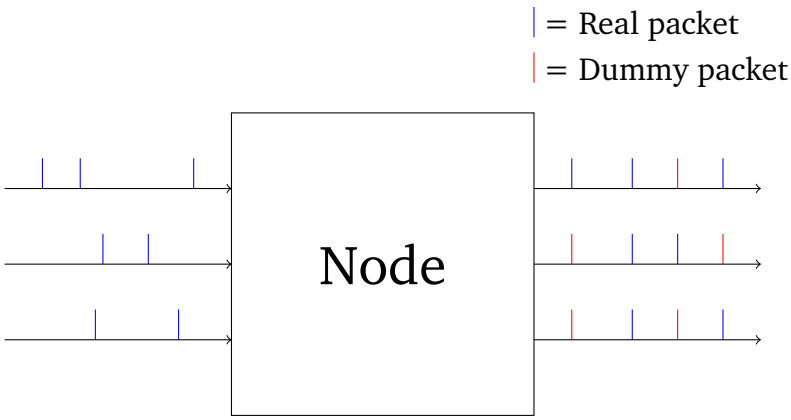


Figure 7: Graphical illustration of Dependent Link Padding

Dependent link padding has the advantage that the maximum delay can be fixed, i.e. the introduced latency can be controlled. But at the same time this is problematic, because a smaller delay bound needs a higher sending rate in order to satisfy the delay bound. This in turn requires more bandwidth, because the time frame to receive real packets is smaller as well, thus, more dummy packets must be transmitted. Therefore, if no packets are dropped, the sending rate is unlimited by smaller delay bounds [7]. This introduces a denial-of-service opportunity for an attacker. A single incoming flow with a high sending rate can force *all* other flows to forward packets with a sending rate as least as high as the rate from the attacker. In order to defend against this attack the sending rate can be limited. But then packets could be dropped, because they cannot satisfy the delay bound. In Tor packet drops are undesirable, since they cause circuits to fail as the integrity protection becomes unsynchronised [32]. In addition, if only one flow sends real packets, still all other flows must generate dummy traffic at the same sending rate. With bursty Web traffic the flows needs to be padded between the burst as well. Furthermore, Tor creates circuits preemptively, thus, a lot of unused circuits are present in the network, which must be padded all the time generating a lot of dummy traffic.

Dependent link padding has exactly the same implementation issues as adaptive padding, because in both algorithms the nodes must generate long-range dummy traffic to the final destination in order to protect against malicious nodes filtering out dummy packets. In Tor only clients can send long-range dummies in the forward direction as already pointed out. Additionally, both algorithms rely on honest nodes, which follow the protocol. However, dishonest nodes on a path can be to some degree mitigated by subsequent honest nodes on the same path employing the defence. Finally, dependent link padding would actually be a completely new scheduling algorithm for Tor.

5.2 Description

The defence proposed in this thesis is a modification of the adaptive padding algorithm in order to be able to implement the algorithm in Tor efficiently. Ideas from both defensive dropping and adaptive padding are combined to form the proposed algorithm. In the forward direction *client-side adaptive padding* is implemented. The client sends long-

range `drop` relay cells to the *middle* node on a circuit, which drops the received dummy cell. This ensures that a malicious entry node cannot filter out dummy cells. At the same time the number of cells recorded at the entry node and the exit node are different, because the dummy cells do not reach the exit node. As a result the correlation between the same traffic at both nodes is reduced. The adaptive padding algorithm is used by the client to decide when a padding cell should be transmitted. In the backward direction the standard adaptive padding algorithm can be employed, where the nodes on a circuit send long-range `drop` relay cells to the client according the adaptive padding strategy.

The padding starts once a circuit is fully established. This is done in order to obscure the start of a user's conversation, i.e. the point in time when a user opens a connection to a destination. This generates dummy traffic for unused circuits requiring more bandwidth. Alternatively, to save bandwidth padding could be started once a stream is attached to a circuit, i.e. a connection to the destination is established. But then the start of the conversation is not hidden inside dummy traffic, which can be exploited by an attacker [26]. The set-up of a circuit is not concealed with dummy traffic, because setting up a circuit is a slow process involving public-key cryptography and the circuit creation may fail anyway. Thus, only fully established circuits are padded.

This proposed defence inherits several advantages from adaptive padding. The defence can protect against timing attacks, because it fills distinctive gaps inside traffic and thereby reduces the correlation within the same traffic flow. Furthermore, no artificial delays are introduced in traffic flows. On the other hand, unused circuits take up bandwidth for the dummy traffic and latency may be added due to network or node congestion. Additionally, the advantage of Tor's scheduling algorithm is thwarted, because all flows are effectively bulk traffic now, thus, the ability to prioritise Web traffic is removed.

5.2.1 Modified Adaptive Padding

The standard adaptive padding algorithm assumes that the distribution of *inter-packet intervals*, i.e. the time between two consecutive packets, is known. But this distribution is different for every kind of traffic, for instance Web surfing, bulk downloading, instant messaging, remote terminal connections and so forth. Because of that, a learning algorithm is proposed, which adapts over time according to the transported traffic. The main data structure is an array of *bins*. Each bin represents a disjunct range of inter-packet intervals, where the last bin represents all values greater than a specific threshold. In the same way as standard adaptive padding the range of bins is increased exponentially with $2^i, 0 \leq i \leq n$ where n is the number of bins. One bin i starts at the end of the previous bin $i-1$ and goes to 2^i. Thus, shorter intervals are split into more bins than longer intervals. For simplicity every bin is implemented as a counter. Every time a node receives a packet the interval to the previously received packet is calculated and the corresponding bin counter incremented. In the case of the client the interval between two sent packets is measured. Note that in the client case only the time between real user packets is taken into consideration, dummy packets are ignored. This method basically determines the statistical distribution of inter-packet intervals for a specific connection and becomes more accurate as more traffic is transported.

With the above the adaptive padding works as the following. Every time a packet is received by a node or transmitted by a client a bin is randomly chosen according to the distribution, i.e. bins with higher counter values have a higher probability to be selected.

Then inside the bin an *expected inter-packet interval (EIPI)* is uniformly chosen at random. The EIPI gives a value until a new packet is expected. If a new packet does *not* arrive until the EIPI is expired, a long-range dummy packet is sent. If a packet arrives before the EIPI is expired, the packet is forwarded. In both cases a new EIPI is generated and the procedure repeated.

The algorithm should avoid sending dummy packets during bursts, i.e. when packet density is high, in order to operate more efficiently. For this reason the same strategy as the *dual-mode* described in [6] is followed. When a real user packet has been received, the EIPI is only selected from the higher bins associated with longer inter-packet intervals. This reduces the chance to send dummy packets in the case packet density is already high. When a dummy packet was transmitted, the EIPI is chosen from the lower bins associated with shorter intervals in order to fill the gaps in sparse traffic. Lower and higher bins can be separated based on the observed distribution of inter-packet intervals.

5.3 Implementation

In this thesis only the client-side adaptive padding algorithm is implemented and validated, i.e. only dummy traffic in the forward direction is generated. Also implementing the defence in the backward direction would be very similar. The only difference would be that instead of modifying the Tor client Tor nodes need to be altered. This is again left for future work. The implementation is relatively straightforward following the description above. For the number of inter-packet interval bins $n = 10$ is chosen, thus, the inter-packet intervals stated in milliseconds and exponentially distributed go from $2^0 = 1$ msec to $2^{10} = 1024$ msec. The first $\frac{n}{2}$ bins represent the lower bins and the second $\frac{n}{2}$ bins the higher bins, i.e. the distribution is just split in the middle to separate the lower and higher bins. Each bin is initialised with 1. Because of that, after the construction of a circuit the inter-packet intervals are uniformly distributed.

In order to sample from this distribution *Vose's Alias Method* is implemented according to [49]. This algorithm allows to efficiently sample from an arbitrary discrete probability distribution. Vose's Alias Method should not be described here, because it is unnecessary to know the algorithm for understanding the implementation of the defence. The interested reader is referred to [49] for a detailed explanation of the method. The only important point to know about Vose's Alias Method is the fact that this method allows to sample from a discrete probability distribution in constant time after an initialisation step of liner time complexity.

Each time a real user cell is sent, the time interval to the previous real user cell is determined and the corresponding bin counter incremented. Every 100 cells the alias method is reinitialised separately for both the low and high bins. Between two initialisations values are sampled from the same two low and high bin distributions in constant time. First, a bin is selected with the alias method and than a value is chosen uniformly at random inside the range represented by the bin. This is a trade-off between always using the most current inter-packet interval distribution and the time to sample values. In this case every 100 cells the sample distributions are updated with the current distribution of inter-packet intervals. For this reason the algorithm better adapts over time to the pattern of the transported traffic. This adaptation and sampling is performed for every circuit individually.

After a circuit is fully constructed and ready to be used for transporting traffic the

padding of the circuit starts. The inter-packet interval counters and the alias method are initialised. Additionally, a timer is started with a value sampled from the high bins. If the timer expires before a cell is sent, a `drop` relay cell is transmitted to the middle node and the timer restarted with a value sampled from the low bins. In the case a cell is sent before the timer expires, the timer is just restarted with a high bin sample value. This procedure is repeated until the circuit is destroyed.

Furthermore, two Tor configuration options are added. One option specifies if the defence is activated or not, and the second option determines whether unused circuits are padded. If the second option is activated, only circuits which have a stream attached to it actually send `drop` relay cells. Nevertheless, the distribution of inter-packet intervals is still measured and preserved. Both options are necessary for the following experiments.

5.4 Experiment

In order to study the effectiveness of the implemented defence against end-to-end confirmation attacks the attack experiment from Chapter 4 is repeated. The experiment is carried out in exactly the same way as the first experiment. The only difference is that this time the implemented defence is activated inside the controlled client. Dummy traffic is only generated for the circuit sending the actual traffic. Unused circuits are not padded in order to avoid that multiple circuits send traffic over the controlled nodes at the same time. Only one connection should be visible during the experiment. Then the goal of the attacker is the same as before, identifying the single client connection going over both Tor nodes. This tests the defence in isolation when only a single client is using it. With a functioning defence an attacker would not be able to identify the user's traffic, because traffic should show different patterns at the entry node and at the exit node, i.e. in total the false-negative rate of the attack should increase.

The experiment was conducted during the 11th calender week 2015 and data was collected between the 10. and 14. March 2015. During this week Tor1HiG on average received 3.55 MB/s and sent 3.44 MB/s of data, whereas Tor2HiG received 4.01 MB/s and sent 3.88 MB/s. On average the top ten Tor nodes received and sent 38.504 MB/s and 37.754 MB/s, respectively, while all Tor nodes received 1.14 MB/s and sent 1.17 MB/s on average during this week.

5.5 Simulation

The above experiment can only study the effectiveness of the defence for one single client. But it is also important to investigate the effectiveness when *all* clients use the same defence simultaneously, because the traffic of different clients at the entry node should look more similar when they utilise the same defence as the defence injects dummy packets in order to fill distinctive gaps in the traffic. For this reason the effect on the false-positive rate of the attack should be studied in the case all clients utilised the defence at the same time. In order to accomplish this a large-scale simulation of the Tor network with Shadow is carried out. For this a testing Tor network with the following configuration is deployed inside Shadow [50]:

- 4 Tor directory authorities
- 500 Tor nodes
- 500 Web servers
- 1350 Web surfing clients

- 150 bulk downloading clients
- 300 Torperf clients

The ratio of 9:1 Web surfing to bulk downloading clients is consistent with research into the usage of Tor, which showed that the vast majority of users utilise Tor for Web surfing activities [51]. Both categories download data from the Web servers according to the same user model as employed during the experiments on the live Tor network as well. There is only one small difference, because Shadow currently cannot perform the Web surfing model exactly, such that 2000 KiB of data is downloaded once instead of 40 times 50 KiB. In addition, the Torperf clients – 100 in each case – download either 50 KiB, 1 MiB or 5 MiB of data, respectively, and wait one minute between consecutive downloads. The network was generated with tools included in Shadow based on the following data sources [50]:

- The Alexa list of the 1 million most popular Web sites from the 12. March 2015
- Tor Metrics' number of clients from the 12. March 2015
- The consensus document from the 28. February 2015 at 23:00
- Relay descriptors and extra information documents from February 2015
- The Internet topology bundled with Shadow

This ensures that the simulations use a Tor network, which replicates a stripped down but still realistic Tor network as closely as possible in order to be able to transfer the obtained results to the real Tor network. Inside this simulation network two entry nodes and two exit nodes are configured as attacking nodes each logging five minutes of data for the end-to-end confirmation attack. Two victim clients – one Web surfing and one bulk downloading client – route their traffic over these nodes during the logging intervals. This basically reassembles the attack experiment inside one simulation.

In total three simulations are carried out. In the first simulation the defence is not used at all. This gives the baseline data of the simulations. In the second simulation the defence is activated on all Web surfing and bulk downloading clients without padding unused circuits (called *limited defence*). In the third simulation padding is also activated for unused circuits except for the victim clients (called *full defence*). After each simulation the logs from the attacking nodes are gathered and analysed using the self-developed analysis tool in order to study the effectiveness of the defence in the two defence scenarios compared to the baseline data. One simulation runs for 90 virtual minutes, where the first 30 minutes are used to bootstrap the network with all nodes and clients in order to ensure a stable network during the remaining time. The simulations are executed on Amazon EC2 using Shadow version 1.10.1.

5.6 Results

Table 2 on the next page lists the results from the experiment with enabled defence. The number of observed connections was less compared to the first experiment studying the attack, on average 2630 connections were recorded by the attacking nodes (minus 996). The mean of the false-positive rate increased a little bit to 0.000642, but did not change significantly compared to the previous experiment, which had an average false-positive rate of 0.000562. This is as expected, because during this experiment only the controlled client used the defence, such that the cover traffic was not changed artificially. In contrast, the false-negative rate increased to 0.9. Only in two out of 20 cases the attack could

successfully identify the connection from the controlled client. In the other 18 cases the defence could hide the correlation between the traffic at the entry and the exit nodes.

		False-positive rate	False-negative rate	#Connections
2015-03-10	Web 1	0.000689	1.0	2038
	Web 2	0.000696	1.0	2657
	Bulk 1	0.000623	1.0	2209
	Bulk 2	0.000683	1.0	2903
2015-03-11	Web 1	0.000632	1.0	2102
	Web 2	0.000610	1.0	3006
	Bulk 1	0.000647	1.0	2342
	Bulk 2	0.000601	0.0	2908
2015-03-12	Web 1	0.000663	1.0	2636
	Web 2	0.000609	1.0	3161
	Bulk 1	0.000667	1.0	2613
	Bulk 2	0.000631	1.0	3175
2015-03-13	Web 1	0.000691	1.0	2534
	Web 2	0.000598	1.0	3017
	Bulk 1	0.000691	0.0	2602
	Bulk 2	0.000604	1.0	2958
2015-03-14	Web 1	0.000620	1.0	2263
	Web 2	0.000579	1.0	2507
	Bulk 1	0.000663	1.0	2273
	Bulk 2	0.000650	1.0	2696
Mean		0.000642	0.9	2630

Table 2: False-positive and false-negative rates of the end-to-end confirmation attack with enabled defence

During the Web surfing sessions the correlation coefficient between the traffic from the controlled client at both nodes was reduced to $r = -0.254$ on average with a variance of 0.063. For the bulk downloading sessions the correlation coefficient was on average $r = 0.445$ with a variance of 0.154. This means that by sending dummy packets the defence was able to destroy the correlation between the traffic. Especially effective was the defence in the Web surfing cases. The difference between Web surfing and bulk downloading can be explained, if the ratio of transmitted dummy cells to non-dummy cells is considered as shown in Table 3 below. For Web surfing roughly 10 dummy cells were sent for each non-dummy cell, with about a maximum ratio of 17:1 and a minimum ratio of 4:1. In the bulk downloading case only one dummy cell was transmitted for every second non-dummy cell. This explains the difference in the correlation coefficients, because during Web surfing more dummy traffic was generated than during bulk downloading. This does not come as a surprise, because with Web surfing the defence needs to fill the idle times with dummy cells when no traffic is sent, whereas this does not take place in the bulk downloading case, where traffic is transmitted continuously.

In the bulk downloading case the defence is very effective in terms of the ratio between dummy cells and non-dummy cells. With very little dummy traffic the correlation could be destroyed. However, the two successful attacks identified bulk downloading traffic. In these two cases the defence could not prevent the attack, because only very little dummy traffic was sent, with a ratio of 0.22:1 and 0.19:1, respectively. This means that on average as an absolute minimum after every third non-dummy cell a dummy cell needs to be sent in order to defend against the attack. With Web surfing the correlation coefficient was $r = 0.126$ in the case of the least dummy to non-dummy cell ratio of 4.46:1. This shows that the defence sent too much dummy cells during Web surfing in all cases. A ratio of 4:1 is already enough to completely destroy the correlation. A lower ratio could be achieved by reducing the number of dummy cells during idle times. As currently implemented the defence samples from the low bins during idle times. This could be optimised in order to send less dummy traffic during these periods.

	Web 1	Web 2	Bulk 1	Bulk 2
2015-03-10	9.41:1	11.61:1	0.81:1	0.90:1
2015-03-11	4.46:1	9.24:1	0.43:1	0.22:1
2015-03-12	16.94:1	9.36:1	0.69:1	0.26:1
2015-03-13	10.95:1	9.47:1	0.19:1	0.33:1
2015-03-14	5.65:1	7.97:1	0.38:1	1.16:1
Mean	9.51:1		0.54:1	

Table 3: Ratio of transmitted `drop` relay cells to non-dummy cells

5.6.1 Simulation

The results from the Shadow simulations are presented in Table 4 on the next page. The table shows the results from the attacks carried out during each simulation run. During all three executions of the simulation the number of recorded connections was approximately the same compared to the attack experiment from Chapter 4. This gives confidence, that the simulations reassembled a realistic Tor network, such that the obtained results can be transferred to real Tor network. During the first run with the defence disabled the attack could successfully deanonymise the victim clients in both cases. With one exception this was not possible, after the defence was activated, both during the limited defence and the full defence simulations. The exception occurred during the full defence simulation in the Web surfing case. In this case the attack recorded three connections from the victim client. The reason for this could be for example that the circuit failed and a new circuit was created for transmitting traffic afterwards. In this case the attack was able to identify two out of the three connections.

But most interesting is the difference of the false-positive rates between each simulation run, because this can answer the question how the defence behaves in the case all clients are using it. The first observation is that the false-positive rate is in all cases in the same order of magnitude compared to the experiments on the live Tor network. Although the absolute values are less compared to the live experiments, but again, this gives confidence that the simulations are realistic and accurate. The lower false-positive rates only show that the attack is easier to carry out during simulation. The second and

most important observation is that the false-positive rate does not change very much, when the defence is enabled. Especially, the rate does not increase, it even decreases a little bit. The reason for this is the fact that most false-positives occur, when the attack erroneously correlates unrelated cover traffic. The defence changes the patterns at the entry node, which further destroys the correlation between unrelated traffic. These results imply that the defence works in isolation. Users only receive protection from the defence by destroying the correlation between traffic patterns at both the entry and the exit nodes. They do not gain further protection, when other users also use the defence.

	False-positive rate		False-negative rate		#Connections	
	Web	Bulk	Web	Bulk	Web	Bulk
Without defence	0.000254	0.000262	0.0	0.0	3666	3767
Limited defence	0.000180	0.000244	1.0	1.0	3673	3669
Full defence	0.000215	0.000196	0.33	1.0	3501	3878

Table 4: False-positive and false-negative rates of the end-to-end confirmation attack during large-scale simulations of the Tor network

5.6.2 Analysis

In order to better understand how the implemented defence is working the number of recorded cells at both the attacking entry and exit nodes is plotted in Figure 8 and Figure 9 for a Web surfing and a bulk downloading session, respectively.

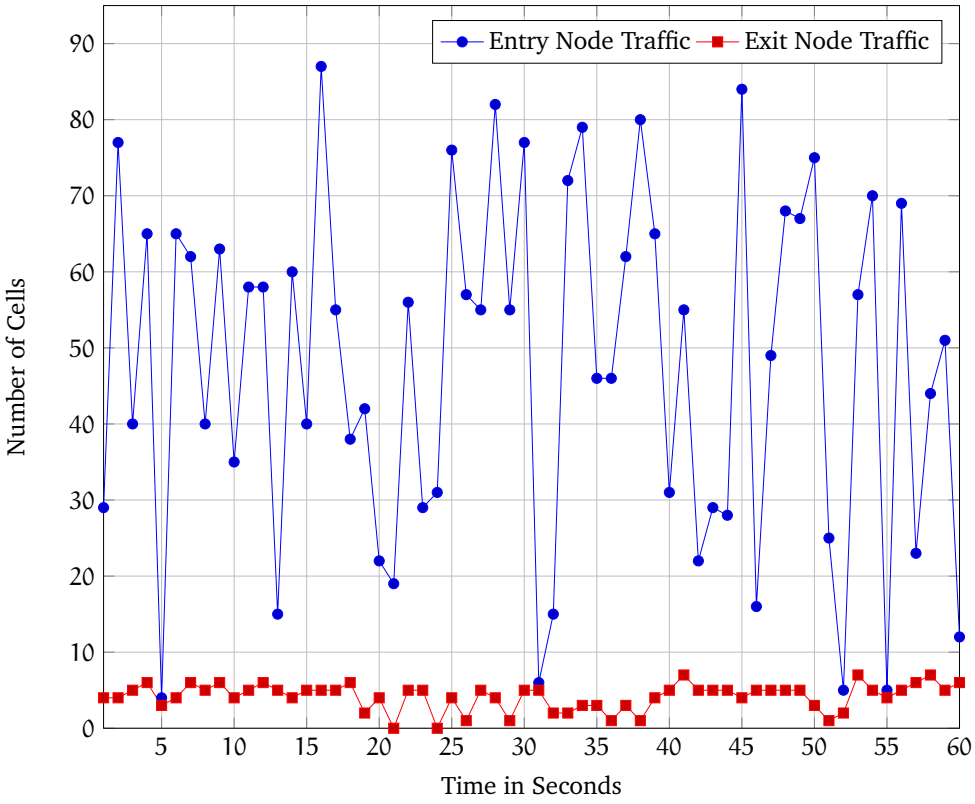


Figure 8: Session 2015-03-10 Web 1 – 60s plot of the number of received cells

Figure 8 above shows for the Web surfing session that at the entry node all non-dummy traffic is hidden inside the dummy traffic. The non-dummy traffic as recorded at the exit node never goes over 10 cells per second, whereas the number goes up to about 80 cells per second at the entry node, which includes both dummy and non-dummy cells. This demonstrates again that too much dummy cells are transmitted during Web surfing. With less dummy traffic user traffic could still be hidden inside the dummy traffic. Moreover, dummy traffic is not constant, it exhibits a more oscillating pattern.

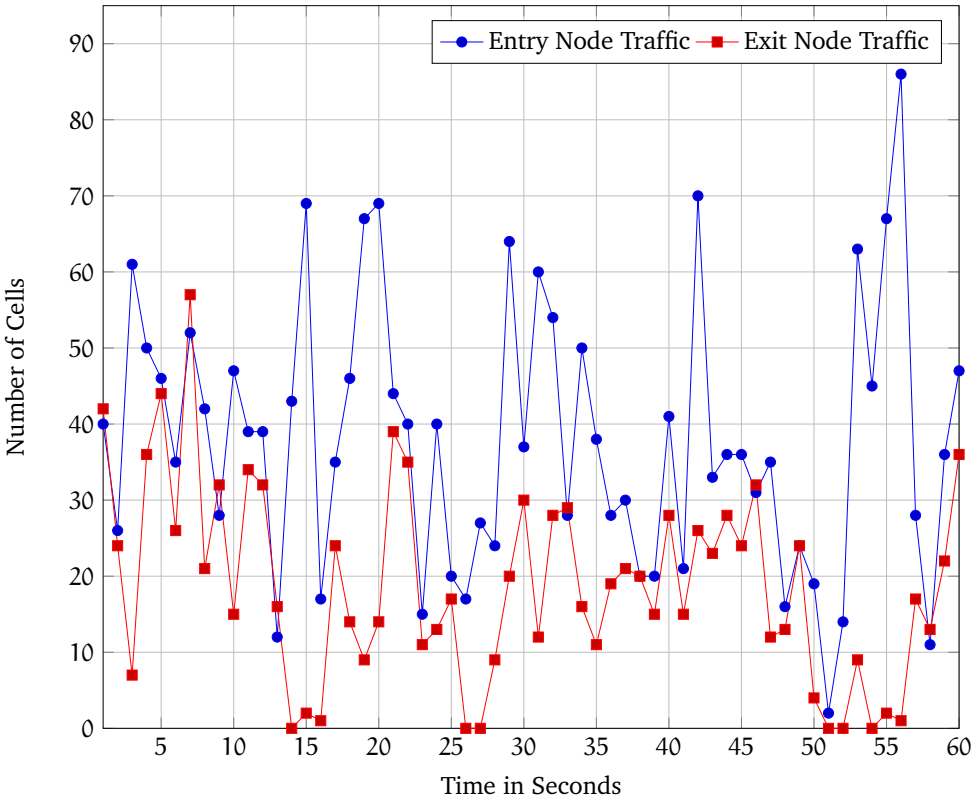


Figure 9: Session 2015-03-10 Bulk 1 – 60s plot of the number of received cells

The bulk downloading case is illustrated in Figure 9 above. It shows that the ratio of dummy cells to non-dummy cells is much lower compared to Web surfing, but user traffic is still hidden inside dummy traffic. In addition, the figure makes clear how the defence fills gaps in user traffic. For example, consider the traffic between second 15 and second 20. At the exit node the user traffic drops very much, but at the entry node the number of cells peaks instead. This demonstrates that the defence injected a lot of dummy cells during the gaps in user traffic.

Furthermore, the probability for an attacker that one correctly deanonymised a user should be calculated again using Equation 4.2 from the previous chapter. From the defence experiment the false-negative rate $FNR = 0.9$, the false-positive rate $FPR = 0.000642$ and $n = 2630$ are taken and plugged into the formula. With these numbers the attacker has only a chance of 5.593% to correctly identify a user. When n increases to 10 000, 100 000 and 1 000 000 the probability drops to 1.534%, 0.156% and 0.016%, respectively. Comparing these numbers to the attack case without enabled defence shows

that the defence reduces the attacker's chance to correctly identify a user by about a factor of 10. This demonstrates that the implemented defence can successfully defend against the implemented end-to-end confirmation attack. With a growing number of observed connections attackers have basically no idea, if their attacks were successful. They would just make to many mistakes.

5.6.3 Summary

In this chapter a defence against end-to-end confirmation attacks based on adaptive padding is proposed. The defence is implemented in Tor and its effectiveness validated both through experiments on the live Tor network and through large-scale simulations of Tor with Shadow. The results show that the defence can indeed protect against end-to-end confirmation attacks, in particular against the implemented timing attack, with a very high false-negative rate. In addition, the defence works in isolation, such that a user can utilise the defence without relying on others to use the defence as well.

These results answer research question 2: "How can such attacks be defended? How effective is the defence? Can it actually defend the attack?" – End-to-end confirmation attacks can be countered by defences based on dummy traffic. These defences can be very effective and, yes, they can actually defend against such attacks.

6 Costs of the Defence

As pointed out both in Chapter 2 and Chapter 5 it is important that a defence against end-to-end confirmation attacks does not decrease the usability of the Tor network too much. Otherwise people might stop using Tor, which not only keeps them from using the Internet anonymously but also makes the Tor network for the remaining users less secure. In the context of such a defence usability is associated with performance of the network, which in turn consists of two components. First, the performance a user perceives, i.e. how fast the network is performing from a user's perspective. Second, the network performance as a whole. If the network needs to handle more data because of the implemented defence, this puts a greater burden onto the voluntarily operated relays forming the backbone of the network and therefore increases the costs to contribute to the network by running a relay.

In order to understand the costs associated with the implemented defence the performance impact of the defence is studied in this chapter. It is investigated how the defence impacts the performance when only *one* client uses the defence and when *all* clients use it. The former is tested with experiments on the live Tor network and the latter with large-scale simulations of Tor with Shadow. Thereby performance is measured in two terms answering two different questions:

1. How does the defence impact performance from a user's perspective? How long does it take to download files over Tor with the defence enabled compared to the case without activated defence?
2. How much additional load is put onto the network by the defence? How much more bandwidth does the defence require?

The first question is answered by both the experiments and the simulations, whereas the second question can only be solved by simulations, because it requires to control a lot of clients using the defence. One client alone does not create a difference, which can be measured on the live Tor network. But in contrast, a large impact on the live Tor network by experiments should be avoided as previously described in Chapter 3. Moreover, it is important to note that the *absolute* values are not essential. It is much more crucial to identify the *relative* differences between the cases of enabled defence and deactivated defence, because only this can give conclusive answers to the question how the defence impacts Tor's performance. Together the experiments and the simulations answer research question 3.

6.1 Experiment

The experiments on the live Tor network investigate how the defence impacts the performance of Tor as a single user perceives it. Files are download through Tor and it is recorded how long it takes to download each of them. This effectively measures the decrease in performance a user experiences, when the defence is activated, and is therefore the main consequence observed by the user. In contrast, one cannot directly watch the increase in security introduced by the defence. But using Tor to surf the Internet is al-

ready slower than surfing without Tor, because the traffic is relayed over three nodes possibly distributed over multiple continents. Because of that, the defence should not increase the download time significantly in order to not drive users away from utilising Tor. With a slower performance users may be tempted to deactivate the defence, because they cannot see the benefits but only the drawbacks.

For the purpose of the experiments Torperf is utilised [52]. Torperf is a simple application requesting data via Tor and logging the current time at different steps in the process. Three different files of various sizes are repeatedly download with Torperf: 50 KiB[1], 1 MiB[2] and 5 MiB[3]. Each file is requested 500 times and 15 seconds are waited between subsequent requests. The Tor client is configured to abandon circuits after 10 seconds with the `MaxCircuitDirtiness` configuration option. This ensures that each download uses a freshly created circuit. Additionally, the use of entry guards is deactivated with `UseEntryGuards` in order to avoid that the selection of entry guards affects the results.

Three experiments were carried out on the 28. and 29. March 2015. The first experiment did not use the defence, the second experiment deployed the limited defence not padding unused circuits and the third experiment utilised the full defence also padding unused circuits.

6.2 Simulation

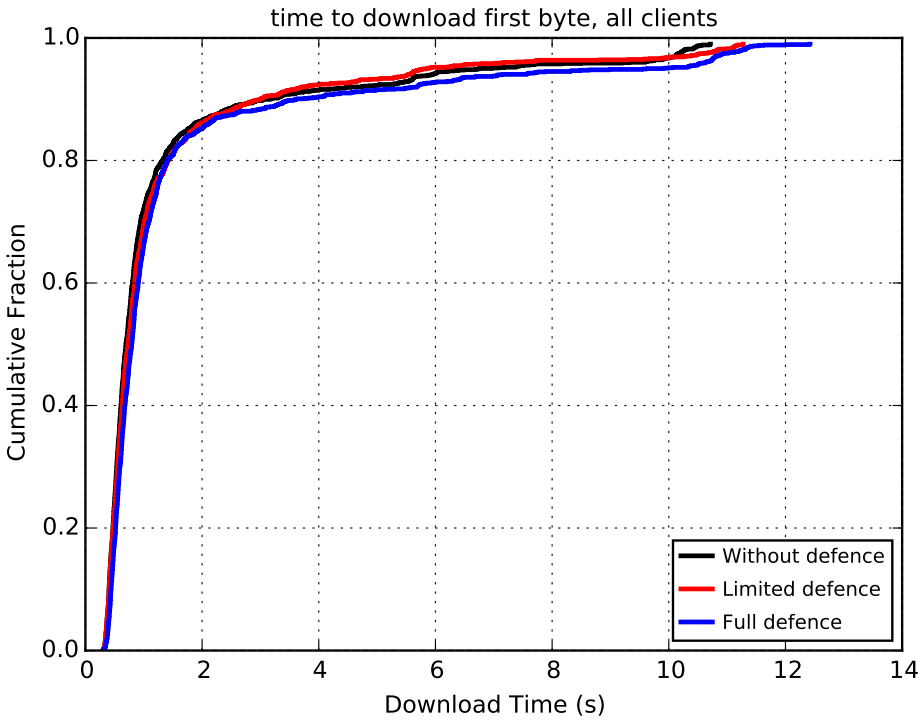
The experiments are important in order to study on the live Tor network the performance decrease caused by the defence as a user perceives it. But this cannot make any statements about the impact in the case all clients are employing the same defence simultaneously. This is achieved by large-scale simulations of the Tor network using Shadow. In addition, the simulations are able to study the performance penalties of the defence for the Tor network as a whole.

During each simulation Shadow automatically logs different kinds of performance information [53]. For each completed download it is saved inter alia how long it took to receive the file. This is used to investigate the download times during simulations. Furthermore, information is logged about how much data was transported during a simulation by each node. This is used to study the bandwidth requirements of the defence for both the whole Tor network and a single Tor node. During the simulations described in the previous chapter all these kinds of information were already logged and gathered. The information is now analysed with tools included in Shadow in order to examine the performance impact of both the limited and the full defence on the Tor network.

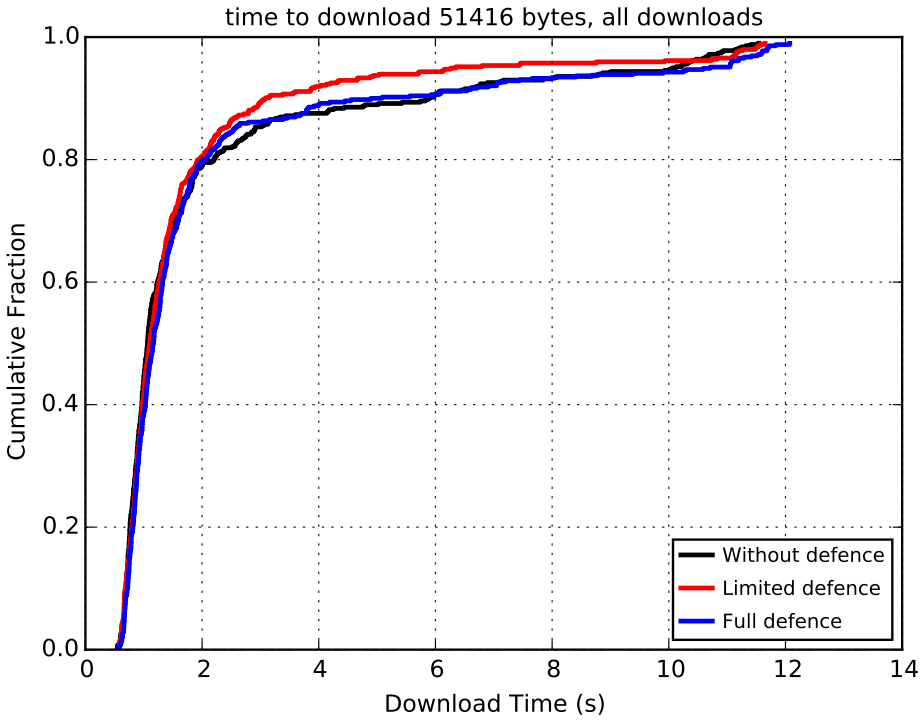
6.3 Results

Figure 10 illustrates the results from the experiments using Torperf. Figure 10a shows the time in seconds until the first byte is received for all downloads regardless of file size, whereas Figures 10b, 10c and 10d plot the times until the download of 50 KiB, 1 MiB and 5 MiB, respectively, is completed. In each case the Y-axis shows the cumulative fraction of the download times. A curve which tends towards the upper left corner means better results, i.e. more downloads were finished in shorter time.

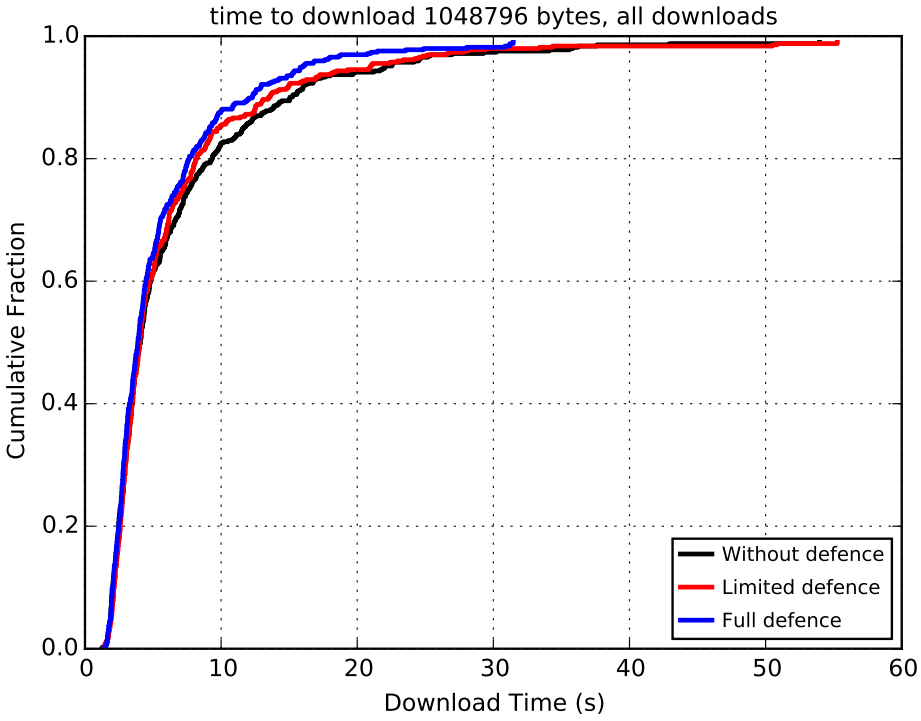
[1] `http://torperf.torproject.org/.50kbfile`
[2] `http://torperf.torproject.org/.1mbfile`
[3] `http://torperf.torproject.org/.5mbfile`



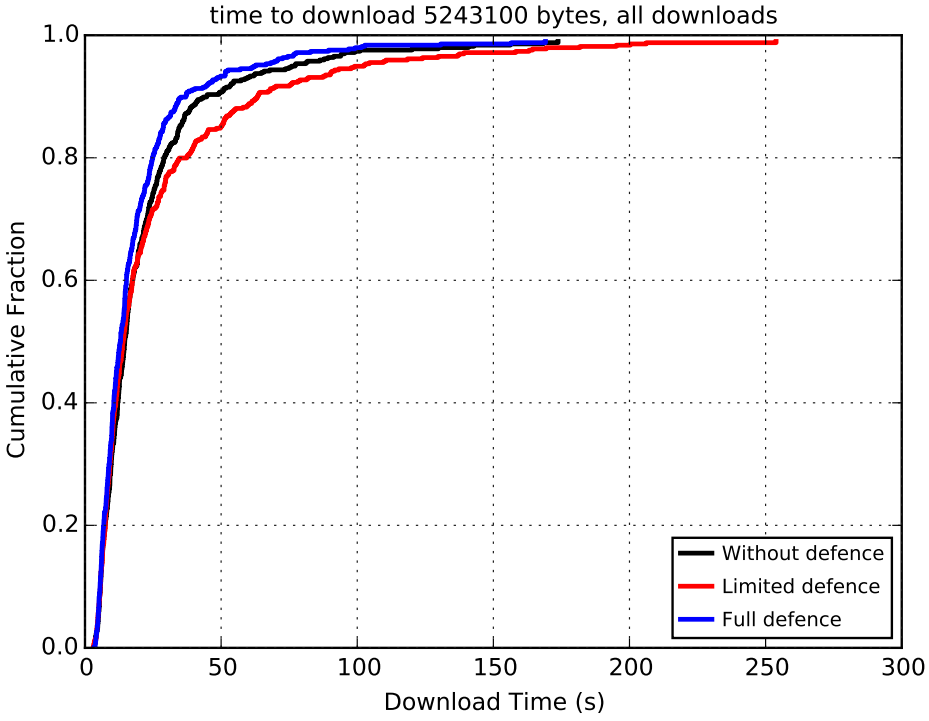
(a) Time until the first received byte from a response



(b) Time to download 50 KiB



(c) Time to download 1 MiB



(d) Time to download 5 MiB

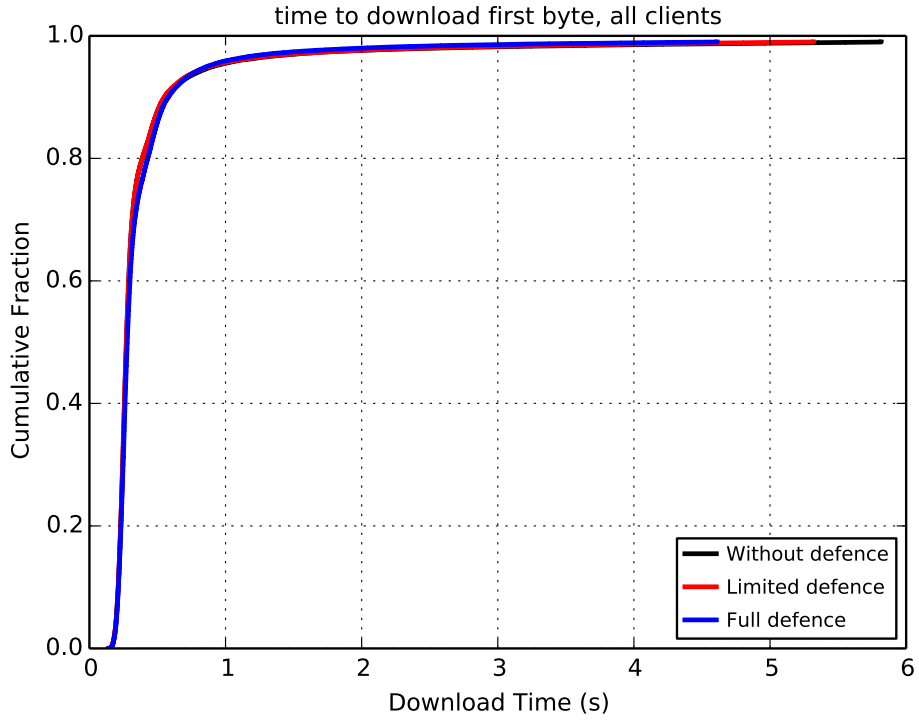
Figure 10: Time to download files of various sizes with and without enabled defence (measured with Torperf on the live Tor network)

All four graphs do not show a difference up to a fraction of 80% of all downloads for all three cases (without defence, limited defence and full defence). This means that in the vast majority of downloads there is no difference in download times observable, whether the defence is activated or not. Above 80% the graphs only show a very small difference. Interestingly, in some experiments, for instance as shown in Figure 10c for 1 MiB downloads, the performance was a little bit better with activated defence. Only in the 5 MiB download case the limited defence performed worse compared to the experiment without enabled defence. The reason for this behaviour could be that less data points were available for longer downloads as 80% of all downloads were completed during short time frames.

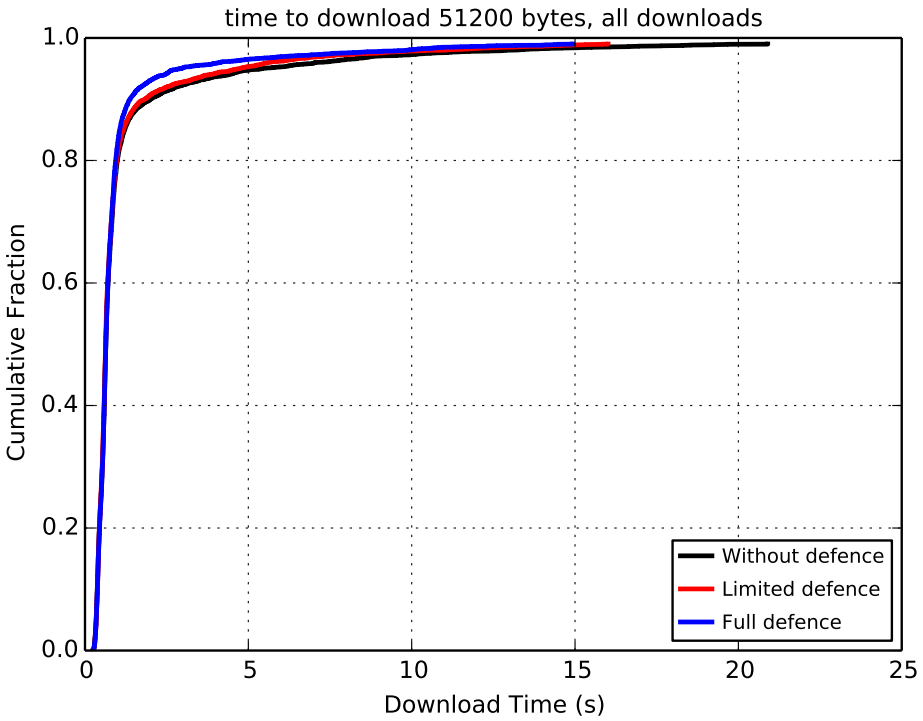
These experiments show that the implemented defence does not increase the time to download files over Tor, i.e. the performance from a user's perspective is not decreased. Furthermore, the defence does not add any extra latency. These results are reasonable, because the defence does not artificially delay real user packets. It only inserts dummy packets into traffic gaps, when no real packets are sent anyway. For those reasons the defence does not decrease the usability of Tor for a single user.

6.3.1 Simulation Download Times

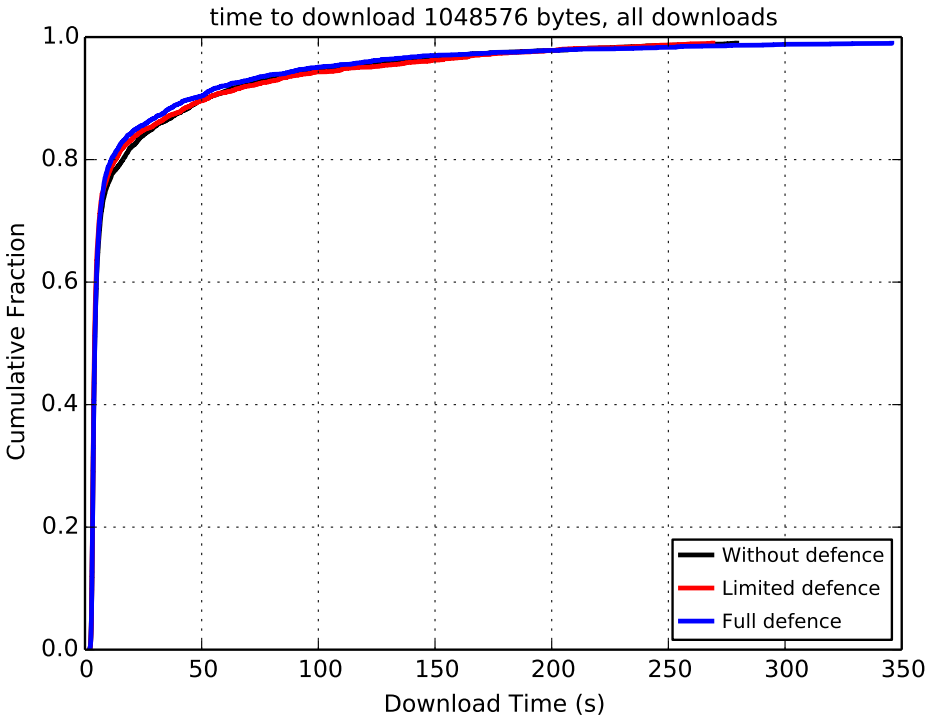
The time-to-download graphs for the simulations are depicted in Figure 11. The same graphs as before are included with one addition. The graphs for 50 KiB, 1 MiB and 5 MiB are based on traffic from the Torperf clients during the simulations, with the 5 MiB graph also including data from the bulk downloading clients. In addition, Figure 11d shows the data generated by the Web surfing clients downloading 2 MiB files.



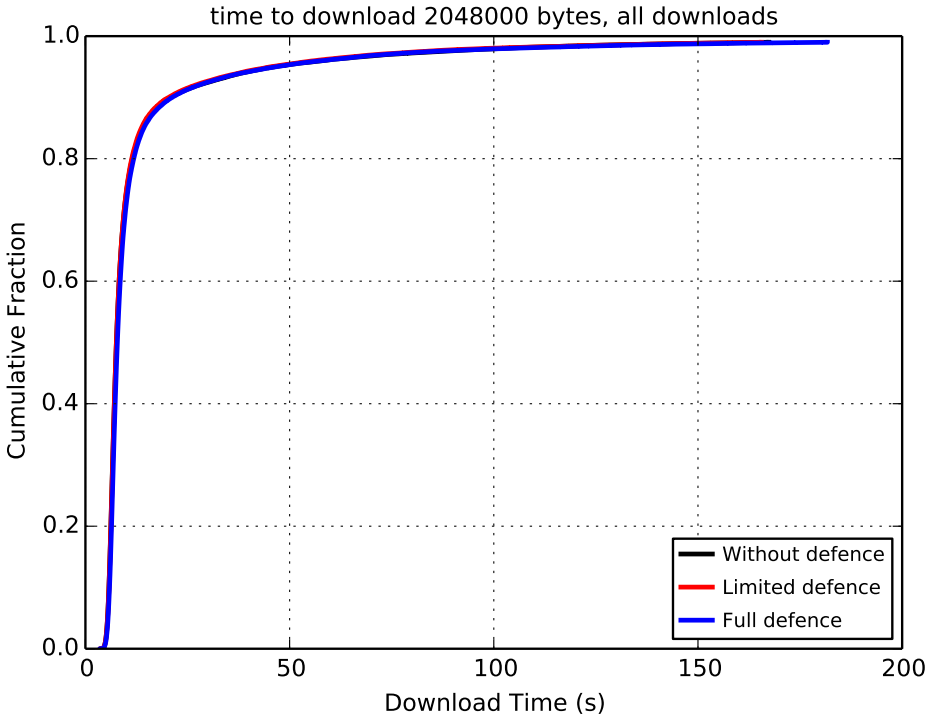
(a) Time until the first received byte from a response



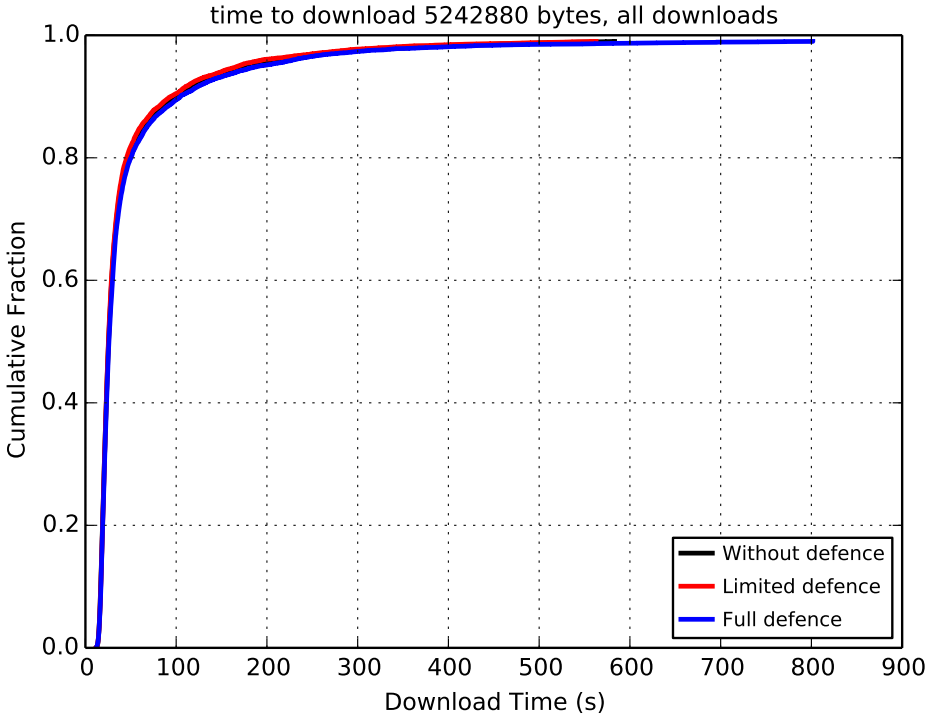
(b) Time to download 50 KiB



(c) Time to download 1 MiB



(d) Time to download 2 MiB



(e) Time to download 5 MiB

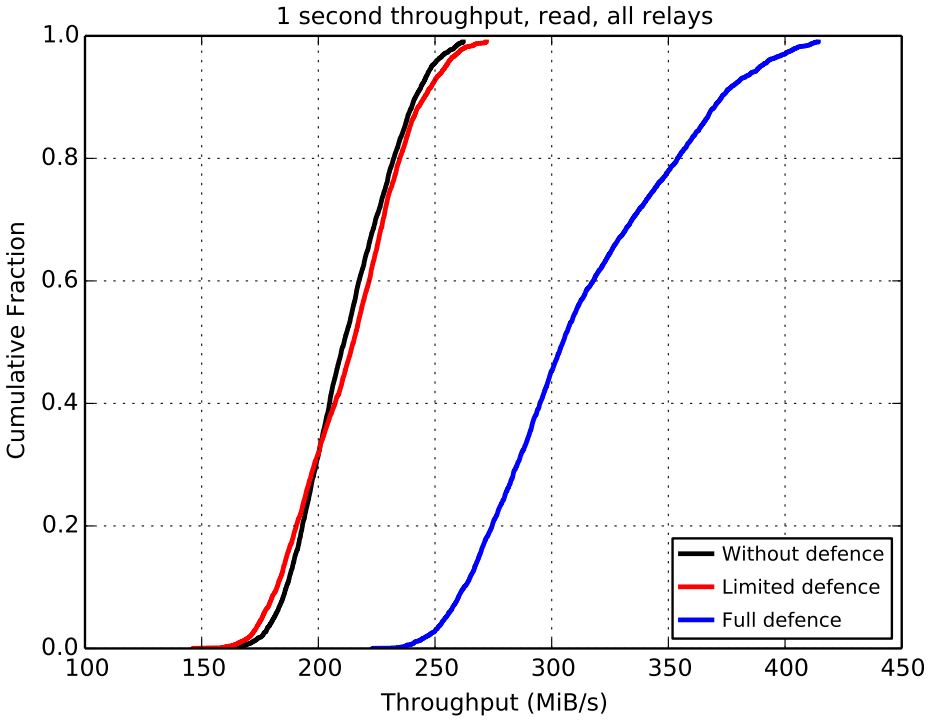
Figure 11: Time to download files of various sizes with and without enabled defence during large-scale simulations of the Tor network

Looking at the graphs confirms the results from the experiments. There is no difference in download times, neither without defence, with limited defence nor with full defence, even when all clients are using the defence. The simulation results are even more clear than the experimental results as the curves in almost all graphs nearly completely overlap. This could be attributed to the simulation environment, nevertheless, it definitely shows that the defence itself does not cause longer download times.

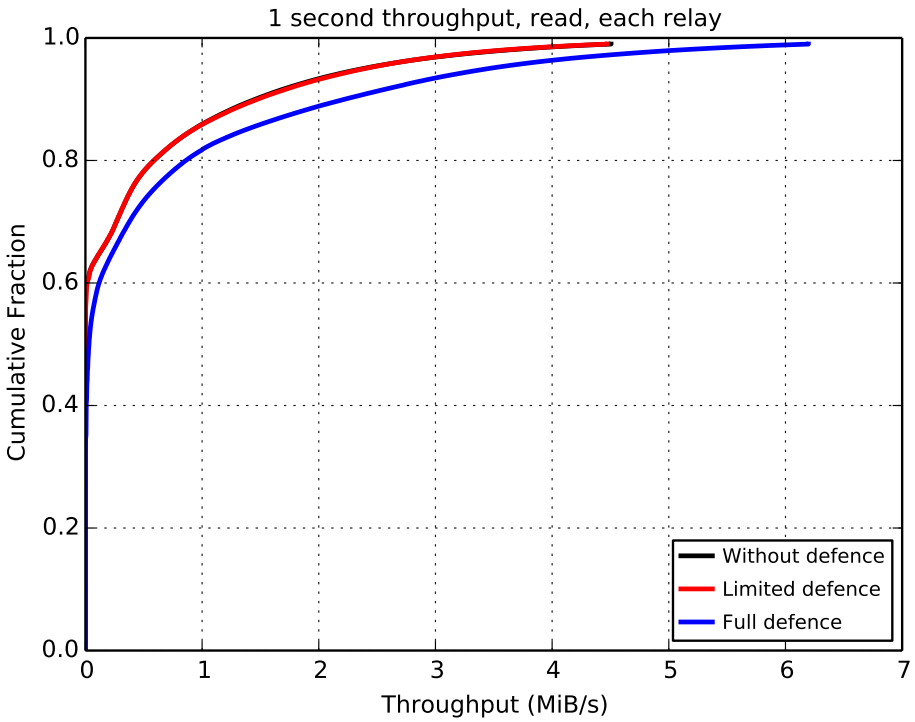
6.3.2 Simulation Throughput

Figure 12 shows how the defence impacts the load on the Tor network measured as throughput in MiB/s. A higher throughput value means that the nodes needed to handle more data every second. Figures 12a and 12b illustrate the throughput for receiving data for the whole network in total and for a single node, respectively, whereas Figures 12c and 12d illustrate the same for sending data. Again the Y-axis shows the cumulative fraction of the throughput every second, where smaller throughput values are preferred.

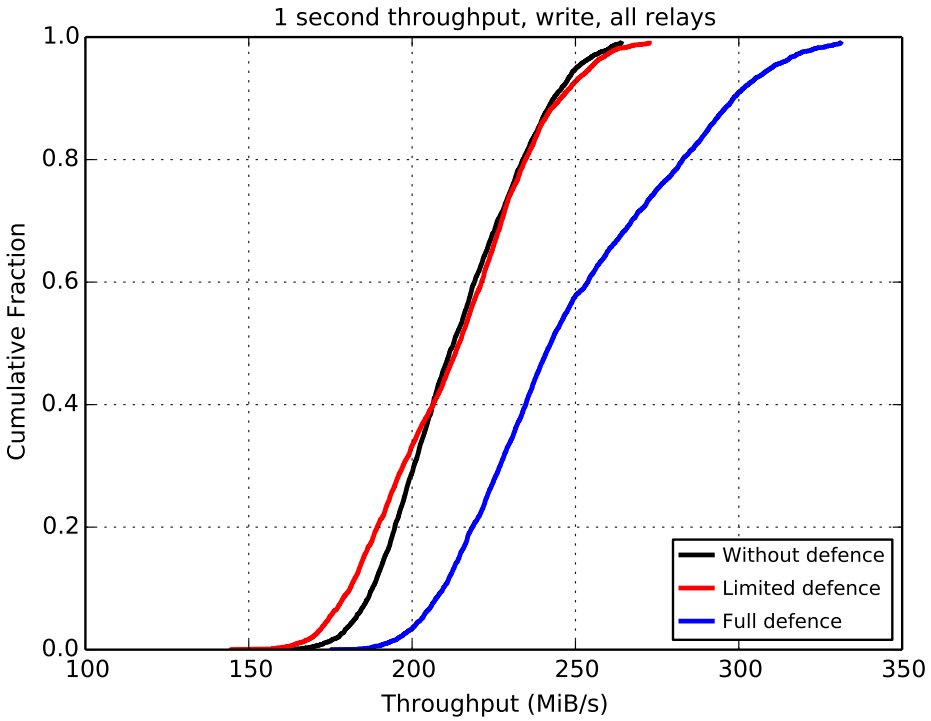
In all four cases there is no difference between the limited defence and no defence, thus, if only used circuits are padded, no significantly more load is put onto the network. However, with the full defence padding also unused circuits the network needs to handle much more data. Although each node alone only needs to process a little bit more data as shown in Figures 12b and 12d, this adds up to a lot of data for the whole network, see Figures 12a and 12c. The added load is different for receiving and sending data, because both the entry and the middle node on a circuit receive a dummy cell but only the entry node sends it forward. Because of that, the additional load is smaller for sending data than for receiving data.



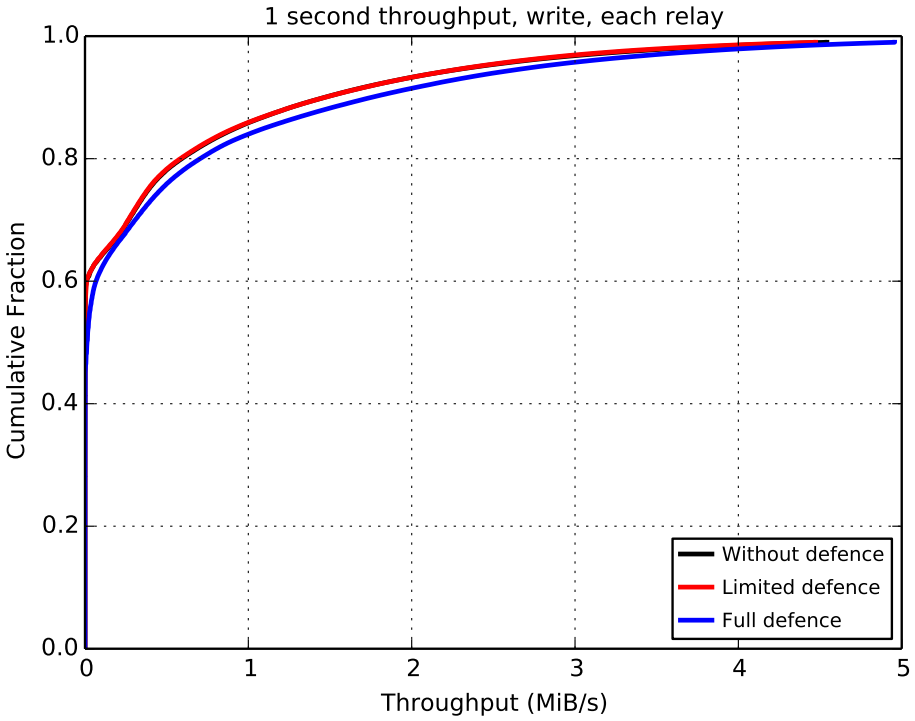
(a) Overall throughput of the network (receiving data)



(b) Throughput of a single relay (receiving data)



(c) Overall throughput of the network (sending data)



(d) Throughput of a single relay (sending data)

Figure 12: Network and relay throughput with and without enabled defence during large-scale simulations of the Tor network

The results demonstrate that the limited defence does not add significantly more load onto the network and therefore does not require relay operators to contribute more bandwidth. But the full defence adds additional load onto the network. The added load for a single node is small, but this sums up to a large value if the whole network is considered. Thus, the overall costs for operating the network is increased by the full defence. All results together also reveal that the security from the defence is gained at the expense of higher bandwidth requirements due to the added dummy traffic but not through increased latency. This confirms the theoretical considerations about the defence made in Chapter 5.

6.3.3 Summary

In this chapter the performance penalties imposed by the implemented defence against end-to-end conformation attacks are investigated through experiments on the live Tor network and through large-scale simulations of Tor using Shadow. The results show that the defence does not increase the time to download files over Tor, i.e. the usability of Tor from a user's perspective is not decreased by the defence. In contrast, the full defence adds extra load onto the network, which increases the costs to operate a Tor relay.

The results answer research question 3: "What are the costs associated with the defence? What is the decrease of performance by the usage of the defence, for a single user and for the Tor network as a whole? What additional load would be placed onto the network in terms of bandwidth usage?" – The costs associated with the defence are

increased bandwidth requirements for Tor relays. But the performance for users is not decreased by the usage of the defence. Instead only the full defence adds significantly more load onto the network.

7 Analysis of Results

The previous chapters examine the different research questions of this thesis in isolation. How effective is an end-to-end confirmation attack against Tor? How can such an attack be defended? What costs the defence? In this chapter these aspects are tied together in order to put the results of the thesis into a broader context. First, the findings of this thesis are summarised and their implications for the Tor network explained. Second, the work is discussed in a bigger context and related to other research. Third, directions where further research is needed are given.

7.1 Findings

From an attacker's perspective Chapter 4 shows that Tor is vulnerable to end-to-end confirmation attacks. It is possible with a simple method to correlate traffic at two controlled Tor nodes in order to link users to their destinations. This can be done in a very short time frame under five minutes. Carrying out such an attack is feasible with high accuracy and low false-negative and false-positive rates, thus, an attacker can on the current size Tor network deanonymise users. Because of that, end-to-end confirmation attacks are a valid threat against Tor, which needs to be considered during the further development of Tor. However, it is very important to note that this result is nothing new in particular. This thesis *confirms* that end-to-end confirmation attacks are still a threat against Tor, even as the network has grown in size significantly in the past years. This work does not present a novel previously unknown attack against Tor. *Tor is as secure as it was before!*

On one hand Tor is vulnerable to end-to-end confirmation attacks, but on the other hand attackers may not be able to draw relevant conclusions from their attacks if the number of connections they observe increases. This is especially a problem with Tor's entry guard design, because running only two out of approximately 7000 Tor nodes makes it very unlikely that attackers are able to compromise circuits, i.e. they control both the entry and the exit nodes of a circuit. They would need to control more nodes inside the network, which as demonstrated in Chapter 4 decreases the attackers' chance to indeed conclude that A communicates with B. This implies that large-scale untargeted deanonymisation of Tor users may not be possible, because attackers would just make to many mistakes. Instead attackers are more likely able to identify targeted users, i.e. they have a priori suspicion that A is communicating with B. With a targeted attack they could confirm that this communication between A and B takes place.

The main focus of this thesis is placed on defending against end-to-end confirmation attacks. The defence developed and implemented in Chapter 5 demonstrates that it is possible to defend against such attacks with a scheme based on dummy traffic. In particular it is shown that the defence can protect against the implemented timing attack. Furthermore, the defence can be implemented in Tor without large efforts. In addition, the experiments and simulations demonstrate that the defence works in isolation, i.e. single users can protect themselves alone without the need that others use the same defence simultaneously. Because of that, the defence is easy to deploy. In the forward direction only the Tor client needs to be updated, however, in the backward direction

Tor nodes must cooperate and perform the defence as well. But it is easier to gradually update 7000 nodes than several million clients. At the same time due to the fact that the defence works in isolation some users already receive protection in the early deployment phase. When more and more nodes and clients update to the defence, more and more users receive better protection as well. Because of that, gradually introducing the defence also gradually improves Tor's overall security.

It is not only important that the defence is effective in terms of protecting users, but at the same time the costs of the defence must be acceptable. Those costs are analysed in Chapter 6. It is shown through experiments and simulations that the full defence requires significantly more bandwidth from the network. But this is shared among all nodes in the network, thus, also sharing the costs among all nodes. In the last couple of years the capacity of the network in terms of available bandwidth has grown a lot as illustrated in Figure 13 below. Furthermore, the gap between available and consumed bandwidth has grown itself, such that today around half of the network's capacity is unused. In addition, the defence as implemented is not optimised. There could be better ways to generate the dummy traffic with the goal to reduce the number of transmitted dummy cells, especially in the Web surfing case as previously pointed out. Furthermore, the defence does not add any extra latency and the performance from a user's perspective is not degraded.

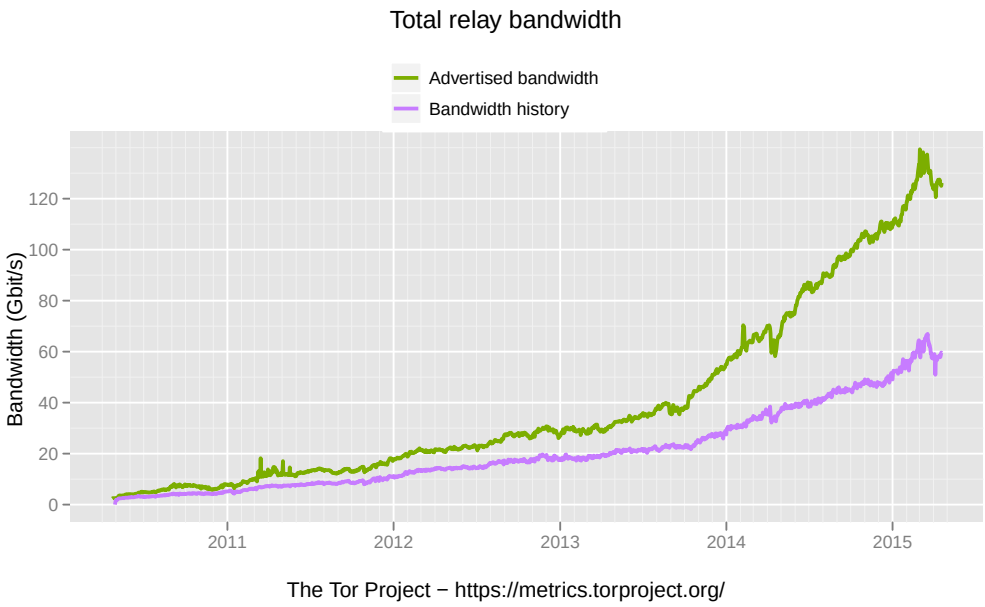


Figure 13: Development of available (green curve) and consumed (purple curve) bandwidth in the Tor network over the last five years

The above results challenge the common believe that dummy traffic just kills the network's performance and makes the network unusable for users. This assumption was true ten years ago, but today the situation is completely different. In fact, the performance for a user is not degraded by the defence. Because the Tor network today has a lot of unused bandwidth the network can tolerate additional load due to dummy traffic. For those reasons the author argues that it is worthwhile to use some of the unused bandwidth in order to improve Tor's security.

The results obtained in this thesis and the above findings help to answer research question 4: “What is the trade-off between the gained anonymity and the costs? Is the defence worth the effort or does it just slow down the network and drives users away from using Tor?” – The proposed defence trades off higher anonymity/security with increased bandwidth requirements, but it does not slow down the network from a user’s perspective. Instead the costs are shared among all node operators. Because of that, it is worth to use the defence as it makes the Tor network more secure for its users.

7.2 Discussion

This thesis shows that the implemented defence can successfully defend against the timing attack described in [5, 6]. The question remains if the same defence can also protect against other types of attacks. It can defend against the simple packet counting and start tracking attacks presented in [26], because the number of packets – cells in Tor’s context – is different at both the entry and the exit nodes due to the dummy traffic. In addition, with the full defence the start of a connection to a destination is hidden inside the dummy traffic. If the construction of a circuit is not immediately followed by the start of the connection, it is very difficult to identify the start at the entry node and to link it to an outgoing connection at the exit node. The defence can also protect against the attacks employed in [8] and [35]. The former attack is an extended version of the packet counting attack supplemented by timing information between cells. The latter attack uses statistics of transferred bytes in order to correlate traffic, which would be distorted due to the use of dummy traffic. In addition, a countermeasure against this specific attack is based on a padding scheme similar to the defence proposed in this thesis [35].

Against more sophisticated attacks, for instance as presented in [14, 28], there is not such an easy answer. At least the defence should make those attacks more difficult, because the observed patterns of traffic entering and leaving the anonymity network are very different. It is important future work to test the effectiveness of the defence against these and other attacks in order to get a deeper understand of the defence’s effectiveness against different types of attacks. In contrast, [13] shows that extended versions of the statistical disclosure attack can break protections based on dummy traffic. It would be interesting to see if this attack can also be applied against the defence presented in this thesis. The defence cannot protect against the attack demonstrated in [15], because this attack exploits Tor’s circuit building process. As the defence starts *after* a circuit is established it cannot defend against this attack.

Active Attacks

The attacks described above are mainly *passive* end-to-end confirmation attacks, i.e. an attacker only records traffic in order to correlate it afterwards. In principle, the defence is only designed to protect against such passive attacks. Nevertheless, it is important to also consider *active* end-to-end confirmation attacks, which actively manipulate traffic with the goal to easier link users and destinations together. For this purpose an attacker can drop packets or insert artificial delays between packets at one node in order to subsequently detect this manipulation at another node. For example, consider the attack described in [29]. This attack inserts a *fingerprint* into traffic by encoding the fingerprint as delays between packets and recovers it from the manipulated traffic at the receiving node. In the forward direction, the defence can defend against this attack, because the

malicious entry node would modulate the traffic including the client-generated dummy cells. But the middle node drops the dummy cells, such that the fingerprint would be destroyed at the exit node. In the backward direction, an honest middle node would insert dummy cells into the artificial delays caused by the traffic modulation of the malicious exit node. Again, the fingerprint would be destroyed at the entry node.

A similar active end-to-end confirmation attack against Tor is described in [33]. In this attack a binary “0” is encoded as a single cell and a binary “1” as a burst of three consecutive cells forwarded directly after another. A binary *signal*, also referred to as a *watermark*, is inserted into traffic in this way by one node and recovered by another in order to confirm that a particular entry connection corresponds to a specific exit connection. The same considerations as before apply in this case as well. Removing or inserting dummy cells could destroy the watermark. But encoding the watermark only needs a couple of cells, such that it would be possible that the signal could be inserted into a burst of real user traffic without dummy traffic in between. Because of that, it needs to be empirically verified if the defence actually works against this attack or if the attacker would be able to reliably insert the signal into traffic.

An interesting approach to counter active timing attacks is given in [30]. This work takes *any* padding scheme, which is effective against passive timing attacks such as the defence proposed in this thesis, as a basic building block and creates a protection against active timing attacks based on the chosen scheme. Thus, any effective defence against passive timing attacks can be transformed into a defence against active timing attacks. The idea is to send multiple copies of the same packet via different paths through the anonymity network. Each packet includes a timestamp at which the packet needs to be forwarded by the next node on a path. If the attacker does not control more than half of all nodes inside the network, this defence works even when packets are delayed or dropped, because packets not manipulated by the attacker will reach the destination in time via an uncontrolled path. But implementing this defence in Tor is not trivial, because sending the same cells over multiple circuits is currently not supported by Tor.

More active end-to-end confirmation attacks exploiting protocol specific properties of Tor are introduced in [34] and [32]. The former attack sends exactly 50 dummy cells from a controlled node to a hidden service. If the attacker also operates the entry node of the hidden service, recognising the exact number of forwarded cells – the 50 dummy cells plus cells for circuit establishment – reveals the IP address of the hidden service to the attacker. Utilising the proposed defence will thwart this attack, because the signal of dummy cells is not a reliable signal any longer as the number of sent cells varies due to the dummy traffic generated by the defence. The latter attack manipulates a single cell (add, delete, modify or replay) at the entry node and detects this manipulation at the exit node, because it causes a protocol error at the exit node. This protocol manipulation cannot be prevented by the proposed defence.

Attack the Defence

The previous paragraphs discuss if and how the proposed defence can protect against different kinds of passive and active end-to-end confirmation attacks. It is argued that this defence can counter various attacks. But are there ways to attack the defence itself? Is there an easy method to circumvent the defence? If an attacker is able to filter out the generated dummy traffic at the entry node, the defence is useless, because in this

case the attacker can perform the attack against the user traffic without considering the dummy traffic. On the Tor protocol-level this is not possible, because an attacker cannot recognise long-range `drop` relay cells due to Tor's layered encryption. But exists another method to distinguish between dummy traffic and real user traffic?

An attacker might want to look at the density of traffic. Maybe dummy and user traffic have a different density. Here *traffic density* is defined as "number of packets per time frame", for example the number of cells per second. But this definition basically captures what the implemented timing attack is doing: counting cells per second. Besides this there is an additional problem. How does the attacker know that a particular density is typical for dummy traffic or real traffic? For real traffic one can calculate the density at the exit node, because the traffic at the exit node does not include dummy cells. But at the entry node dummy and user traffic is mixed together during the same time frame. How does the attacker distinguish between real and dummy traffic in this case? The attacker has only one easy chance, when there is no traffic at the exit node. In this case all traffic at the entry node must be dummy traffic. But in addition to this the attacker faces another obstacle. Traffic density does not stay constant. User traffic changes based on the user's behaviour and dummy traffic changes based on user traffic plus randomness included in the defence. For all those reasons the author argues that there is no simple way to distinguish between dummy and user traffic based on traffic density in order to filter out the dummy traffic. Every way to do it would be a more sophisticated end-to-end confirmation attack on its own.

Instead of looking at traffic density an attacker might want to try another trick to circumvent the defence. The defence, i.e. the method to generate dummy traffic, is known to the attacker. One can apply the same algorithm to the unpadded traffic at the exit node and then correlate the padded entry traffic and the padded exit traffic. How good is the correlation in this case? Can the attacker successfully correlate the traffic again? This would decrease the false-negative rate of the attack. But by applying the same padding scheme to the exit traffic the attacker faces a problem. The defence contains randomness. Because of that, the attacker cannot repeat the algorithm deterministically. As a consequence of this an implementation of the defence should use a strong random number generator. Another issue arises as the defence removes distinct gaps in traffic. Thus, repeating the padding at the exit node could increase the false-positive rate of the attack, because all the padded entry and exit traffic should look more similar. From the defender's perspective the question is: Does this higher false-positive rate compensate the potential lower false-negative rate?

In order to study the feasibility of this *repeat-padding attack* it is implemented in the analysis tool previously used to correlate traffic. Thereby the same padding scheme as employed by the defence is applied to the recorded exit traffic. The time between recorded cells is used to regenerate the distribution of traffic for each exit connection. Using this distribution dummy cells are introduced in gaps between cells. The modified analysis tool was executed against the recorded traffic from the defence experiments, which resulted in an overall false-positive rate of 0.000387 (previously 0.000642) and a false-negative rate of 0.85 (previously 0.9). The repeated-padding attack was only able to identify one more connection compared to the standard attack. The false-positive rate decreases, because during the experiments only the controlled client used the defence, such that the correlation between unpadded entry traffic from other clients with the

padded exit traffic is reduced. Looking at the correlation between traffic from the controlled client at both nodes reveals that in some cases the correlation increases but in other cases the correlation actually decreases. The average correlation coefficient for the bulk downloading sessions stays nearly constant with $r = 0.43$ (previously $r = 0.445$), but the correlation coefficient for the Web surfing sessions is reduced to $r = -0.425$ (previously $r = -0.254$). Repeating this attack with the simulation data confirms the experimental results. Especially, the false-positive rate does not increase, when all clients use the defence.

These results show that the repeat-padding attack cannot be easily carried out. This can be explained in the following way. The recoding of traffic started when communication between user and destination already took place, such that the client also already adapted the distribution of traffic. But with the repeat-padding attack the attack had to regenerate the distribution from scratch. This means that during the recording period the distribution used by the client to generate dummies was different than the distribution used during the repeat-padding attack. For this reason the dummy traffic introduced for exit traffic is significantly different than the actual dummy traffic at the entry node. However, if the repeat-padding attack would be implemented directly at the exit node, this would not be an issue as the attacker can start adapting the distribution once a circuit is established. Then both client and attacker would use the same distribution to generate dummy cells. This considerations lead to the following idea. Currently, the distribution is initialised uniformly with a constant value. This initialisation could be performed based on a random number selected for each circuit independently. This number would never leave the client and an attacker cannot guess it, such that the repeat-padding attack would need to use a different initialisation again. But this also means that the initialisation is not uniform any longer, however, as the distribution is adapted the initialisation plays a smaller and smaller role over time. In addition, the problem of non-determinism in the defence still remains for an attacker.

Relay-level vs. Network-level Adversaries

In order to carry out a successful end-to-end confirmation attack two steps are necessary. First, the attacker must come into the position to correlate traffic. Second, one must correlate the traffic. This thesis mainly investigates the second step. How can an attacker link a user to the destination? But the first step is equally important, because if an attacker is not in the position to conduct the correlation, a perfect method for the second step is useless. The first step must be considered under the light of two different attackers: *relay-level* and *network-level* adversaries. A relay-level adversary controls Tor nodes inside the network by either running own nodes or by compromising existing nodes. This is the experimental set-up used in this thesis. But a network-level adversary can perform the same end-to-end confirmation attack, if one is able to watch the connection between the client and the entry node as well as the connection between the exit node and the destination. Such an attacker could be an Internet service provider, an Internet exchange point [14] or the operator of an autonomous system [54], basically every entity who can observe a large fraction of the Internet including parts of the Tor network.

Tor's entry guard design tries to make the first step harder for an attacker, whereas currently no defence against the second step is implemented in Tor. In order to strengthen Tor's security relying only on one single defence may not be enough. In fact, the entry

guard design and the proposed defence can complement each other. With entry guards an attacker must observe a larger part of the network in order to increase the chance to see both entry and exit traffic of one user. But at the same time this decreases the confidence that one actually deanonymised the right user using an end-to-end confirmation attack. Thus, combining both defences makes the attack even more difficult. The entry guard design works reasonable well against a relay-level attacker, especially when the recently discussed improvements are integrated into Tor [25], which try to fix some of the weaknesses of entry guards [10]. But the situation is different when network-level attackers are considered. Even if the entry guard is fixed, the underlying network connections between the client and the same guard could change due to changes in the network topology or due to active attacks with the goal to observe more Tor traffic [55]. Because of that, the same protection of entry guards against relay-level attackers does not necessarily apply to network-level attackers as well.

But can network-level adversaries identify Tor traffic? The authors of [13] write in their paper: *"The attacks in this paper fail to work when: [...] The attacker cannot tell when Alice is originating messages."* In addition, the authors of [14] write: *"We also assume that the adversary can distinguish Tor traffic from other traffic, which may trivially done by IP address and port number, based on information in the Tor directory."* Thus, if attackers cannot distinguish between Tor and non-Tor traffic, they may not be able to carry out the attack even when they would be in the position to observe both ends of the communication. Because of that, another layer of protection against network-level adversaries could be the disguise of Tor traffic. This is an area actively researched in the context of censorship circumvention, which has the goal to stop an attacker of blocking Tor traffic [56]. Tor can make use of *pluggable transports* [57], which transform Tor traffic between the client and the first node, such that an attacker using deep packet inspection cannot identify Tor traffic and block it. The same pluggable transports can also be utilised to hide Tor traffic under the observation of a network-level adversary. But this does not work against relay-level attackers, because they can see the plain Tor traffic at the nodes they control.

Honest vs. Dishonest Nodes

The proposed defence relies on honest nodes, which must generate the dummy traffic in the backward direction. This is not a problem as long as at least one node is honest. Even with two nodes being malicious an honest third node alone would generate enough dummy traffic similar to the client in the forward direction. But it would be desirable to identify the dishonest nodes. This can be achieved, because clients see the dummy cells sent to them and can recognise the sender. Based on this the client can determine if a node is sending dummy traffic. This can be used to spot dishonest nodes and to change circuits with malicious nodes.

7.3 Future Work

As previously mentioned one very important direction for future work is studying the effectiveness of the defence against other kinds of attacks. Furthermore, this thesis focuses on the forward direction. But the implemented timing attack can be further enhanced by also correlating traffic in the backward direction as already pointed out in Chapter 4. In addition to this, the defence needs to be implemented and tested in the backward di-

rection, too. Because the design of the defence already considers the backward direction this should be straightforward.

Future work should also test the start tracking attack against the current Tor network without implemented defence. If it turns out that the attack does not work against Tor, padding of unused circuits would not be necessary. This means that deploying the limited defence padding only active circuits would be enough. In this case substantial less bandwidth would be required as shown by the simulations in Chapter 6.

Improve the Defence

A better strategy for generating dummy packets as part of the defence should be possible. Currently, the defence uses the strategy from the original paper proposing adaptive padding. But this strategy could be tweaked in different ways, for instance the number of bins, the separation between low and high bins, or using another distribution for the bin intervals than the exponential distribution. The goal should be to send less dummy traffic but still destroying the correlation at the entry and exit nodes. The analysis of the defence in Chapter 5 shows that in the Web surfing cases more dummy traffic than actually necessary is generated. The reason for this is that the defence sends to much dummy cells during idle times when no real user traffic is transported. Sending less dummy cells during these periods would also reduce the load on the network with the full defence as unused circuits would transmit less dummy cells as well. One simple solution might be swapping the sampling from the low and high bins. Instead of sampling from the high bins after a real user cell and sampling from low bins after a dummy cell the opposite may provide some benefits. If the interval after a real user cell would be sampled from the low bins, this would increase the chance to introduce a dummy cell. This could help to decrease the correlation within bulk downloads as more dummy cells should be sent. In addition, sampling from the high bins after a dummy cell would decrease the number of dummy cells during idle times as less dummy cells would be placed between gaps in traffic.

Preliminary tests with three Web surfing and three bulk downloading sessions confirm that the approach of swapping the sampling from low and high bins has the desired effect. During the Web surfing sessions less dummy cells are generated compared to the original sampling strategy, whereas during bulk downloads more dummy cells are sent. But at the same time this change caused a higher correlation coefficient for traffic from the controlled client. Especially, in all three bulk downloading cases the attack could successfully identify the user with a very high correlation coefficient of $r > 0.9$, thus, with the changed sampling algorithm the defence could not hide the correlation between traffic. This shows that every change to the defence needs to be carefully studied in order find the best balance between efficiency and anonymity.

Network-level Adversaries

The experimental set-up in this thesis simulated a relay-level attacker. This makes the attack generally easier compared to a network-level attacker, because the attack is independent from the underlying network. It is easy to count fixed-size cells and there is a one-to-one correspondence between cells at the entry and the exit nodes. If a cell is sent to the exit node, there will only be one corresponding cell at the entry node. In addition, using TLS connections between nodes as well as clients and nodes guarantees that cells are not lost. In contrast, a network-level adversary has to face several challenges due to

network conditions.

Assume that such an attacker counts IP packets between the client and the entry node as well as the exit node and the final destination and correlates these numbers with the timing attack employed in this thesis. First of all, one cell does not equal one IP packet as several cells could be transported inside one IP packet. At the same time an IP packet travelling from the client to the entry node contains a TCP segment containing a TLS connection containing Tor cells containing data send to the destination. An IP packet between the exit node and the destination contains only a TCP segment containing data send to the destination. Thus, one IP packet on the entry connection cannot necessarily be mapped to a corresponding packet on the exit connection, because not all cells contain data transported to the destination and the sizes of IP packets and TCP segments could be different at both ends of the communication. Additionally, the packet count could be different, because IP packets may be fragmented into several smaller IP packets due to the physical network. Packet drops on the physical network link also requires retransmission of packets. One further problem is that circuits are multiplexed over a single TLS connection. Thus, circuits sending data to different destinations from the same client enter the network via one TLS connection, but leave the network via multiple exit connections. Network observers cannot distinguish between different circuits inside one TLS connection as they cannot look into this connection.

The point here is that an end-to-end confirmation attack might not be that easy and straightforward for a network-level adversary than for a relay-level adversary, however, they are possible as shown by [14, 35]. Future work should explore the effectiveness of end-to-end confirmation attacks for a network-level adversary and how good the proposed defence protects against such an attacker. Another related and important question is the problem of large-scale deanonymisation. Can a state-level attacker who observes say 30% of the network carry out large-scale deanonymisation of a large number of users using end-to-end confirmation attacks? This question could be answered with simulations utilising Shadow.

Tor Hidden Services

This thesis looks at the “standard” use case of Tor, where for example a user wants to access a public Web site on the Internet anonymously. But Tor can also provide anonymity to service operators with Tor hidden services [16]. With hidden services an attacker cannot find out the location, i.e. the IP address, of such a service. The defence should be able to provide additional security to hidden services as well, for instance by defending against the end-to-end confirmation attacks described in [8, 34], which try to identify hidden services. The attack scenario is different depending on the attacker’s goal – identifying the user or the service –, but the basic idea of correlating traffic is the same, such that the defence can protect against these attacks. But are there additional benefits compared to the standard use case? Are there any drawbacks or ways to attack or circumvent the defence in the special hidden services case? What is the threat model with hidden services? Does an attacker want to find out what hidden service a user accessed or does one want to deanonymise the hidden service? In which cases can the defence protect the user or the service operator, in which not? In the context of hidden services a client can send long-range padding to the service and vice versa, because both parties establish an end-to-end encrypted anonymous channel during the connection set-up between

the client and the service [58]. Can this be used to provide a better padding scheme for hidden services, possible replacing the proposed defence? Against what types of attacks could such a padding scheme protect? In summary, hidden services are a special case in the Tor network, which requires special considerations as well, although the proposed defence already provides hidden services with more security, too.

Web Site Fingerprinting Attacks

A similar problem related to end-to-end confirmation attacks are *Web site fingerprinting attacks* [59]. Such an attack does not try to correlate the traffic at both ends of the communication, instead it only observes the traffic patterns at the entry connection between client and entry node. The attack compares the observed patterns to previously classified traffic patterns of Web sites using some machine learning algorithm in order to find out which Web site the user accessed. For example, this can be useful for an attacker who wants to know if a user accessed a forbidden or censored Web site. Because of that, Web site fingerprinting is only an one-sided confirmation attack, which in principle is easier to carry out as only one connection needs to be monitored. Every Internet service provider can perform this attack against its customers, for instance.

Because both attacks are strongly related a defence against one attack could also protect against the other attack. Therefore, it would be interesting future work to apply the proposed defence to Web site fingerprinting attacks and to evaluate its effectiveness. Can the dummy traffic introduced by the defence confuse the attacker in order to cause a misclassification? Or does the padding increase the false-positive rate as the traffic to different Web sites looks more similar? In fact, [60] proposes to use adaptive padding as a defence against fingerprinting attacks and suggests that the same defence could also protect against end-to-end confirmation attacks. Thus, in the other way round the proposed defence could be deployed against Web site fingerprinting attacks as well.

Integration into Tor Browser

The newest generation of the Tor Browser, which is a modified Firefox browser combined with Tor and designed to fix several application-level anonymity issues while surfing the Web, contains a feature called the “Security Slider” [61]. With the Security Slider users can decide between usability and anonymity based on their own security requirements by choosing an appropriate security level. On higher levels more features shown in the past to be prone to vulnerabilities are deactivated. The proposed defence could be integrated into the Security Slider, such that only users who require higher security could use the defence. This is possible because the defence works in isolation and it could reduce the load on the network as not all users would activate the defence.

The implementation in the forward direction would be trivial, because sending dummy traffic in this direction is controlled by the client. In the backward direction, the implementation would be more difficult as the Tor client must inform the nodes on a circuit to activate the defence. This requires a protocol change and could be achieved in two ways, either by defining a new handshake between a client and a node, which would contain an option specifying that the node should generate dummy traffic, or by specifying a new relay cell with the same option sent by the client to the node. See [23] for details about Tor’s protocol.

8 Conclusion

End-to-end confirmation attacks are a well-known problem in the context of anonymity networks. In fact, most researchers assume that such an attack will generally be successful if the attacker comes into the position to observe both ends of the communication, i.e. traffic entering and leaving an anonymity network. This thesis confirms this assumption for the Tor network. Despite the fact that the Tor network has grown a lot in the last couple of years end-to-end confirmation attacks are still a valid and serious threat. However, no defence directly protecting against end-to-end confirmation attacks is currently implemented in Tor. This is reflected in the following quote from the original Tor design paper from August 2004:

> *"Tradeoffs between padding protection and cost were discussed, and traffic shaping algorithms were theorized to provide good security without expensive padding, but no concrete padding scheme was suggested. Recent research and deployment experience suggest that this level of resource use is not practical or economical; and even full link padding is still vulnerable. Thus, until we have a proven and convenient design for traffic shaping or low-latency mixing that improves anonymity against a realistic adversary, we leave these strategies out."* [2, August 2004]

Especially, the last sentence from the quote still holds today – over a decade after the initial design of Tor. In February 2015 Roger Dingledine, one of the Tor inventors and still a leading person in the Tor community, replies to a question on the tor-talk mailing list about the necessity of very expensive defences to *"fix anonymity"* [62]. He answers the following:

> *"It's actually worse than that – we have no idea.*
>
> *I'd love to have a graph where the x axis is how much additional overhead (latency, bandwidth, whatever) we're willing to add, and the y axis is how much additional security (anonymity, privacy, whatever) we can get.*
>
> *Currently we have zero data points for this graph."* [62, February 2015]

This shows that despite over ten years of Tor research the problem of end-to-end confirmation attacks is not solved at all. In fact, to the best knowledge of the author this thesis is the first work to propose, implement and evaluate a general defence directly defending against end-to-end confirmation attacks in the Tor network. This defence can protect against several attacks, is simple, easy to implement and deploy as well as usable. Hopefully this work will stimulate further research into the topic of end-to-end confirmation attacks in order to improve the proposed defence and ultimately to provide better security and anonymity to the users of Tor.

Bibliography

[1] Moglen, E. Privacy under attack: the NSA files revealed new threats to democracy (online). May 2014. URL: http://www.theguardian.com/p/3pfn5/sbl (Visited 2015-05-18).

[2] Dingledine, R., Mathewson, N., & Syverson, P. August 2004. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*.

[3] Acquisti, A., Dingledine, R., & Syverson, P. January 2003. On the Economics of Anonymity. In *Proceedings of Financial Cryptography (FC 2003)*, Wright, R. N., ed, volume 2742 of *Lecture Notes in Computer Science*. Springer-Verlag.

[4] Dingledine, R. & Mathewson, N. June 2006. Anonymity Loves Company: Usability and the Network Effect. In *Proceedings of the Fifth Workshop on the Economics of Information Security (WEIS 2006)*, Anderson, R., ed.

[5] Levine, B. N., Reiter, M. K., Wang, C., & Wright, M. February 2004. Timing Attacks in Low-Latency Mix-Based Systems. In *Proceedings of Financial Cryptography (FC 2004)*, Juels, A., ed, volume 3110 of *Lecture Notes in Computer Science*, 251–265. Springer-Verlag.

[6] Shmatikov, V. & Wang, M.-H. September 2006. Timing Analysis in Low-Latency Mix Networks: Attacks and Defenses. In *Proceedings of ESORICS 2006*.

[7] Wang, W., Motani, M., & Srinivasan, V. October 2008. Dependent Link Padding Algorithms for Low Latency Anonymity Systems. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS 2008)*, 323–332. ACM Press. doi:10.1145/1455770.1455812.

[8] Øverlier, L. & Syverson, P. May 2006. Locating Hidden Servers. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*. IEEE CS.

[9] Dingledine, R. The lifecycle of a new relay (online). September 2013. URL: https://blog.torproject.org/blog/lifecycle-of-a-new-relay (Visited 2015-01-13).

[10] Dingledine, R. Improving Tor's anonymity by changing guard parameters (online). October 2013. URL: https://blog.torproject.org/blog/improving-tors-anonymity-changing-guard-parameters (Visited 2015-01-13).

[11] Johnson, A., Wacek, C., Jansen, R., Sherr, M., & Syverson, P. November 2013. Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries. In *Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS 2013)*. ACM Press.

[12] AlSabah, M. & Goldberg, I. Performance and Security Improvements for Tor: A Survey. Report 2015/235, Cryptology ePrint Archive, March 2015. URL: https://eprint.iacr.org/2015/235.

[13] Mathewson, N. & Dingledine, R. May 2004. Practical Traffic Analysis: Extending and Resisting Statistical Disclosure. In *Proceedings of the Fourth Workshop on Privacy Enhancing Technologies (PET 2004)*, volume 3424 of *Lecture Notes in Computer Science*, 17–34.

[14] Murdoch, S. J. & Zieliński, P. June 2007. Sampled Traffic Analysis by Internet-Exchange-Level Adversaries. In *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)*, Borisov, N. & Golle, P., eds. Springer-Verlag.

[15] Bauer, K., McCoy, D., Grunwald, D., Kohno, T., & Sicker, D. October 2007. Low-Resource Routing Attacks Against Tor. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2007)*.

[16] Müller, K. Past, Present and Future of Tor Hidden Services. Høgskolen i Gjøviks rapportserie 01/2015, January 2015. URL: http://hdl.handle.net/11250/274863.

[17] Gollmann, D. 2011. *Computer Security*. John Wiley & Sons, 3rd edition.

[18] Chaum, D. February 1981. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2).

[19] Reiter, M. & Rubin, A. June 1998. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security*, 1(1).

[20] Leech, M., Ganis, M., Lee, Y.-D., Kuris, R., Koblas, D., & Jones, L. March 1996. SOCKS Protocol Version 5. RFC 1928 (Proposed Standard). URL: https://www.ietf.org/rfc/rfc1928.

[21] The Tor Project. Tor directory protocol, version 3 (online). URL: https://gitweb.torproject.org/torspec.git/blob/HEAD:/dir-spec.txt (Visited 2015-01-14).

[22] Dingledine, R. & Mathewson, N. Tor Path Specification (online). URL: https://gitweb.torproject.org/torspec.git/blob/HEAD:/path-spec.txt (Visited 2015-01-14).

[23] Dingledine, R. & Mathewson, N. Tor Protocol Specification (online). URL: https://gitweb.torproject.org/torspec.git/blob/HEAD:/tor-spec.txt (Visited 2015-01-14).

[24] Kadianakis, G. & Hopper, N. The move to a single guard node (online). URL: https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/236-single-guard-node.txt (Visited 2015-01-14).

[25] Dingledine, R., Hopper, N., Kadianakis, G., & Mathewson, N. July 2014. One Fast Guard for Life (or 9 months). In *7th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2014)*.

[26] Serjantov, A. & Sewell, P. October 2003. Passive Attack Analysis for Connection-Based Anonymity Systems. In *Proceedings of ESORICS 2003*, Snekkenes, E. & Gollmann, D., eds, volume 2808 of *Lecture Notes in Computer Science*, 116–131. Springer-Verlag. doi:10.1007/978-3-540-39650-5_7.

[27] Danezis, G. May 2003. Statistical Disclosure Attacks: Traffic Confirmation in Open Environments. In *Proceedings of Security and Privacy in the Age of Uncertainty (SEC 2003)*, Gritzalis, Vimercati, Samarati, & Katsikas, eds, 421–426. IFIP TC11, Kluwer.

[28] Danezis, G. May 2004. The Traffic Analysis of Continuous-Time Mixes. In *Proceedings of the Fourth Workshop on Privacy Enhancing Technologies (PET 2004)*, volume 3424 of *Lecture Notes in Computer Science*, 35–50.

[29] Houmansadr, A. & Borisov, N. July 2013. The Need for Flow Fingerprints to Link Correlated Network Flows. In *Proceedings of the 13th Privacy Enhancing Technologies Symposium (PETS 2013)*.

[30] Feigenbaum, J., Johnson, A., & Syverson, P. July 2010. Preventing Active Timing Attacks in Low-Latency Anonymous Communication. In *Proceedings of the 10th Privacy Enhancing Technologies Symposium (PETS 2010)*.

[31] O'Gorman, G. & Blott, S. December 2007. Large Scale Simulation of Tor: Modelling a Global Passive Adversary. In *Advances in Computer Science – ASIAN 2007. Computer and Network Security*, Cervesato, I., ed, volume 4846 of *Lecture Notes in Computer Science*, 48–54. Springer-Verlag. doi:10.1007/978-3-540-76929-3_5.

[32] Fu, X. & Ling, Z. February 2009. One Cell is Enough to Break Tor's Anonymity. In *Black Hat DC 2009*. URL: https://www.blackhat.com/presentations/bh-dc-09/Fu/BlackHat-DC-09-Fu-Break-Tors-Anonymity.pdf.

[33] Ling, Z., Luo, J., Yu, W., Fu, X., Xuan, D., & Jia, W. November 2009. A New Cell Counter Based Attack Against Tor. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS 2009)*, Al-Shaer, E., Jha, S., & Keromytis, A. D., eds, 578–589. ACM Press. doi:10.1145/1653662.1653732.

[34] Biryukov, A., Pustogarov, I., & Weinmann, R.-P. May 2013. Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, 80–94.

[35] Chakravarty, S. *Traffic Analysis Attacks and Defenses in Low Latency Anonymous Communication*. PhD thesis, Graduate School of Arts and Sciences, Columbia University, 2014.

[36] Dingledine, R. Tor security advisory: "relay early" traffic confirmation attack (online). July 2014. URL: https://blog.torproject.org/blog/tor-security-advisory-relay-early-traffic-confirmation-attack (Visited 2015-01-19).

[37] Murdoch, S. J. & Danezis, G. May 2005. Low-Cost Traffic Analysis of Tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*. IEEE CS.

[38] Murdoch, S. J. November 2006. Hot or Not: Revealing Hidden Services by their Clock Skew. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS 2006)*, 27–36. ACM Press.

[39] Bauer, K., Sherr, M., McCoy, D., & Grunwald, D. August 2011. ExperimenTor: A Testbed for Safe and Realistic Tor Experimentation. In *Proceedings of the USENIX Workshop on Cyber Security Experimentation and Test (CSET 2011)*.

[40] Jansen, R. & Hopper, N. February 2012. Shadow: Running Tor in a Box for Accurate and Efficient Experimentation. In *Proceedings of the Network and Distributed System Security Symposium (NDSS 2012)*. Internet Society.

[41] Jansen, R., Bauer, K., Hopper, N., & Dingledine, R. August 2012. Methodically Modeling the Tor Network. In *Proceedings of the USENIX Workshop on Cyber Security Experimentation and Test (CSET 2012)*.

[42] Loesing, K. Overview of Statistical Data in the Tor Network. Technical Report 2011-03-001, The Tor Project, March 2011. URL: `https://research.torproject.org/techreports/data-2011-03-14.pdf`.

[43] Liu, C., White, R. W., & Dumais, S. July 2010. Understanding Web Browsing Behaviors through Weibull Analysis of Dwell Time. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR 2010, 379–386. ACM Press. `doi:10.1145/1835449.1835513`.

[44] Josefsson, S. October 2006. The Base16, Base32, and Base64 Data Encodings. RFC 4648 (Proposed Standard). URL: `http://www.ietf.org/rfc/rfc4648.txt`.

[45] National Institute of Standards and Technology. The Keyed-Hash Message Authentication Code (HMAC). Federal Information Processing Standards Publication FIPS PUB 198-1, Information Technology Laboratory – National Institute of Standards and Technology, Gaithersburg, MD, USA, July 2008. URL: `http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf`.

[46] Fawcett, T. June 2006. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8), 861–874. `doi:http://dx.doi.org/10.1016/j.patrec.2005.10.010`.

[47] The Tor Project. Tor FAQ (online). URL: `https://www.torproject.org/docs/faq.html.en` (Visited 2015-02-18).

[48] Tang, C. & Goldberg, I. October 2010. An Improved Algorithm for Tor Circuit Scheduling. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS 2010)*, Keromytis, A. D. & Shmatikov, V., eds. ACM Press.

[49] Schwarz, K. Darts, Dice, and Coins: Sampling from a Discrete Distribution (online). December 2011. URL: `http://www.keithschwarz.com/darts-dice-coins/` (Visited 2015-03-02).

[50] Jansen, R. shadow-plugin-tor wiki (online). URL: `https://github.com/shadow/shadow-plugin-tor/wiki` (Visited 2015-03-20).

[51] McCoy, D., Bauer, K., Grunwald, D., Kohno, T., & Sicker, D. July 2008. Shining Light in Dark Places: Understanding the Tor Network. In *Proceedings of the Eighth Privacy Enhancing Technologies Symposium (PETS 2008)*, Borisov, N. & Goldberg, I., eds, 63–76. Springer-Verlag.

[52] The Tor Project. Torperf – Tor performance evaluation tools (online). URL: `https://gitweb.torproject.org/torperf.git/` (Visited 2015-04-08).

[53] Jansen, R. Shadow – Simulation Execution and Analysis (online). URL: `https://github.com/shadow/shadow/wiki/2-Simulation-Execution-and-Analysis` (Visited 2015-04-08).

[54] Edman, M. & Syverson, P. November 2009. AS-awareness in Tor Path Selection. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS 2009)*, Al-Shaer, E., Jha, S., & Keromytis, A. D., eds, 380–389. ACM Press.

[55] Vanbever, L., Li, O., Rexford, J., & Mittal, P. October 2014. Anonymity on QuickSand: Using BGP to Compromise Tor. In *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*. ACM Press. `doi:10.1145/2670518.2673869`.

[56] Dingledine, R. & Mathewson, N. Design of a blocking-resistant anonymity system. Technical Report 2006-11-001, The Tor Project, November 2006. URL: `https://research.torproject.org/techreports/blocking-2006-11.pdf`.

[57] The Tor Project. Tor: Pluggable Transports (online). URL: `https://www.torproject.org/docs/pluggable-transports.html.en` (Visited 2015-04-25).

[58] The Tor Project. Tor Rendezvous Specification (online). URL: `https://gitweb.torproject.org/torspec.git/blob/HEAD:/rend-spec.txt` (Visited 2015-04-27).

[59] Panchenko, A., Niessen, L., Zinnen, A., & Engel, T. October 2011. Website Fingerprinting in Onion Routing Based Anonymization Networks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2011)*. ACM Press.

[60] Perry, M. A Critique of Website Traffic Fingerprinting Attacks (online). November 2013. URL: `https://blog.torproject.org/blog/critique-website-traffic-fingerprinting-attacks` (Visited 2015-04-30).

[61] Perry, M., Clark, E., & Murdoch, S. The Design and Implementation of the Tor Browser [DRAFT] (online). May 2015. URL: `https://www.torproject.org/projects/torbrowser/design/` (Visited 2015-05-14).

[62] Dingledine, R. [tor-talk] Who said it takes hours of latency to fix anonymity? (online). February 2015. URL: `https://lists.torproject.org/pipermail/tor-talk/2015-February/036912.html` (Visited 2015-05-15).