Analysis of Client Anonymity in the Tor Network

Christian August Holm Hansen

Master of Science in Communication Technology

Submission date:	June 2015
Supervisor:	Stig Frode Mjølsnes, ITEM
Co-supervisor:	Otto Wittner, ITEM
	Rune Sydskjør, Uninett

Norwegian University of Science and Technology
Department of Telematics

Title:	Analysis of Client Anonymity in the Tor Network
Student:	Christian August Holm Hansen

Problem description:

The Tor network combines source routing with layers of encryption to achieve sender anonymity. The general goal of this assignment is to understand the technical strengths and weaknesses of the Tor network, and recommend the best practice for operating Tor nodes.

The first task is to establish, configure and successfully run a Tor exit node in an acceptable and secure mode within the network domain of NTNU. The second task is to experiment with and verify a selection of practical attacks reported in the literature. For instance, Hopper, Vasserman and Chan-Tin [1] claim that packet latency through Tor will leak identifying information. Finally, propose an improved attack on Tor security mechanisms, and if time allows, substantiate this with experimental measurements.

[1] N. Hopper, E. Y. Vasserman and E. Chan-Tin. "How much anonymity does network latency leak?" ACM Transactions on Information and System Security (TISSEC) 13.2 (2010)

Responsible professor:	Stig Frode Mjølsnes (ITEM)
Co-supervisors:	Otto Wittner (ITEM/Uninett)
	Rune Sydskjør (Uninett)
	Øystein Vik (NTNU)

Abstract

The Tor Network has emerged as the most popular service providing sender anonymity on the Internet. It is a community-driven network with most of the infrastructure operated by volunteers.

Peer-To-Peer (P2P) file sharing applications, such as BitTorrent, take up a large portion of the available resources in Tor, which reduce the quality of service for those browsing the web through Tor. In this thesis, experiences from operating a Tor exit relay with a reduced exit policy are recounted. Additionally, the lifecycle of the exit relay is presented and an analysis of the application distribution of exit traffic is done. This analysis uncovers that the reduced exit policy may reduce the BitTorrent traffic share as the total, byte-wise traffic share constituted by BitTorrent was 25.4%, which is lower than in similar analyses done earlier.

Tor is a low latency service, thus it is possible that packet latency can leak information about either the source, the destination or both ends of the encrypted Tor traffic. There have been numerous proposals for side-channel attacks in the Tor Network, with one of the most interesting being the website fingerprinting attack. The website fingerprinting attack attempts to map encrypted client-side traffic with a web page by utilizing side-channel information from web page visits to train a machine learning classifier, which in turn is used to predict the web page corresponding to encrypted, client-side Tor traffic. This thesis aims to review existing website fingerprinting attacks as well as to propose a basic attack sorting under this category. The thesis argues that it is feasible that state of the art web site fingerprinting attacks can be applied in a real-world scenario under the assumption that certain Tor users visit censored web pages repeatedly.

Website fingerprinting attacks proposed up until now attempt to identify individual web *pages* from an encrypted traffic stream. This thesis proposes a web *site* fingerprinting attack, an attack related to the general website fingerprinting attack, but instead of web *pages*, it attempts to identify web *sites*. The attack utilizes, among other things, the browsing pattern to attempt to map encrypted client-side traffic to a web *site*. The browsing pattern data is collected from a test group made up of volunteers who are asked to browse web sites as they feel natural. In one of the most successful experiments, the attack resulted in a True Positive Rate (TPR) of 91.7% and a corresponding False Positive Rate (FPR) of 0.95% from a total of 222 attempted web site predictions.

Sammendrag

Tor-nettverket er den mest populære tjenesten for senderanonymitet på Internett og mesteparten av infrastrukturen er drevet av frivillige.

Fildelingsapplikasjoner som BitTorrent bruker store deler av de tilgjengelige ressursene i Tor-nettverket og fører til en reduksjon i tjenestekvaliteten for de som bruker Tor til å besøke nettsider. I denne avhandlingen vil erfaringer fra å drifte en utgangsnode bli presentert. En utgangsnode er den siste noden klartekstpakker blir sendt gjennom før de blir sendt videre til sin endelige destinasjon i Internett. Livsløpet til utgangsnoden samt en analyse av utgangstrafikken vil også bli presentert. Denne analysen fastslår at en redusert utgangspolicy, som begrenser hvilke porter som tillater utgangstrafikk, kan redusere andelen BitTorrent-trafikk. Denne typen trafikk utgjorde 25.4% av totalen, noe som er betydelig mindre enn i lignende analyser gjort tidligere.

Tor er en tjeneste med lav tidsforsinkelse, noe som tilsier at pakkemønsteret kan lekke informasjon om kilden, destinasjonen eller både kilden og destinasjonen til den krypterte trafikken. Det har tidligere blitt publisert en rekke forslag til sidekanalsangrep i Tor-nettverket hvor et av de mest interessante er "website fingerprinting"-angrep (nettsidefingeravtrykksangrep). Dette angrepet forsøker å kartlegge koblinger mellom kryptert klient-trafikk og nettsider ved å bruke sidekanalsinformasjon fra nettsidebesøk som treningsdata til en veiledet maskinlæringsmodell. Denne modellen blir deretter brukt til å predikere hvilken nettside som hører til gitt, kryptert klient-trafikk. Denne avhandlingen gjør en vurdering av eksisterende angrep i denne kategorien, i tillegg til å foreslå et enkelt angrep samt eksperimentere på dette. Avhandlingen argumenterer for at et nettsidefingeravtrykksangrep kan være gjennomførbart i praksis dersom man antar at enkelte Tor-brukere besøker sensurerte nettsider gjentatte ganger.

De fleste nettsidefingeravtrykksangrep forsøker å identifisere nettsider. Denne avhandlingen foreslår et nettstedfingeravtrykksangrep som tar sikte på å identifisere nettsteder ved hjelp av blant annet brukermønster. Brukermønsterdataene blir samlet ved hjelp av en testgruppe med frivillige hvor testerne blir bedt om å besøke nettstedene slik de føler er naturlig. I et av de mest vellykkede forsøkene resulterte angrepet i en sann positivrate på 91.7% og en tilhørende falsk positiv-rate på 0.95% på totalt 222 forsøk på å predikere nettsteder basert på kryptert Tor-trafikk.

Acknowledgements

I would like to thank Professor Stig F. Mjølsnes for giving me the opportunity to write this thesis and for his encouragement and help in deciding the topics of the thesis as well as throughout the course of the thesis work.

Secondly, I want to thank my supervisors at UNINETT, Otto Wittner and Rune Sydskjør for being part of the thesis process and answering my questions, whether they be about statistics or what the next logical step in the thesis work is.

I would also like to thank the guys at the IT department, Øystein Vik and Ole Langfeldt for allowing me to operate an exit node in the university network domain as well as helping with the thesis work.

The volunteers who took time out of their own hectic thesis periods to gather data for my experiments also deserves a big thanks. Without you, I would not be able to substantiate the web *site* fingerprinting attack with experimental results.

Additionally, I want to thank Ida and Kristian for reviewing the thesis. The viewpoints of somebody without a technical background has been a great help in improving the overall quality of the thesis.

Last, but not least, I want to thank Mia for putting up with my jabber on Tor, even though this falls completely outside your field of interest. You have also been a great help in the thesis work with your encouragement and linguistic knowledge. My time in Trondheim would not have been the same without you - I love you.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

DHT Distributed Hash Table.

DMCA Digital Millennium Copyright Act.

DoS Denial of Service.

FPR False Positive Rate.

HMM Hidden Markov Model.

IP Internet Protocol.

ISP Internet Service Provider.

MitM Man-in-the-Middle.

NPV Negative Predictive Value.

NSM Norwegian National Security Authority.

NTNU Norwegian University of Science and Technology.

NTP Network Time Protocol.

P2P Peer-To-Peer.

PDU Protocol Data Unit.

PPV Positive Predictive Value.

SMTP Simple Mail Transfer Protocol.

SSH Secure Shell.

SVM Support Vector Machine.

TCP Transmission Control Protocol.

TPR True Positive Rate.

List of Symbols

C Website fingerprinting classifier.

T Set of training examples used to train a website fingerprinting classifier.

M Set of pages or sites included in the attack model of a website fingerprinting attack.

Φ Set of monitored pages or sites chosen from the attack model, M.

Γ Set of pages or sites marked as censored chosen from the set of monitored pages or sites, Φ, in an open world model.

Ω Set of every existing web page or site.

s A single web site.

p A single web page.

$\vec{\mathbf{f}}$ Feature vector constructed from the traffic generated from visiting a web page or site in a website or web *site* fingerprinting attack.

$\vec{\mathbf{f}}_p$ Feature vector to be predicted as a web page or site.

$|Z|$ The cardinality of set Z. The cardinality of a set of feature vectors, $|\{\vec{\mathbf{f}}\}|$, signifies the number of feature vectors in the set.

b Burst tolerance threshold, the maximum time allowed to the previous and next Tor cell for a set of Tor cells to be labelled as a burst.

$\Delta v_{threshold}$ Threshold for the minimum difference between the highest and second highest cumulative vote for a feature vector to be classified as a censored site.

$v_{threshold}$ Threshold for the minimum value of the highest cumulative vote for a feature vector to be classified as a censored site.

Glossary

Censored Web Page or Site A web page or site included in a subset of the monitored sites in an open world model classifier constituting one of the two classes in the classifier.

Clearnet Comprised of all Internet services that can be accessed using a normal browser.

Closed World Model Each web page or site in the classifier constitutes a unique class [PNZE11]. The closed world model classifier does not handle web pages or sites not included in a limited set of monitored pages or sites.

Feature Vector A vector containing information, extractable from the traffic data, that may aid in the effort to uniquely identify a web page or site from a given set of pages or sites.

Hypothesis Set A set of functions mapping an input object to an output value [MRT12]. In website fingerprinting, a feature vector is used as input to the hypothesis set to predict the corresponding class.

Monitored Web Page or Site A web page or site that has at least one training example in the set of training examples used to train the classifier.

Onion Network Comprised of all Internet services that can only be accessed by a client connected to the Tor Network.

Open World Model The training examples of different web pages or sites has an output value corresponding to the page or site, but there are only two possible outcome classes of a prediction, censored and uncensored [PNZE11]. The open world model classifier has the ability to handle web pages or sites not included in the set of monitored web pages or sites.

The Tor Network The most popular implementation of Tor in the form of the overlay network consisting of Tor relays, hidden services, directory authorities and Tor clients.

Tor The underlying design of the anonymous communication service as defined in "Tor: The Second-Generation Onion Router" [DMS04].

Tor Relay Proxy server that forwards the traffic through the Tor Network. A Tor relay has a policy which describes, among other things, whether the relay can be used as an exit relay or not.

Training Example A pair consisting of an input object and an output value used to train a classifier [MRT12]. In website (or web *site*) fingerprinting, a feature vector represents the input object and a web page (or site) represents the output value.

Uncensored Web Page or Site A web page or site not included in the set of censored web pages or sites. It is either a member of the monitored web pages or sites, or the unmonitored web pages or sites included in the corresponding open world model.

Web *Site* Fingerprinting Attack A side-channel attack attempting to identify web sites from encrypted web traffic.

Web Page A single document accessible through the Clearnet or the Onion Network.

Web Site The entire collection of linked content under a single domain, normally comprised of multiple web pages.

Web Site and Web Page Prediction The application of the functions in the hypothesis set on a feature vector to produce a qualified guess on the class from a given set of classes. A class may be a single web page or site or a set of web pages or sites.

Website Fingerprinting Attack A side-channel attack attempting to identify web pages from encrypted web traffic.

Website Fingerprinting Classifier Supervised learning model, trained with training examples, able to infer a hypothesis set used for web page or site prediction.

Chapter 1

Introduction

1.1 Motivation

Anonymity in digital communications has received an increasing amount of attention over the last few decades. The users of such anonymous services range from governments, military personnel and law enforcement to whistleblowers, criminals and individuals wanting to protect their privacy and avoid leaving a digital footprint everywhere they go in their digital lives. Because of the user-base of these systems, a continuous war is fought between those seeking to provide true anonymous services and those seeking to deanonymize the users of the same services. Receiver anonymity in the Internet and other digital communications systems is fairly easy to achieve as the data could simply be broadcasted. Sender anonymity, on the other hand, has proven to be a more challenging problem.

The research on anonymity in the Internet dates back to the early days with David Chaum's seminal work on an electronic mail system hiding not only the content of the communication, but also the identity of the participants [Cha81]. Anonymous remailers were the big thing in Internet anonymity in the 90's, and in 1993, the pseudonymous remailer penet.fi was set up by Johan Helsingius. In an interview with IC Magazine, Helsingius said the following when asked to explain the reason why he set up such a remailer [Gra94]:

> *"Some people from a university network really argued about if everybody should put their proper name on the messages and everybody should be accountable, so you could actually verify that it is the person who is sending the messages. And I kept arguing that the Internet just doesn't work that way, and if somebody actually tries to enforce that, the Internet will always find a solution around it."*

penet.fi was discontinued in 1996 as a result of reports of a compromise in 1994, and, another, in 1995 when the Church of Scientology contacted Interpol requesting the identity of a penet.fi user. This lead to the issuing of a search warrant by the Finnish police, forcing Helsingius to disclose the real e-mail address of said user. However, Helsingius was right in saying that the Internet would find a solution. After the downfall of penet.fi, an array of anonymous remailers appeared, some of the most notable ones being Mixmaster in 1998 [Cot94] and Mixminion in 2002 [DDM03], both based on the work of David Chaum [Cha81].

The next frontier concerning anonymity on the Internet is sender anonymity in the Web. This is a more challenging problem than anonymous remailers as there are stricter latency requirements, higher differentiation in traffic type and limited padding possibilities in the Web compared to in e-mail. Services to increase the anonymity and privacy of the sender include, among other things, proxy servers and VPNs. Proxy servers can be compared to penet.fi as they forward the traffic, causing the address of the proxy server to appear as the source address of the traffic. However, proxy servers have, like penet.fi, a single point of failure as the identities of the clients can be disclosed by the Internet Service Provider (ISP) monitoring the traffic or by the operator of the proxy server, e.g. as the result of a search warrant.

Onion routing, an infrastructure principle using a series of proxies to achieve anonymous communication over a public network was developed by researchers working for the U.S. Naval Research Laboratory in the mid-'90s [GRS96]. In 2002, one of the designers of the onion routing principle, Paul Syverson, was joined by Roger Dingledine and Nick Mathewson to develop an implementation of onion routing dubbed the TOR project or The Onion Routing project, releasing the first version of Tor in September 2002. In 2004, they presented "Tor: The Second-Generation Onion Router" which contained improvements to the original design and implementation [DMS04]. Later that same year, the Naval Research Laboratory released the Tor code under a free license which would mark the beginning of the community driven Tor Network, providing the means for web anonymity for all. The decision to make the Tor Network readily available for everyone was probably not done for altruistic reasons, but rather to increase the level of anonymity of the users. One can not be anonymous alone, and the larger the user base of an anonymization system, the stronger the anonymity.

The Tor Network is distributed and operated by a community of volunteers. For the Tor Network to function as intended, the community must include a decent number of altruistic relay operators who maintain the proxy servers, which are the relays used to route the traffic through the network. The exit relay, which is the last proxy server that the Tor traffic is routed through before it is sent out in the Internet as normal traffic, is a minority in the pool of relays. It is therefore especially important that relay operators agree to operate exit relays as well as intermediates.

Many have posed the question of whether the Tor Network is safe or not. Does it really provide sender anonymity for its connected clients? Even though the cryptographic primitives used in Tor are considered sufficiently secure, side-channel attacks have proven to be successful to a certain degree.

1.2 Scope and Objectives

1.2.1 Scope

This thesis will touch areas outside of Tor, but the scope is mainly limited to Tor and the Tor Network. Concerning the new side-channel attack that is proposed (see O.3 and R.4), the experimentation is limited to a proof of concept implementation and a limited data set. The reason for this is that the data collection phase in the proposed attack is very time-consuming and involves using a test group made up of volunteers. Additionally, because the data collection should be done in a controlled environment, it can not be externally distributed and the volunteers must be physically present when gathering the data.

1.2.2 Objectives

O.1: Operate a Tor exit relay

Objective one involves establishing, configuring and successfully running a Tor exit relay in an acceptable and secure mode within the network domain of the Norwegian University of Science and Technology (NTNU).

O.2: Review website fingerprinting

Objective two involves reviewing some of the existing publications on the website fingerprinting attack, which is a passive side-channel attack on encrypted web traffic to reveal the web page the traffic corresponds to.

O.3: Propose a side-channel attack on Tor

The final objective involves the proposal of a side-channel attack on clients in the Tor Network. The attack is similar to the general website fingerprinting attack, but with a new spin.

1.2.3 Research Questions

R.1: What applications are responsible for the Tor exit traffic?

To answer this research question, the application distribution of the traffic exiting through a given Tor exit relay is analyzed.

R.2: To what extent can BitTorrent traffic in Tor be minimized while still offering a decent number of services?

To answer this research question, the built-in configuration features of Tor are used in an attempt to reduce BitTorrent traffic. The BitTorrent traffic share is then compared to earlier application distribution analyses done on the Tor Network to estimate to what extent the BitTorrent traffic is reduced.

R.3: Are state of the art website fingerprinting attacks feasible in a real-world scenario?

To answer this question, state of the art website fingerprinting attacks are reviewed, and the implications of extending the theoretical attacks to a real-world scenario are discussed.

R.4: Can the browsing pattern of Tor users be used to train a website fingerprinting classifier to identify web sites?

To answer this question, a web *site* fingerprinting attack is proposed, based on, among other things, browsing patterns. The attack is substantiated with experiments on a limited data set through a proof of concept implementation of the attack.

1.2.4 Deviation from Problem Description

The problem description states that an "improved attack on Tor security mechanisms" is to be proposed. The web *site* fingerprinting attack proposed differentiates from other attacks to such an extent that it is difficult to compare it and insistently state that it is "improved". Additionally, the data used to experiment on the attack is limited, and it can not be said for certain whether it scales well. Instead of an improved attack, it makes more sense to call it a new attack making use of new ideas and side-channels compared to previously published attacks.

1.3 Outline and Results

Chapter 2 describes the technical details of Tor and the Tor Network deemed important for the thesis in addition to mentioning some of the attack threats present for the users of Tor.

Chapter 3 gives a first-hand experience of operating a high-bandwidth Tor exit relay within the network domain of a university campus. This chapter is related to objective O.1. The life-cycle of the exit relay is given and an analysis of the application distribution is presented. Concerning research questions R.1 and R.2, chapter 3 shows that HTTP traffic amounted to the highest portion of the exit traffic with 55.69% with BitTorrent coming in second amounting to 25.4% of the exit traffic. The traffic share of BitTorrent and other applications is also compared to the data from previous research.

Chapter 4 describes the website fingerprinting attack in detail. This chapter is related to objective O.2 as important research in the website fingerprinting domain is reviewed. A basic website fingerprinting attack is also proposed and experimented upon. The source code of this attack is available on the author's public GitHub page [Han15]. Chapter 4 shows that it is the time interval in which the data collection phase is carried out, that has the biggest impact on the result of the basic website fingerprinting attack proposed. This chapter also provides the foundations for answering research question R.3.

Chapter 5 proposes a new side-channel attack on Tor and thus fulfills objective O.3. A proof of concept implementation is used to gather data and to perform experiments on the gathered data to determine whether browsing patterns can be used to discover the web *site* corresponding to encrypted HTTP Tor traffic in a web *site* fingerprinting attack. The source code of this attack is available on the author's public GitHub page [Han15]. The experimental results presented in chapter 5 are promising in regards to answering research question R.4 with an average accuracy of 85.58% in a closed world model and an average accuracy of 79.97% in an open world model. Possible countermeasures against the proposed attack are also discussed.

Chapter 6 summarizes the thesis' findings and compares these to the original objectives, O.1 through O.3 to assess whether the objectives have been met. The chapter also discusses and explicitly answers the thesis' four research questions R.1 through R.4. Finally, concluding remarks and recommendations for future work are given in chapter 7.

Chapter 2

The Tor Network

Tor is a low latency anonymity system striving to provide sender anonymity through Transmission Control Protocol (TCP) streams that are infeasible to trace. Although similar services exist, such as I2P[1] and `AN.ON`[2], Tor is currently by far the most popular one, with an estimated number of directly connected users in excess of two million on a daily basis according to the Tor Metrics Portal[3].

Tor and the Tor Network are based upon principles developed as far back as in 1996. Since the first version of Tor was published in 2002, it has evolved through the effort of volunteers contributing with possible attacks, proposing mitigations for these attacks and updating to the source code to mitigate attacks and improve the user experience for Tor users. This chapter provides a high-level introduction to some of the important concepts to familiarize the reader to Tor.

Firstly, in section 2.1 some of the core concepts of Tor are defined to avoid ambiguity later in the thesis. Secondly, a quick introduction to some of the basic concepts of Tor is given in section 2.2. This includes the encryption method, the method of maintaining the state information in Tor, what defines an exit relay, latency in Tor and hidden services. The low-level technical details of the design are omitted as they are somewhat irrelevant for the research presented later in this thesis. Finally, some attacks and threats to the users of Tor are discussed in section 2.3. This section presents two categories of threats against the Tor Network important for this thesis, namely malicious relays and traffic analysis attacks.

[1] jrandom et al. *The Invisible Internet Project (I2P)*. `URL: https://geti2p.net`. Accessed: May 2015.

[2] H. Federrath et al. *Project AN.ON*. `URL: http://anon.inf.tu-dresden.de/`. Accessed: May 2015.

[3] Tor Metrics. `URL: https://metrics.torproject.org/`. Accessed: May 2015.

2.1 Conceptual Definitions

Tor is the underlying design of the anonymous communication service as presented in [DMS04].

The **Tor Network** is the implementation of Tor in the form of the overlay network consisting of Tor relays. Since the source code of Tor is freely available, there may exist multiple tor networks. In this thesis, however, the Tor Network consistently refers to the most popular implementation of Tor administered by the Tor Project.

A **Tor relay** forwards the traffic through the Tor Network. A Tor relay has a policy which determines, among other things, whether the relay can be used as an exit relay. In the literature, the terms relay, node and server are used interchangeably to refer to the same thing. In this thesis, the term Tor relay is used consistently.

The **Clearnet** is comprised of all Internet services that can be accessed using a normal browser while the **Onion Network** is comprised of services that can only be accessed by a client connected to the Tor Network.

2.2 A Quick Introduction to Tor

The Tor Network uses source routing and cryptography to achieve communication anonymity. To communicate with a service located in the Clearnet, a client builds a circuit through the Tor Network consisting of three relays: an entry relay, an intermediate relay and an exit relay. By default, these relays are chosen semi-randomly by the client based on requirements such as the geographical distribution of the relays. A client may manually choose which relays to include in the circuit as well.

When building the aforementioned circuit, the client establishes a cryptographic relationship with each of the three relays in the form of a symmetric key. The original data is encrypted in layers so that the entry relay knows the client, in the form of an Internet Protocol (IP) address, but not the destination. The exit relay, on the other hand, knows the destination but not the client. In other words, there is no single relay that has information about both the destination and the origin of the data. This concept of multi-layer encryption is the fundamental principle of onion routing, which Tor is based upon [DMS04].

The symmetric keys shared between the client and each of the relays contained in the circuit are established through a TLS handshake done with each of the circuit's relays in turn. The encryption of a given client's circuit for traffic destined for the Clearnet is explained through the analogy in Figure 2.1.

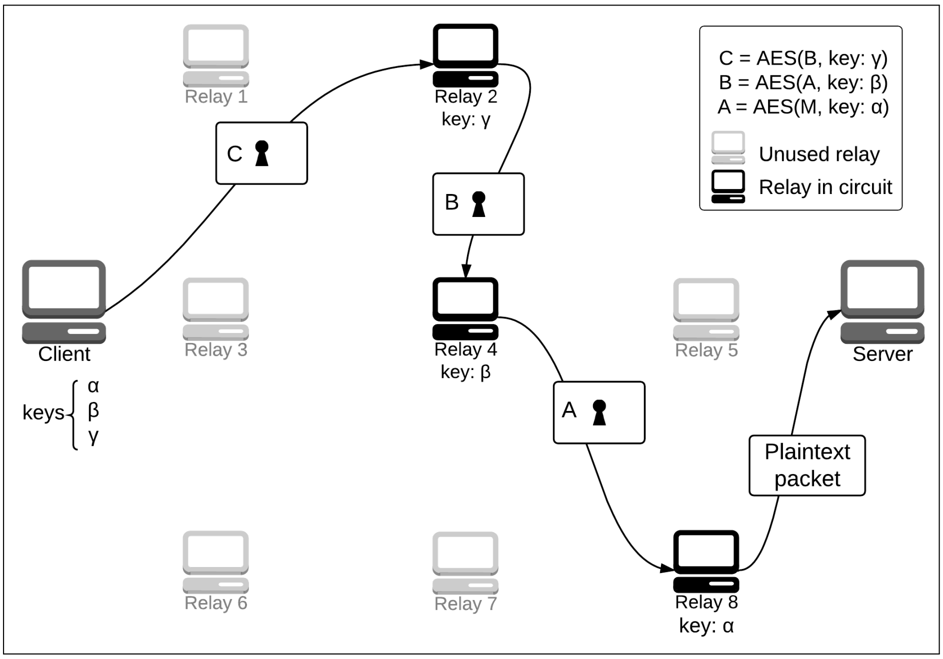


Figure 2.1: Analogy for the encryption method of traffic over a circuit in the Tor Network. The client locks the plaintext packet inside a box, called box `A`, using key α shared between the client and the exit relay. Box `A` is then locked inside box `B`, using key β shared between the client and the intermediate relay, together with a note describing that box `A` is to be sent to the exit relay. Finally, box `B` is locked inside box `C`, using key γ shared between the client and the entry relay, together with a note describing that box `B` is to be sent to the intermediate relay. The client sends box `C` to the entry relay and, even if the box is intercepted, it can not be opened without key γ. The entry relay then opens box `C` with key γ, forwards box `B` to the intermediate relay which in turn opens that box and forwards box `A` to the exit relay. The exit relay then opens the box with key α and forwards the enclosed plaintext packet to its destination in the Clearnet. In the event that a packet arrives at the exit relay destined for the client, the same boxes and keys can be used to forward said packet along the circuit to the client.

2.2.1 The Tor Exit Relay

The Tor exit relay is responsible for connecting circuits in the Tor Network to the Clearnet. Since the exit relay is the final point the Tor traffic passes through before reaching its destination, the IP address of the exit relay is interpreted as the source of the traffic. This signifies that, in the event of abuse complaints on Tor traffic, it is likely that the operator of the exit relay involved in said traffic will be the recipient of these complaints.

The operator of a Tor relay specifies the policy which determines, among other things, whether the relay can be used as an exit relay. Even though the relay operator allows exit traffic through the relay, it can also be used as an entry relay or an intermediate relay [DMS04]. *Directory servers* are relays in the Tor Network tasked with maintaining the state information [DMS04]. The state information includes the addresses of the relays currently in the network as well as *flags* for each relay, describing the capabilities of said relay. One of these flags is the exit flag, which indicates whether it is more useful to use the relay as an exit relay or as a regular relay. Some of the important flags and their explanations are listed in Figure 2.2. The task of deciding the state information falls on the *directory authority servers* which will vote on the flags for each of the network's relays once every hour to achieve a consensus on the state information of Tor.

`Exit`
: The relay is configured to act as an exit relay and has more use as the final hop in a circuit than as a middle relay.

`BadExit`
: The relay advertises exit capabilities but is believed to be useless as an exit relay because it is malicious or misconfigured. The relay can still be used as an entry or middle relay in a circuit.

`Guard`
: The relay is suitable to be used as an entry guard. Clients have the option to select a few entry guards and, whenever they construct a new circuit, they always choose one of their entry guards as the entry relay. In this way, the client's circuit can not be compromised unless an adversary is controlling one of their entry guards.

`Authority`
: The relay is a directory authority server.

`V2Dir`
: The relay is a directory server.

`HSDir`
: The relay is a hidden service directory.

`Fast`
: The relay has a high advertised bandwidth and has use as a relay in a high-bandwidth circuit.

`Running`
: The relay is running and usable.

`Valid`
: The relay is running a valid version of Tor and is thought to be a legitimate relay by the directory authority servers. If the valid flag is deactivated, it will not be selected to be part of Tor circuits.

Figure 2.2: Selected flags set by a consensus among the directory authorities for each relay in the Tor Network.

2.2.2 Latency in Tor

Tor is a low latency service, as opposed to some of the other popular, anonymous communication systems such as the Mixminion remailer [DDM03]. The low latency requirement for Tor lead to certain trade-offs being done between anonymity and efficiency [DMS04].

The fact that Tor is low latency is a trait that has been taken advantage of in some of the attacks on Tor. Because the traffic going over a given Tor circuit typically contains many packets, an attacker who can eavesdrop both ends of the circuit (e.g. at the entry relay and exit relay) can correlate the timing of the packets to deanonymize the client. Section 2.3 discusses some attacks based on the low latency of Tor. Additionally, the low latency trait is important for the attacks proposed later in this thesis.

2.2.3 Hidden Services

The technical details behind hidden services fall somewhat outside of the scope of this project. However, hidden services are an important part of Tor and are therefore briefly mentioned here, but are mostly omitted from the rest of the thesis.

In addition to client anonymity, Tor can also provide responder anonymity in the form of hidden services. These services are often normal web services, but they are deployed in the Onion Network and are not reachable from regular browsers through the Clearnet.

As with all Tor circuits, the central concept in communicating with hidden services is that no single relay knows both the sender and the receiver. A hidden service has a number of *introduction points* used to announce the service and for clients to connect to the hidden service. A hidden service announces itself by sending its URL and public key to an introduction point, which subsequently sends it to a Tor relay with the `HSDir` flag activated for storage. The URL of a hidden service is generally a 16 character hash based on the public key of the hidden service and are placed on the virtual top-level domain `.onion` (e.g. `http://3g2upl4pq6kufc4m.onion/`).

For a client to connect to a hidden service, it requests the public key of the hidden service corresponding to the supplied URL from a `HSDir` relay. The client subsequently utilizes a *rendezvous point* (RP), an introduction point and a *rendezvous cookie* (RC) to build a circuit to the hidden service. The rendezvous point mediates the traffic, both in the construction of the circuit and in the communication itself, between the client and hidden service. The rendezvous cookie is a one-time secret used in a Diffie-Hellman (DH) handshake to build a cryptographic relationship between the client and the hidden service. The method used to construct a circuit between a client and hidden service is shown in Figure 2.3.

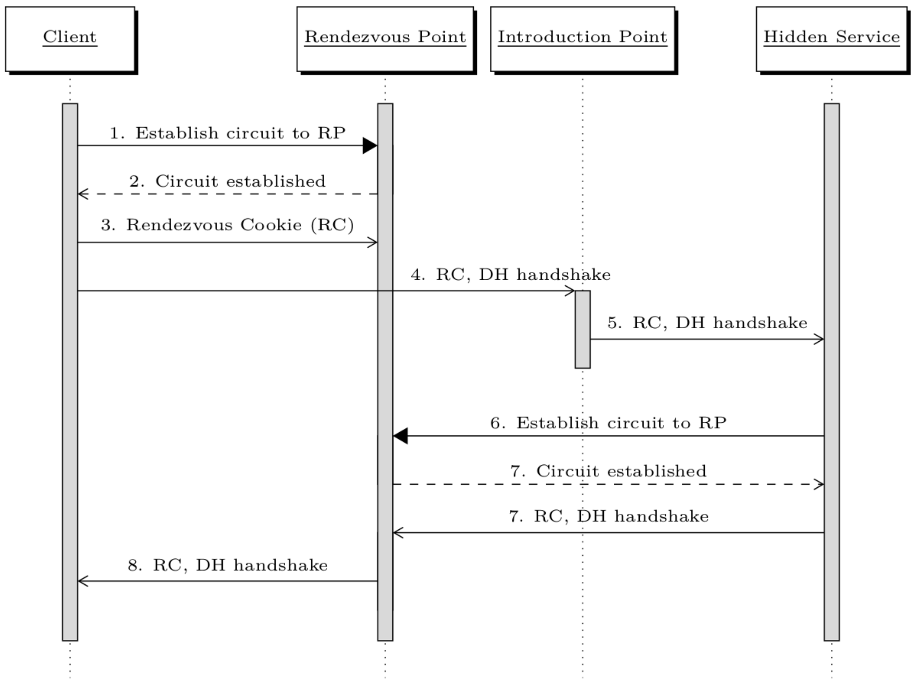


Figure 2.3: Establishment of an anonymous stream between a Tor client and a Hidden Service. Message 4. is encrypted with the public key of the hidden service. Once the hidden service receives the rendezvous cookie in message 5. it can establish the session key, k_s. If the hidden service wishes to communicate with the client, it establishes a circuit to the rendezvous point. It subsequently sends the second half of the Diffie-Hellman handshake, signed with k_s through the rendezvous point to the client. The rendezvous point can identify which client the message is destined to by looking at the rendezvous cookie. When the client receives the second half of the Diffie-Hellman handshake in message 8. k_s is shared between the client and the hidden service. The communication can now begin and will be routed through 7 relays: client entry; client intermediate1; client indermediate2; rendezvous point; hidden service intermediate2; hidden service intermediate 1; hidden service entry.

2.3 Attacks on Tor

2.3.1 Malicious Relays

Since the Tor Network is a distributed system comprised of relays run by voluntary operators, the possibility of malicious relays will likely always be present. Variants of the Sybil attack [Dou02] can be employed to reduce the availability, through Denial of Service (DoS) attacks, or to deanonymize clients. If a single relay operator controls all three relays in a circuit, matching a client with the plaintext packets at the exit relay is trivial.

In the past, each relay advertised its own bandwidth capabilities to the network. According to Bauer et al., these advertisements were not audited to a high enough extent [BMG$^+$07]. In the PlanetLab research network, they introduced a number of low-bandwidth relays, a certain portion of them maliciously advertising high bandwidth and high uptime. Tor clients will prefer stable relays offering high-bandwidth when constructing their circuits. In an experiment with 60 legitimate relays and 6 malicious relays, 46% of the total circuits included one of the malicious relays [BMG$^+$07]. These results are, however, outdated as the bandwidth for each relay is now audited through the bandwidth consensus feature, which will be explained in chapter 3.

Another critical point in the security of the Tor Network is the exit relay. A single malicious exit relay can, among other things, perform SSL stripping[4], HTTPS Man-in-the-Middle (MitM) attacks, HTML injection and traffic sniffing. Winter et al. attempted to map the malicious and misconfigured relays in the Tor Network, and, during their experiments, which lasted a few months, they found 65 malicious or misconfigured relays [WKM$^+$14]. In the course of the research, there were on average about 1000 exit relays in the Tor Network [WKM$^+$14]. However, this does not mean that 65 out of every 1000 exit relays are malicious or misconfigured, as the high churn rate of the exit relays also needs to be taken into account. If the Tor community actively runs software to check if any exit relays employs an active attack, the `BadExit` flag can be activated relatively quickly and the damage done by the malicious exit relay can be minimized. However, passive attacks, such as traffic sniffing, are normally more difficult to detect. Winter et al. used unique credentials to access an FTP server and fetch mail through IMAP for each exit relay [WKM$^+$14]. They found that 27 of the exit relays reused the credentials at a later time and hence they are assumed to sniff traffic. Even though their research showed that it is possible to detect exit relays sniffing traffic when these are reusing credentials, exit relays can passively sniff the traffic without attempting to reuse anything, in which case they can continue doing so for an indefinite amount of time.

2.3.2 Traffic Analysis Attacks

Even though the Tor Network has its own overlay infrastructure in the relays, the traffic propagates the same network infrastructure as regular Internet traffic. This implies that in addition to the relay operators, the network operators can also monitor

[4] Moxie Marlinspike. *sslstrip*. URL: `http://www.thoughtcrime.org/software/sslstrip/`. Accessed: May 2015.

encrypted Tor traffic. In the article "How Much Anonymity does Network Latency Leak?" by Hopper et al. [HVC10], traffic analysis attacks against Tor and other low latency anonymity networks are discussed. They list timing, packet count and traces left by the visited Tor relays and web pages, maliciously or not, as leaks that can be used in various attacks to retrieve location information on Tor clients or hidden services. Two attacks are also proposed that use malicious web sites to acquire information about the visiting client's current Tor circuit or the location of the client itself [HVC10].

End-to-end Correlation

One family of traffic analysis attacks called end-to-end correlation or traffic confirmation attacks have received much attention in the research community [BMG+07, MZ07, PYFZ08]. It is a kind of timing attack where the adversary can observe both ends of a Tor circuit and correlate the timing patterns. These attacks are highly effective in deanonymizing Tor clients, but they all assume a global, passive adversary that have a high number of trusted Tor relays spread around the globe. This falls outside the threat model of Tor [DMS04] as the low latency requirement makes correlating the traffic streams at both ends a trivial task. Tor rather has a countermeasure in place to mitigate this, which is a requirement of distributed geolocations of IP addresses in a circuit to minimize the risk that a given adversary has control of both the entry and exit relay.

The end-to-end correlation attack is still a very real threat to the anonymity of Tor users and due to recent years' revelations on the surveillance techniques of powerful institutions such as the NSA and GCHQ [NSA07], there is reason to believe that such attacks may take place in the Tor Network already. Recently, Starov et al. proposed a mitigation against attacks from a global adversary with an AS (Autonomous System) aware Tor client, which is able to reduce the amount of vulnerable circuits, i.e. circuits that may include both an entry and exit relay from the same adversary, from 58% of all Tor circuits to 5.8% [SNZ+15]. This research is still at an early stage and there will probably be quite some time until AS-aware clients become widespread in the Tor Network.

Website Fingerprinting

Another interesting family of traffic analysis attacks are the so-called website fingerprinting attacks. These only require the attacker to be able to observe the traffic at a single point between the client and the entry relay, and are therefore within the threat model of Tor. The general concept of this type of attack is to keep an updated list of fingerprints of web pages to monitor. These fingerprints differ from attack to attack but generally include the number of packets, packet sizes, total load time and inter-packet timing of a single, complete page-load. With the list of fingerprints in hand, the attacker monitors a single web page visit of the victim and calculates the correlation between this and the list of fingerprints. The attacker can then say, with a certain degree of confidence, which web page in the fingerprint list the victim has visited, or if the visited web page is not contained in the list.

Website fingerprinting attacks can be used, not only in the Tor Network, but in all cases where the traffic out of the client is encrypted. Website fingerprinting in Tor is more challenging than in normal Internet traffic because Tor sends traffic in *cells* of a fixed size of 512 bytes [DMS04]. Individual packet sizes, which can give a lot of identifying information from a given web page visit, can therefore not be used as one of the fingerprint dimensions. However, research such as that of Panchenko et al. shows that website fingerprinting attacks can be more or less effective in Tor despite the fixed size cells [PNZE11]. Website fingerprinting attacks are discussed further in chapters 4 and 5.

Chapter 3

Operating a Tor Exit Relay

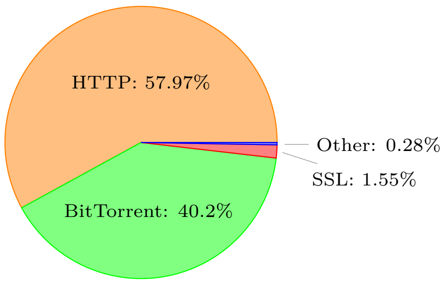
This chapter serves to give a first-hand experience of operating a high-bandwidth Tor exit relay in a secure mode within the network domain of a university campus. The *secure mode* signifies an attempt to limit the malicious activities and the BitTorrent traffic, which are harmful to the quality of the Tor Network. Additionally, the traffic going through the relay is analyzed and compared to earlier, similar studies.

The structure of this chapter is as follows. A summary of related work is given in section 3.1, the necessary preparations regarding the operational security of the relay and abuse prevention are recounted in section 3.2, the methodology for the experiment is presented in section 3.3 and the observations made are discussed in section 3.4. Finally, some recommendations for the best practice when operating a Tor relay is given in section 3.5.

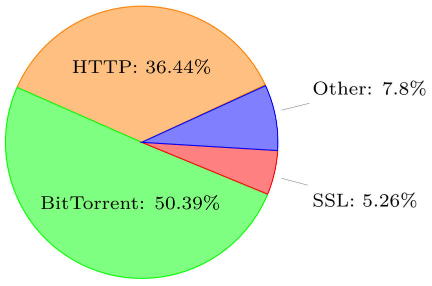
3.1 Related Work

There have not been many scientific publications on the implications of running Tor exit relays or on what kind of traffic that is tunneled through Tor. Mailing lists and blogs made by the Tor community, for the Tor community, do, however, provide sufficient insight into how one should operate exit relays and what the operator can expect. Statistical data on traffic in the Tor Network is an interesting topic but has not received a lot of attention. This is partly because gathering this data can be challenging and partly because storing *any* information about the traffic tunneled through Tor is frowned upon by the community as it potentially can be sensitive data or be used to challenge the anonymity of clients and hidden services.

McCoy et al. published a study to characterize the usage of Tor in 2008 by investigating how Tor is being used, how Tor is being *mis*-used and who is using Tor [MBG$^+$08]. The authors used the Default Exit Policy on a high-speed exit relay, running for four



(a) [MBG+08]



(b) [AMK10]

Figure 3.1: Byte-wise application distribution in Tor traffic with the Default Exit Policy (a) and with all ports open (b).

days, in addition to running a non-exit relay for 15 days. The Default Exit Policy accepts traffic on most ports, but blocks a few in an effort to reduce malicious activity and P2P traffic such as 25 (SMTP), 1214 (Kazaa), 6346-6347 (Gnutella) and 6881-6999 (most often used ports in BitTorrent).

McCoy et al. found, through complaints, that common malicious activities tunneled through Tor are copyright infringement, hacking attempts, botnet command and control and web page defacement [MBG+08]. Furthermore, as can be seen from the application distribution in Figure 3.1a, they found that HTTP is the most used protocol and that BitTorrent is the second most used protocol measured in bytes.

Abdelberi et al. did a similar study in 2010 where they, in addition to doing a study on the application distribution, studied the HTTP content type distribution [AMK10]. They ran six separate exit relays with all ports open (i.e. not the Default Exit Policy) for 23 days. Under the assumption that the application traffic of an unknown type was all BitTorrent traffic (the handshakes were encrypted in these cases), BitTorrent traffic amounted to a whopping 50.39% while HTTP traffic amounted to 36.44% as can be seen in Figure 3.1b. When the application distributions in Figure 3.1a and 3.1b are compared, it can be observed that the amount of SSL traffic is increased, which in all likelihood is because more web sites offered SSL in 2010 than in 2008. Moreover, as the figures show, BitTorrent is responsible for a bigger piece of the total traffic when all ports are open in comparison to having the Default Exit Policy active. Furthermore, Abdelberi et al.'s HTTP content type analysis presented that social networking applications amounted to 9.52% of the HTTP traffic [AMK10]. This is somewhat surprising as social networks implies users have logged in with credentials, and, given the low share of SSL traffic, there is a possibility that some credentials were sent in plaintext through the exit relay.

3.2 Preparations

3.2.1 Operational Security

Running a Tor exit relay involves opening up incoming and outgoing ports to the public in addition to handling the traffic itself. Therefore, it is desirable to take certain precautions concerning the operational security of the relay, including restricting access for the Tor service, limiting running services to a bare minimum and protecting against DoS attacks. This is done to ensure a high uptime percentage and to make sure the research data is protected.

The **physical security** of the relay includes password-protecting the computer itself and keeping it in a locked room where few have access. In addition to this, the swap space and root partition are encrypted to prevent unauthorized access to the sensitive traffic data.

A **restricted environment** can be used to minimize the portion of the system the Tor service has access to. In this project, a *chroot jail* is set up to confine the access of the Tor process and its child processes to an apparent root directory. This root directory only includes the libraries Tor depend on, in addition to the Tor software itself. The recorded traffic data is stored outside this directory.

Stopping unnecessary services is another trick to minimize the available attack surface. Since the relay runs on a Linux distribution intended for normal use, various unnecessary services are started upon boot as the default setting. The only services allowed to run on the relay are Tor, `ntopng`[1] including the dependency `redis-server` and Network Time Protocol (NTP) daemon. Unfortunately, `ntopng` has no option to restrict access based on IP addresses, but requires login credentials, which is considered strong enough (assuming the password is changed from the default "admin" to something stronger). NTP daemon is needed to keep the clock synchronized, but can be secured against DoS attacks by only allowing incoming NTP messages if they are replies to an outgoing request. The following `iptables` commands were run to ensure the firewall was correctly configured for this.

```
-A INPUT -p udp -sport 123 -m state -state ESTABLISHED -j ACCEPT
-A OUTPUT -p udp -dport 123 -m state -state NEW,ESTABLISHED -j ACCEPT
```

Mitigating memory-based DoS attacks is not as trivial as the above security measures, but should still be considered when operating a Tor relay. The Sniper Attack [JTJS14] is an example of such an attack. This is a relatively low resource

[1] ntopng. High-speed web-based traffic analysis and flow collection. URL: `http://ntop.org/products/ntop/`. Accessed: May 2015.

attack, as it only requires a single Tor client and a web server (possibly run on the same client). It works by having the adversary's client build a Tor circuit that includes the victim exit relay and sends a continuous stream of Tor cells destined for the adversary-controlled web server, ignoring the window size. The destination server in turn stops reading from the TCP connection it shares with the exit relay, making the exit relay buffer all the cells until its memory is depleted, thereby effectively crashing the system or causing the OS to kill the Tor process [JTJS14]. Alternatively, the adversary-controlled web server can continuously send *relay SENDME cells* [DMS04] to the exit relay to achieve the same result [JTJS14]. Without any countermeasures in place, effective, targeted DoS attacks can not only seriously harm the total bandwidth of the Tor Network, but can also threaten client anonymity in combination with a carefully orchestrated Sybil Attack [Dou02].

Fortunately, an out of memory handler was introduced in Tor version 0.2.5, that prevents the Sniper Attack (as it is described in the research paper [JTJS14]) by killing the circuits taking up too much memory. However, the identification of memory usage that can boundlessly grow is a work in progress and the developers of Tor make no claim that Tor completely protects against memory-based DoS attacks. As an operator, a simple mitigation against this type of attack is to limit the amount of memory available to Tor and restart the service if it supersedes this limit. This leads to the destruction of all active circuits, both “good” and “bad”, but it is better than having the relay go down for an extended time period.

3.2.2 Abuse Prevention

The Clearnet traffic will have the exit relay address as the source address. This can lead to abuse complaints directed to the operator of the exit relay and, in excessive cases, even an order to take the relay down. This, in combination with the fact that some traffic is harmful to the Tor Network (e.g. BitTorrent), signifies that as a relay operator, one should take some preventative measures to minimize abuse. A simple step that can be done to reduce complaints is to put up an exit relay notice, an informative notice web page, on port 80, that describes the nature of the server.

As an operator, malicious activities can be difficult to mitigate without limiting the service of the relay. When deciding the exit policy of the relay (i.e. which ports to open), the operator should balance the number of services available to the Tor clients up against the probability of malicious activities using the services.

BitTorrent traffic can be the source of copyright infringement complaints such as Digital Millennium Copyright Act (DMCA) notices, even if the relay is situated outside the United States, in addition to having an unwanted effect on the Tor Network. The previous research presented in section 3.1 demonstrated that BitTorrent takes up a significant portion of the total traffic size [MBG$^+$08, AMK10]. In an attempt to reduce this, a *Reduced Exit Policy* is used where only a few selected outgoing ports on the exit relay are opened corresponding to the services allowed. The complete exit policy employed on the relay used in this thesis is accounted for in Appendix A.

Simple Mail Transfer Protocol (SMTP) uses port 25 and can be used for e-mail spamming and is blocked both in the Default and Reduced Exit Policy.

Secure Shell (SSH) bruteforcing is a hacking attempt commonly done through the Tor Network. This was brought to the attention of the author by another exit relay operator and resulted in the denial of SSH traffic altogether by blocking port 22. This can also be seen in the exit policy in Appendix A.

3.2.3 Notifying Affected Personnel

Running a Tor exit relay within the domain of a university campus is bound to trigger reactions. When running an exit relay on a private Internet connection, the Tor community recommends the relay operator to notify their ISP of the intent to run a relay to mitigate the risk that the operator is personally held accountable for the traffic. Similarly, when running a relay on a university campus, the operator should inform, and get permission from, the IT security department, the network operations and other personnel that may be affected.

In this project, an effort was made to notify all affected instances, the project was planned in collaboration with the IT security department and a risk analysis was composed to prepare them for what might happen as a result of operating the exit relay. In the end, it turned out that it was not done extensively enough as the relay was taken down for a period because there was registered contact with a known, blacklisted, IP address. This will be further discussed in section 3.4.1.

3.3 Methodology

3.3.1 Experimental Setup

The details of the system running the exit relay are listed in Table 3.1. The CPU and memory resources should be sufficient to run a high-speed Tor relay in addition to traffic analysis software. `ntopng` was used to monitor the traffic going through the relay. The bandwidth used for the Tor exit relay was limited to a steady 24 Mbps with bursts up to 40 Mbps allowed.

3.3.2 Data Collection

To collect exit traffic data, the exit relay was run for a total period of 18 days from February 23-March 16, 2015. It was taken down for three days between March

Component	Description
Tor version	`0.2.5.10`
Operating System	`Linux Mint 17.1`
Processor	`Intel i7 2.8 GHz`
Memory	`16 GB DDR3`
Network	`100/100 Mbps`

Table 3.1: Exit relay system description.

9-12 as a result of contact with blacklisted IPs (this is more thoroughly explained in section 3.4).

To gather statistics about the application distribution, `ntopng` was used to capture data on the network interface. The choice fell on `ntopng` in favor of the more lightweight `tcpdump` because it made searching for the desired data easier. The flows going to and from other, publicly listed, Tor relays were not included in the application distribution analysis to ensure that only exit traffic was registered. Because of privacy concerns, the application data was discarded and the only traffic data stored were the headers.

The throughput data was collected from the Tor log file together with Onionoo and CollecTor. These are services under the Tor Project umbrella providing more detailed Tor relay statistics such as active flags and other directory authority server consensuses.

3.3.3 Traffic Classification

The traffic was classified based on what application it was destined to. In most cases, the classification was apparent from the packet headers, but a small portion of the traffic could not be classified as one of the applications allowed by the exit policy. In these cases, the traffic was assumed to be BitTorrent traffic, as was the case in the study done by Abdelberi et al. [AMK10]. The reason why a portion of the BitTorrent traffic could not directly be identified is that the connection obfuscation extension, which encrypts the BitTorrent handshakes between peers, was used.

3.4 Observations

In the entire lifetime of the relay, 842 GB was sent and 818 GB was received. It had an average throughput of 4.72 Mbps, something which accounts for less than 20% of the limit set in the exit policy. Furthermore, it had 591 active circuits on average in the period it was running. The flags `Exit`, `Fast`, `Running`, `V2Dir` and `Valid` were present whenever the relay was running and the `HSDir` flag was set to active when it had been running for a while. An incident causing the relay to be taken down is recounted in section 3.4.1 and analyses on the relay life cycle and application traffic distribution is presented in sections 3.4.2 and 3.4.3 respectively.

3.4.1 Malicious Activity

As the application data itself was not stored, it is difficult to identify possible malicious activity tunneled through the relay. On the 14th operation day of the relay, an incident was detected by the Norwegian National Security Authority (NSM), originating from the IP address of the exit relay. They had registered contact with a known, blacklisted botnet agent. This resulted in a report sent to the campus' network operations which subsequently pulled the plug on the relay, causing it to go down for some time. The nature of the relay was made clear to the ones responsible for taking it down and it was allowed back on-line after three days.

3.4.2 Exit Relay Life Cycle

In the life cycle of a Tor relay, a differentiation is made between two phases, the unmeasured phase and the measured phase.

Unmeasured Phase

As mentioned in 3.3.1, the bandwidth is limited to a steady 24 Mbps. However, even though this bandwidth is made available for use to Tor clients, it is not used right away.

Bandwidth authority servers normally estimate the available bandwidth to assist the directory authority servers in achieving a *bandwidth consensus weight* for each relay once every hour. However, in the first 72 hours of operation, a relay is in an *unmeasured phase.* This is done to mitigate malicious relay attacks involving advertising a higher bandwidth than it actually offers, as discussed in section 2.3.2. The bandwidth consensus weight is used as an indication of the reasonable data rate to be sent through a relay and is set by comparing the relay to other relays with similar bandwidth capabilities.

The unmeasured phase of this exit relay is shown in Figure 3.2, which depicts the data rate and active circuits with a data point every six hours. Throughout the unmeasured phase, the bandwidth consensus weight is set to a constant 160 Kbps. As can be seen from the figure, this leads to a maximum bandwidth utilization of 0.4% of the policy threshold of 24 Mbps.

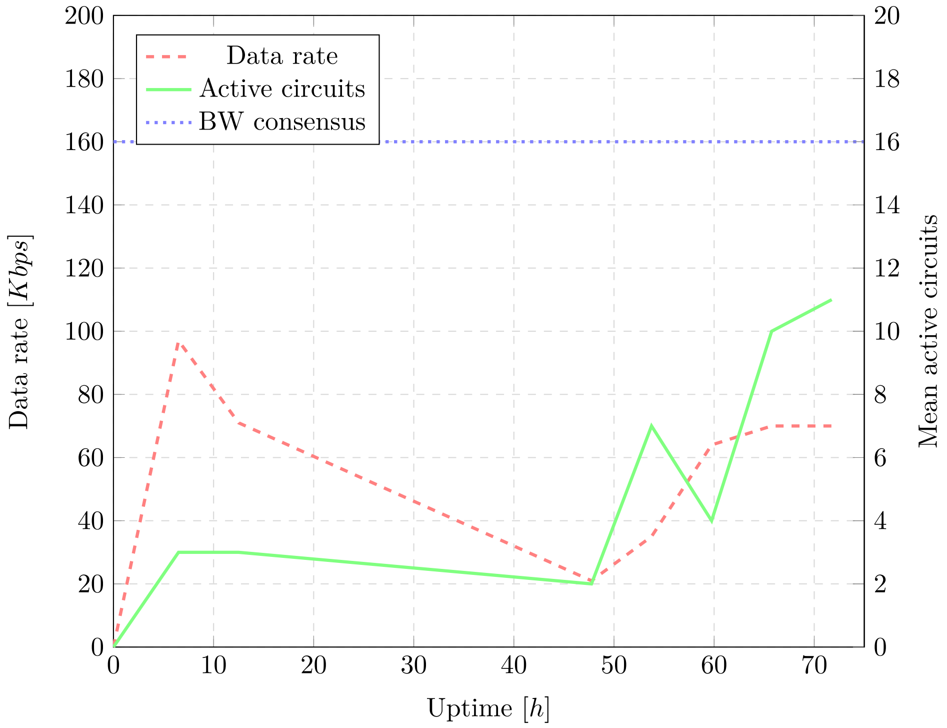


Figure 3.2: Data rate and the number of active circuits for the first 72 hours of operation for the exit relay.

Measured Phase

Following the unmeasured phase, the directory authority servers, with the help of estimations from the bandwidth authority servers, vote on a bandwidth consensus every hour. The bandwidth consensus is set to 1.19 Mbps in hour 74 and rises steadily. The data rate of the exit relay throughout its life is shown in Figure 3.3. Note that the bandwidth consensus in the figure is scaled down by a factor of 5. It can be seen from the figure that the data rate rises steadily, and in step with the bandwidth consensus. Figure 3.3 also shows that the number of active circuits correlates strongly with the data rate. This is expected, as most of the traffic is comprised of HTTP and SSL (see section 3.4.3) and thus it is assumed that most circuits utilize a similar bandwidth.

An interesting observation is that right after the relay is back on-line after the botnet incident, the bandwidth consensus reaches an all time high and the data rate spikes at 14.2 Mbps 12 hours later. 15 hours after the relay is back on-line, the directory authority servers reach a consensus to set the unmeasured flag once again and the data rate drops significantly. This second unmeasured phase lasts for 35 hours and when it ends, the bandwidth consensus, along with the data rate, reaches levels similar to the ones before it was taken down.

If the relay had been kept on-line long enough to reach a steady state, the data rate would likely stabilize around the threshold of 24 Mbps set in the policy, with certain spikes up towards 40 Mbps.

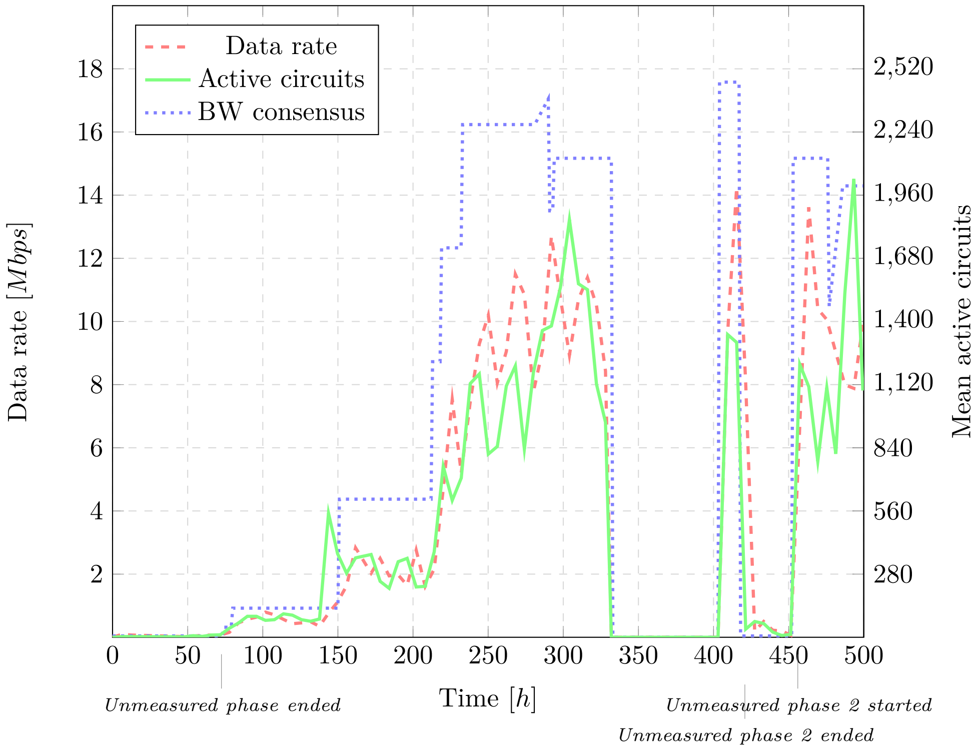


Figure 3.3: Data rate and the number of active circuits for the entirety of the exit relay life. The bandwidth consensus is scaled down by a factor of 5.

3.4.3 Traffic Distribution

All the headers belonging to the exit traffic destined to the Clearnet were recorded throughout the life of the exit relay. This data was used to determine the application distribution which can be seen in Figure 3.4. DNS queries are not included in the traffic distribution analysis. The "Other" portion includes applications such as FTP, IMAP, telnet and other applications the exit policy in Appendix A allows. The exit traffic distribution of a single exit relay, as is shown in Figure 3.4, gives a fairly representative image of the traffic distribution in the Tor Network as a whole.

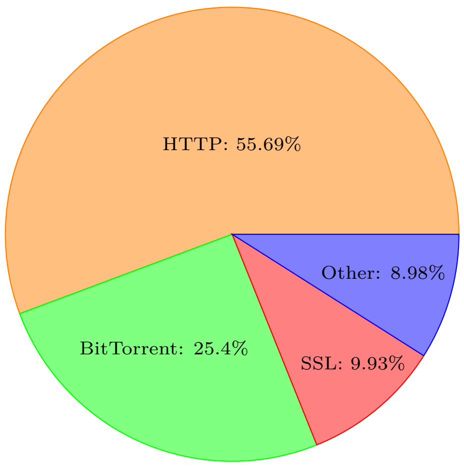


Figure 3.4: Byte-wise application distribution in Tor traffic with exit policy as displayed in Appendix A.

If compared to the publications presented in section 3.1, a further increase in SSL traffic to the 1.55% in the analysis by McCoy et al. [MBG$^+$08] and 5.25% in the analysis by Abdelberi et al. [AMK10] is apparent. It is possible that this increase is the result of a larger portion of the Clearnet implementing SSL in 2015 as opposed to in 2008 and 2010. It is also possible that the Tor community, in combination with the research community, has made Tor users aware of the fact that the possibility of malicious exit relays is present and that end-to-end encryption should be used when applicable to secure the data itself.

When comparing the BitTorrent traffic share, it can seem like the modified Reduced Exit Policy used on this relay has successfully reduced the amount of BitTorrent traffic.

The share of traffic accounted for by BitTorrent is down to 25.4% from 50.39% in the study from 2010, with all ports open [AMK10], and down from 40.2% in the one from 2008, where the Default Exit Policy is used [MBG$^+$08]. However, since these studies are done at different times, the numbers may be affected by the BitTorrent popularity in the period the study was performed.

Sandvine Network Demographics publishes their *Global Internet Phenomena Report* twice a year with detailed information about the traffic share constituted by BitTorrent from 2011 through 2014 [San14]. However, the data used in the reports from 2008 to 2010 provides no specific information about the BitTorrent traffic share, but rather the cumulative P2P file sharing traffic (of which BitTorrent is the leading application). It is clear from Figure 3.5, which shows the numbers from these reports, that the

P2P file sharing traffic share has decreased in the course of recent years. It should be noted, however, that the total traffic volume generated by these applications has not decreased, and that the main reason why this traffic share has decreased is the increase in real-time entertainment applications such as Netflix and Hulu. It is the author's opinion that the BitTorrent traffic share in Tor has not undergone a drastic decrease, mainly because real-time media streaming services do not have the same popularity in Tor as in the Clearnet.

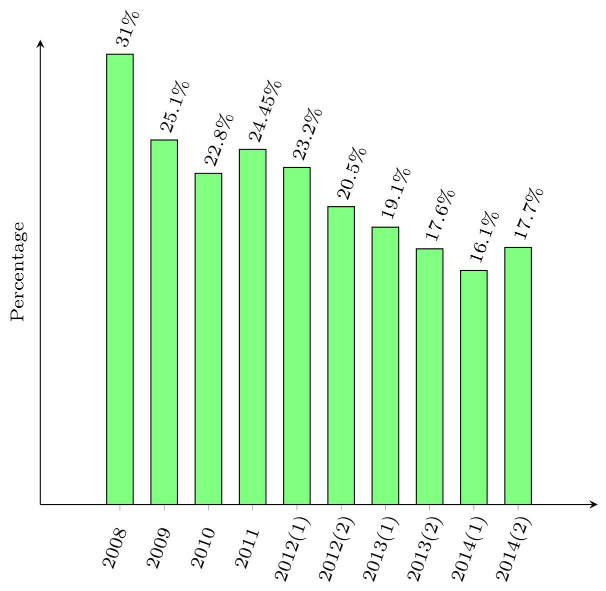


Figure 3.5: Peak period aggregate traffic share of peer-to-peer file sharing applications in fixed access networks [San14].

BitTorrent Traffic Share Relative to the Default Exit Policy

The popularity of P2P file sharing applications in the Clearnet underwent a decrease of 42.9% from 2008 to the second half of 2014. The measured decrease in BitTorrent activity with the Default Exit Policy used in the publication from 2008 [MBG$^+$08] is 36.8%. Thus, an *increase* of 6.1% of BitTorrent traffic relative to the popularity of P2P file sharing is observed. Therefore, it can not be said for certain that the BitTorrent traffic share will experience a significant decrease with the exit policy used for this relay over the Default Exit Policy.

BitTorrent Traffic Share Relative to the All Ports Open Exit Policy

Similarly, the BitTorrent popularity decreased by 10.4% from 2010 to the second half of 2014. Comparing my results with the publication from 2010 [AMK10], the decrease in BitTorrent traffic share is 49.6%. The decrease of BitTorrent traffic relative to the popularity of P2P file sharing is 39.2%, something which suggests that BitTorrent traffic can be reduced by allowing a few select ports to be open in comparison to leaving all the ports open.

The Importance of End-to-End Encryption

Another interesting observation, made from the traffic distribution shown in Figure 3.4, is that, even though it has increased in comparison to earlier studies, the SSL traffic share constitutes a mere 15.2% of the regular web traffic (SSL and HTTP). The newest Global Internet Phenomena Report presents a peak period SSL share of 24.0% [San14], almost 60% higher than what was observed in Tor.

The matter of fact is that if users of Tor refrain from using end-to-end encryption from their client to the web server in the Clearnet, they risk, among other things, leaking the application data to a malicious exit relay. The exit relay in this thesis stored only the data headers, but it would be a trivial task to store all plaintext HTTP application data in addition to this, without this being detected. The HTTP application data may contain sensitive data and, in some cases, data that can be linked directly to the user, such as log-in credentials. The notion that Tor users will log in with their credentials through Tor may seem unlikely, but the DNS queries generated by the exit relay say otherwise. The three large social networks Google+, Facebook and Twitter were responsible for 6.54% of the DNS queries sent by the operated exit relay. As mentioned in section 3.1, the study of HTTP content distribution in 2008 by Abdelberi et al. showed that 9.52% of the HTTP traffic was connected to social networks [AMK10]. Facebook is even reachable through the Onion Network as of 2014 with the URL `https://facebookcorewwwi.onion/`. The three mentioned social networks operate with SSL as a default, but the results show that the users of Tor are willing to use credentials, maybe even without end-to-end encryption in certain cases. Imperceptive users may additionally be the victim of an active attack such as SSL stripping by a malicious exit relay.

As discussed in 2.3.1, a malicious exit relay can also perform other active attacks against regular HTTP traffic such as HTML injection and, if scripts are allowed, make malicious scripts run on the client machine, possibly acquiring information about the location of the client.

3.5 Best Practice for Operating Tor Relays

Operating a Tor exit relay may result in various complaints directed to the operator of the relay. For people who wish to contribute to the Tor Network, but for some reason want to avoid the risk of receiving these complaints, a regular Tor relay that disallows all exit traffic is a good option. No matter what type of Tor relay one operates, the operational security of the relay, including what preventative measures to deploy, should be considered.

If the choice falls on operating an exit relay, the Default Exit Policy is a good choice, but as a measure to reduce abuse and BitTorrent traffic, the author recommends a variant of the Reduced Exit Policy. It is also recommended, as an operator, to inform the relevant instances of the intent to operate an exit relay. Who the relevant instances are, depends on what kind of network the relay is to be deployed on, and for a private home network, the relevant instance is the ISP.

Chapter 4

Website Fingerprinting Attacks in the Tor Network

The website fingerprinting attack has, in recent years, emerged as an interesting method of attempting to identify the destination of encrypted web traffic. The focus of this thesis is mainly website fingerprinting attacks in Tor, but the general attack can be deployed on other anonymization systems as well. Because of its fixed-size traffic units, Tor is one of the most challenging systems to deploy a successful website fingerprinting attack on. Thus, if such an attack can be used in a real-world scenario in the Tor Network, it can likely be used to the same, or even greater, success in most other, low latency, anonymization systems.

Section 4.1 gives a detailed description of the concepts of website fingerprinting attacks. Following this, a summary of important related work on website fingerprinting in Tor and other anonymization systems is given in section 4.2 before a basic website fingerprinting attack is proposed and experimented on in section 4.3.

4.1 What is a Website Fingerprinting Attack?

The website fingerprinting attack is a passive side-channel attack in which the attacker eavesdrops encrypted traffic between the client and destination. In website fingerprinting attacks in Tor, the attacker eavesdrops the traffic on its path from the client to the entry relay, as depicted in Figure 4.1. The purpose of the attack is to predict what web page the victim visits, not what they do while visiting the page, and can be defined as follows:

The goal of a website fingerprinting attack is to determine if a given client is browsing a web page included in a given set of web pages. This set of web pages can include a single web page (i.e. the attacker wants to determine the exact web page that is browsed) or multiple web pages such as a list of pages marked as censored.

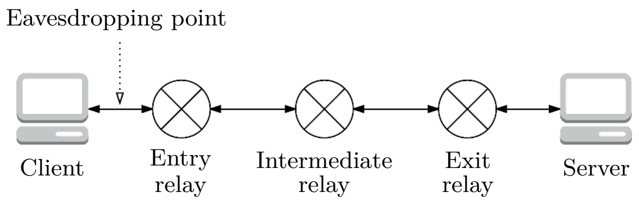


Figure 4.1: The point where the attacker eavesdrops the encrypted data in the website fingerprinting attack on Tor users.

4.1.1 Data Collection

The first phase of a website fingerprinting attack is the data collection phase. In this phase, the attacker chooses a set of web pages, M, to include in the attack model and a set of monitored pages, Φ, where $\Phi \subseteq M$. The attacker subsequently records the traffic generated from a complete page load of each of the web pages in Φ and calculates the web pages' *feature vector*. The feature vector contains features, which is information that can be extracted from the captured traffic, that may be used to uniquely identify a web page. Such information can include the packet size distribution, inter-packet time, the total number of packets, the total time taken to load the web page and more. The process of recording the traffic and calculating the feature vectors must be done at least once for each monitored page, $p \in \Phi$, but may be repeated any number of times to produce more training data to be used in the next phase of the attack.

Because of the dynamic nature of today's web pages, it is important to keep the feature vectors updated to make sure they depict what the web page is like, as close to the time of the classification phase of the attack takes place as possible.

4.1.2 Classification

The second phase of the attack is the phase where the attacker attempts to achieve the attack goal. The feature vectors from the data collection phase are used to train a *classifier*, a supervised learning model which will be used to predict the web page(s) visited by the victim. Any number of feature vectors $|\{\vec{\mathbf{f}}\}| \in \mathbb{Z}^+$ can be used as training data for the classifier.

In supervised learning, the training data is a set of *training examples*, pairs consisting of an input object and an output value [MRT12]. A trained classifier is able to infer a *hypothesis set*, a set of functions mapping feature vectors to the classes [MRT12].

In the domain of website fingerprinting, a training example is comprised of a feature vector, representing the input object, and a web page, representing the output value.

Once the attacker has captured the encrypted traffic of the victim, the individual web page visits are extracted from the traffic and used to create feature vectors. These feature vectors can now be used as input to the hypothesis set which will produce an output that predicts, depending on the attack model, either what web pages the victim has visited or if the victim has visited a web page included in a set of censored pages.

4.1.3 Model Definitions

A differentiation is made between two attack models, the closed world model and the open world model [PNZE11]. Let Ω be the set of every existing web page, M the set of web pages contained in the model, and Φ the set of monitored pages.

Definition 4.1. The closed world model contains a limited set of web pages in which every single page is included in the set of monitored web pages. Each page in the closed world model constitutes a unique class. The input to the hypothesis set is always a feature vector corresponding to one of the monitored pages and the output will be a class representing a single web page.

$$\Phi = M, \quad M \subset \Omega, \quad |M| \ll |\Omega|$$

Definition 4.2. The open world model contains a portion of every existing web page[1], but the set of pages in the model is usually much larger than the set of monitored pages. A subset of the monitored pages are marked as censored and the attack goal is to determine if the victim is browsing one of the pages included in this censored set, denoted Γ. An input to the hypothesis set can be a feature vector corresponding to any page included in the model (not necessarily in the set of monitored pages) and the output class will be either *censored* or *uncensored*.

$$\Phi \subset M, \quad \Gamma \subseteq \Phi, \quad M \simeq \Omega \ ^1$$

Note that the use of these terms varies slightly across related literature (e.g. in publications by Wang et al. where the term *censored pages* is not used and all the pages in the set of monitored pages are marked as censored, i.e. $\Phi = \Gamma$ [WG13, WCN+14]).

4.2 Related Work

Website fingerprinting attacks are not exclusively used on Tor traffic, but rather on encrypted traffic. In 1996, Wagner and Schneier pointed out that traffic analysis

[1] The perfect open world model contains every existing web page, but since this is not feasible in an implementable attack, an approximation is done by using a carefully chosen portion of the existing web pages, e.g. the 1,000 most popular web pages.

could be used to recover confidential information by looking at unencrypted headers and attributes in their analysis of SSL 3.0 [WS96]. The term “website fingerprinting” showed up for the first time in 2001, when Hintz used resource lengths to guess which web page that was visited through the SafeWeb proxy [Hin02].

Hintz’ method of using the resource length worked with HTTP/1.0 where unique TCP connections are opened for each resource, but, with the persistent connections used today, it is hard to differentiate between single resources in HTTP requests. A more fitting feature for HTTP/1.1 is unique packet lengths. In 2006, Bissias et al. used unique packet lengths combined with inter-packet arrival times to mount a website fingerprinting attack with 23% prediction accuracy using a closed world model of size 100 [BLJL05]. Later, Liberatore and Levine extended this work to a more sophisticated attack on a closed world model containing 2000 web pages [LL06]. They built two attacks, one using the naïve Bayes classifier based on the packet sizes and the frequency of each packet size and the other using Jaccard’s coefficient to measure the distance between two sets of packet sequences mapped to its set of unique packet lengths [LL06]. Each of the attacks produced a prediction accuracy of about 75% [LL06].

In 2009, Herrmann et al. published another website fingerprinting attack based on the same principles as Liberatore and Levine’s attack, but with a multinomial naïve Bayes classifier [HWF09]. They used a closed world model with 775 web pages and presented an impressive result of ~95-98% prediction accuracy for all one-hop anonymization systems tested [HWF09]. They also used the same classifier in the Tor Network, but this gave an accuracy of only 2.95%. The main reason why this and earlier website fingerprinting attacks work poorly on Tor traffic is that the packet lengths are obfuscated by having all the data sent in cells of a constant length of 512 bytes. When there is insufficient data available, the cell is padded to the correct length.

Panchenko et al. published a novel attack on Tor traffic in 2011, which, for the first time, included experiments on an open world model in addition to presenting a significantly increased prediction accuracy on a closed world model. The features introduced to achieve this are summarized in Table 4.1. They used Support Vector Machines (SVMs) as classifiers for both the open and closed world models. With a closed world size of 775, they achieved a prediction accuracy of 54.61%. In the open world experiments, a world size of $|M| = 1,000,000$ was used. A subset of $|\Phi| = 5,000$ pages was chosen as the monitored pages and a further subset of $|\Gamma| = 5$ as the censored pages. Experiments done on three different data sets matching these requirements resulted in a True Positive Rate (TPR) of 56.0%, 73.0% and 56.5% and a corresponding False Positive Rate (FPR) of 0.89%, 0.05% and 0.23%.

Feature	**Description**
Exclude packets of size 52	TCP ACK packets with no payload.
Size markers	Marker with the size of the traffic wherever the direction of the flow change.
HTML markers	Marker with the size of the HTML document (first resource requested).
Number markers	Marker with the number of packets wherever the direction of the flow change.
Occurring packet sizes	The number of occurring packet sizes rounded to increments of 2.
Data size	Total transmitted bytes.
Number of packets	The total number of packets.
Percentage incoming packets	The percentage of total packets that are on the downlink.

Table 4.1: Features used in Panchenko's website fingerprinting attack [PNZE11].

In 2012, Cai et al. proposed a new website fingerprinting classifier. First, they converted the packet traces to strings, then they used the Damerau-Levenshtein distance, which is a metric to measure how similar, or dissimilar, two strings are, to determine the similarity between pairs of traces [CZJJ12]. In the process of converting the packet traces to strings, they propose a higher focus on the Tor cell characteristics, i.e. that Tor data is sent in fixed-size cells of 512 bytes each [CZJJ12]. To estimate the number of cells in each TCP Protocol Data Unit (PDU), they round the packet size to a multiple of 512, then divide it by 512. On two closed world models of size 100 and 800, the classifier gave an accuracy of 87.3% and 70% respectively [CZJJ12]. Cai et al. also proposed a web *site* fingerprinting attack using the same classifier. This will be discussed further in chapter 5.

The most recent published attacks are authored by Wang et al. in 2013 and 2014. The first publication achieved an accuracy of 91% in a closed world of size 100, and a TPR of 95% and a FPR of 1.94% in an open world with $|M| = 864$ pages, $|\Phi| = 4$ monitored pages and the same $|\Gamma| = 4$ censored pages [WG13]. The second publication uses a larger open world model with $|M| = 5,000$ pages where $|\Phi| = 100$ are monitored and all the monitored pages are marked as censored (i.e. $\Gamma = \Phi$). This achieved a TPR of 76-85% and a FPR of 0.1-0.6% [WCN$^+$14].

The efficiency of the attacks discussed here is summarized in Table 4.2.

Authors (year)	**System**	**Model**	**Dataset Sizes**	**Accuracy or TPR/FPR**
Hintz (2001)	SSL Proxy	Closed	$\vert M\vert = 5$	23%
Bissias et al. (2006)	SSH Tunnel	Closed	$\vert M\vert = 100$	23%
Liberatore and Levine (2006)	SSH Tunnel	Closed	$\vert M\vert = 2000$	75%
Herrmann et al. (2009)	SSH Tunnel	Closed	$\vert M\vert = 775$	95-98%
Cai et al. (2012)	SSH Tunnel	Closed	$\vert M\vert = 100$	91.6%
Herrmann et al. (2009)	Tor	Closed	$\vert M\vert = 775$	2.75%
Panchenko et al. (2011)	Tor	Closed	$\vert M\vert = 775$	54.61%
Panchenko et al. (2011)	Tor	Open	$\vert M\vert = 1,000,000$ $\vert\Phi\vert = 5,000$ $\vert\Gamma\vert = 5$	56.0/0.89% 73.0/0.05% 56.5/0.23%
Cai et al. (2012)	Tor	Closed	$\vert M\vert = 100$ $\vert M\vert = 800$	83.7% 70%
Wang et al. (2013)	Tor	Closed	$\vert M\vert = 100$	91%
Wang et al. (2013)	Tor	Open	$\vert M\vert = 864$ $\vert\Phi\vert = 4$ $\vert\Gamma\vert = 4$	95.0/1.94%
Wang et al. (2014)	Tor	Open	$\vert M\vert = 5,000$ $\vert\Phi\vert = 100$ $\vert\Gamma\vert = 100$	76-85%/ 0.1-0.6%

Table 4.2: Performance of former website fingerprinting attacks on various one-hop anonymization systems and Tor.

4.3 A Basic Website Fingerprinting Attack

In this section, a basic website fingerprinting attack on Tor traffic is proposed. The attack differs from several of the other attacks presented in section 4.2 in its core feature, namely the way data units are treated. In most previous work, the TCP PDUs are used directly to classify web pages based on features such as size, the number of packets, timing etc. [BLJL05, LL06, HWF09, PNZE11]. The fact that

Tor uses Pushback (a mechanism for controlling the aggregate upstream [MBF$^+$02]) to do rate-limiting, and that the data is transferred in fixed-size 512 byte cells, makes classification by TCP PDUs sub-optimal. In the proposed attack, however, the TLS record headers matching Tor cells are extracted from the data stream and used as the atomic data unit in the calculation of feature vectors.

The goal of the attack presented here is not to surpass the efficiency of previous work, but rather to evaluate the impact of different variables present in a website fingerprinting attack. These variables include the time between gathering the training examples and the traffic to be predicted, the total number of training examples used to train the classifier and which web pages that are included in the model.

This section is structured as follows. First the methodology is described, including the experimental setup, the data collection method, the data processing method and the classification method used in this attack. Following this, experimental results are presented including the impact the various variables mentioned has on the result. The source code for this attack is publicly available from the author's GitHub account [Han15].

4.3.1 Methodology

Experimental Setup

The data collection phase in this website fingerprinting attack is time-consuming. It was thus necessary to automate this. The implementation of the data collection and processing, as well as classification, is done in Python and the relevant components, including important Python modules, are listed in Table 4.3.

To make sure the automated web page requests generated by Selenium WebBrowser were routed through Tor, a custom Firefox profile was made. The profile makes use of a SOCKS proxy running on the local machine, which routes the web traffic through the Tor application.

Component	Description
Tor version	`0.2.5.10`
Operating System	`Linux Ubuntu 14.04.2`
Web browser	`Mozilla Firefox 37.0.1`
Web browsing automation	`Selenium WebDriver`
Traffic capture tool	`Tshark`
Binary packet loader	`pcapy`

Table 4.3: Attack component descriptions.

Data Collection

Algorithm 4.1 defines the data collection method. A closed world model of 100 web pages was constructed by choosing popular web sites that are censored in parts of mainland China. This data set was chosen because it could be used in a real-world scenario where Chinese officials would like the Great Firewall of China to function on Tor traffic as well. Each page was loaded a total of six times, with a time interval of approximately 45 minutes between each load of a given page. To eliminate the risk of traffic from two individual pages interfering with each other, the data collection process is paused for a period of two seconds in between page loads.

The Tor application is set to the default client mode, which signifies building a new circuit at least every 10 minutes. Therefore, in every traffic capture pair from the same page, the two captures are from different circuits. If the same circuit was to be used throughout the data collection phase, the attacker would have unrealistic advantages, such as no redirection based on the location of the exit node (some pages have different content based on what country the request originates from).

Algorithm 4.1 Basic website fingerprinting data collection.

```
Input: $M$                                   ▷ $|M| = 100$
1: for $0 \leq i < 6$ do
2:     for all $p \in M$ do
3:         Load $p$
4:         Pause 2 seconds
5:     end for
6: end for
Output: 6 packet capture files for each of the 100 web pages.
```

Data Processing

Processing the data involves calculating the feature vector based on the captured traffic from the data collection phase. This involves extracting the TLS record headers matching Tor cells from the data stream. The Tor cells are carved from the data stream with a regular expression explained in Figure 4.2. The vector includes two features: The number of Tor cells on the uplink and the number of Tor cells on the downlink.

When compared to earlier publications, this data processing method is most similar to the one proposed by Cai et al., where the number of Tor cells was calculated by rounding the packet sizes to a multiple of 512 and then dividing by 512 [CZJJ12].

$$\underbrace{\texttt{\textbackslash x17}}_{\text{Type}}\underbrace{\texttt{\textbackslash x03[\textbackslash x00\textbackslash x01\textbackslash x02\textbackslash x03]}}_{\text{Version}}\underbrace{\texttt{\textbackslash x02\textbackslash x30}}_{\text{Length}}$$

`Type`
The content type of the TLS record. `\x17` *signifies application type.*

`Version`
The version of TLS used. Matches with SSL 3.0, TLS 1.0, TLS 1.1 and TLS 1.2.

`Length`
The content length. Matches with 560 byte payloads, 512 bytes represents the Tor cell and 48 bytes represents the MAC (Message Authentication Code).

Figure 4.2: Regular expression matching Tor cells.

Classification

The data collection and processing yields 600 examples, $n = 6$ for each page, on which the experiments are applied. In each experiment, a portion of these examples are used as training examples for a classifier while the rest are used as testing examples, i.e. feature vectors to be predicted web pages. To determine the effect the number of training examples, $|T|$, has on the performance of the classifier, experiments are done on classifiers which are trained with a varying number of training examples ranging from one to five.

To minimize the bias of the accuracy and get the most out of the examples, *cross-validation* is applied. Cross-validation is a technique often used when estimating the performance of a machine learning classifier. It works by dividing the examples into k combinations of training examples and testing examples. The process of training and testing a classifier is then repeated k times, with the different combinations of training and testing examples.

In the experiments in this thesis, the exhaustive cross-validation scheme *leave-p-out* is used [Sha93]. Leave-p-out involves using p examples as testing examples and the remaining examples as training examples. In a set of n examples, this is repeated on all combinations of p testing examples and $n - p$ training examples, resulting in $k = \binom{n}{n-p}$ classifiers [Sha93]. The choice fell on the leave-p-out scheme because its only major drawback is that it is computationally costly, but as the pool of examples is relatively small in these experiments, the experiments can still be done reasonably fast.

When using a single training example to train the classifier, i.e. $|T| = 1$, only

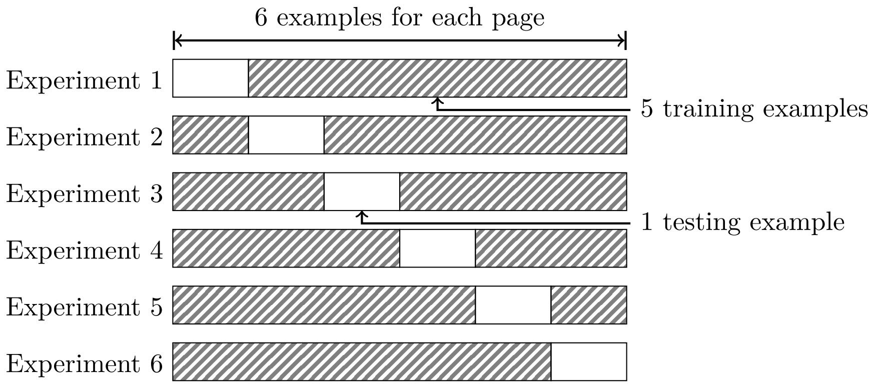


Figure 4.3: Combinations of training and testing examples for the leave-p-out cross validation technique with $p = 1$. Each experiment corresponds to one classifier trained with 5 training examples and tested with $p = 1$ testing example.

$k = \frac{1}{2} \cdot \binom{6}{1} = 3$ classifiers are trained, because when a classifier is trained with A and tested with B, there is no reason to train another classifier with B to test it with A. For other values of $|T| = n - p \in [2, 5]$, $\binom{6}{6-p}$ classifiers are trained. Figure 4.3 illustrates the leave-p-out cross-validation scheme with $|T| = 5$.

Applying the leave-p-out scheme on the $n = 6$ examples available for each page, the following number of classifiers are trained and tested:

$$k = \frac{1}{2} \cdot \binom{6}{1} + \sum_{p=1}^{p=4} \binom{6}{n-p} = 59$$

For all classifiers, the p examples not used as training examples are used as testing examples, i.e. as feature vectors predicted as web pages, $\{\vec{\mathbf{f}}_p\}$. For each value of $|T| = n - p$, this amounts to $p \cdot k$ testing examples. The total number of feature vectors to be predicted as web pages by the various classifiers is thus:

$$|\{\vec{\mathbf{f}}_p\}| = \frac{5}{2} \cdot \binom{6}{1} + \sum_{p=1}^{p=4} p \cdot \binom{6}{6-p} = 171$$

The hypothesis set, which is inferred from training each classifier is a simple one. The feature vectors are considered points in a two-dimensional vector space and the hypothesis set gives the page with the minimum Euclidean distance from the supplied feature vector to the mean feature vector of the training examples for the given page. The *mean* feature vector is acquired by using the mean value of each feature from all training examples corresponding to the given page.

$$H(\vec{\mathbf{f}}_p) = \text{page corresponding to } min\{\|\vec{\mathbf{f}}_p - \vec{\mathbf{f}}_1\|, \|\vec{\mathbf{f}}_p - \vec{\mathbf{f}}_2\|, \ldots, \|\vec{\mathbf{f}}_p - \vec{\mathbf{f}}_{100}\|\}$$

The prediction algorithm for the basic website fingerprinting classifier is given in Algorithm 4.2.

Algorithm 4.2 Single prediction attempt on a feature vector in the basic website fingerprinting attack.

```
Input: $T = \{(\vec{\mathbf{f}}, p)\}$                ▷ Set of training examples.
Input: $\vec{\mathbf{f}}_p$                ▷ Feature vector of the page to be predicted.
1: $C \leftarrow$ Train($T$)                ▷ Gives the mean of all feature vectors for each page.
2: $min \leftarrow \infty$
3: for all $(\vec{\mathbf{f}}, p) \in C$ do
4:     if $|\vec{\mathbf{f}} - \vec{\mathbf{f}}_p| < min$ then
5:         $min \leftarrow |\vec{\mathbf{f}} - \vec{\mathbf{f}}_p|$
6:         $p_{pred} \leftarrow p$
7:     end if
8: end for
Output: $p_{pred}$                ▷ The predicted web page.
```

4.3.2 Experimental Results

The mean accuracy over all the 171 predictions was 21.95%. This signifies that, on average, close to 22 web pages out of the 100 were correctly predicted. Of all the experiments, i.e. classifiers trained and tested, the most successful experiment resulted in a mean accuracy of 57.0%, while the least successful experiment resulted in a mean accuracy of 7.0%. Compared to the accuracy of the previous website fingerprinting attacks on Tor traffic presented in section 4.2, the accuracy in this attack is lower. It is made no claim that this particular website fingerprinting attack yields results that are accurate enough for it to be used in a real-world scenario, but it is still a fairly good result for exclusively using the number of cells as the features in addition to using a trivial classifier.

An observation making this basic website fingerprinting attack more promising is that across all experiments, the number of predictions needed before the correct web page was predicted was reasonably low. In Figure 4.4, the accuracy of the most and least successful experiments for an increasing number of predictions is shown. The data upon which the results presented here is based is shown in a more detailed form in Appendix B.

Based on the results of this attack, the choice was made to use the same source code as a foundation for extending this website fingerprinting attack into a more sophisticated web *site* fingerprinting attack. This extension is presented in chapter 5.

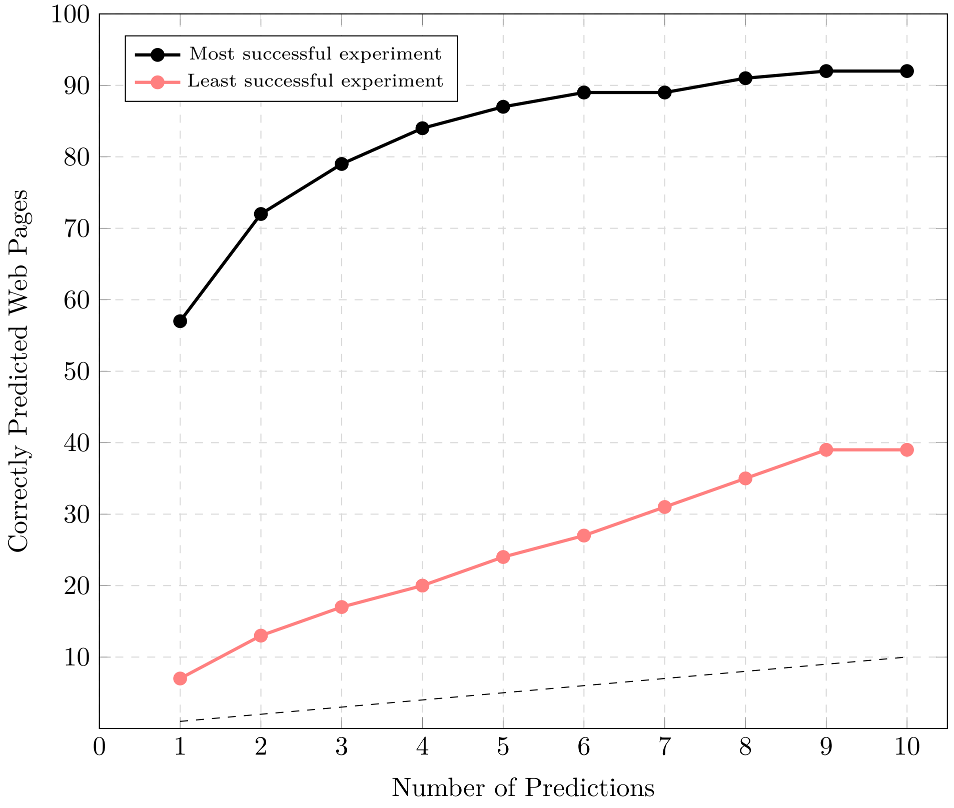


Figure 4.4: Cumulative prediction accuracy for an increasing number of predictions in the most and least successful experiment. The dashed line represents the mean accuracy if the predictions were guessed at random.

The Impact of the Time Interval

The time interval of a single experiment is defined as the time from when the first traffic capture is done to when the last traffic capture is done for any given web page. In the experiments, this ranged from 45 to 225 minutes, depending on which training examples and prediction feature vector that were used. The impact the time interval had on the mean accuracy is shown in Figure 4.5. The figure shows that the accuracy is, significantly, dependent on the time interval. This relationship is credited to the dynamic nature of today's web pages.

Based on this observation, it is clear that one of the important aspects of website fingerprinting is that the training examples must be captured as close to the time that the victim's traffic is captured as possible.

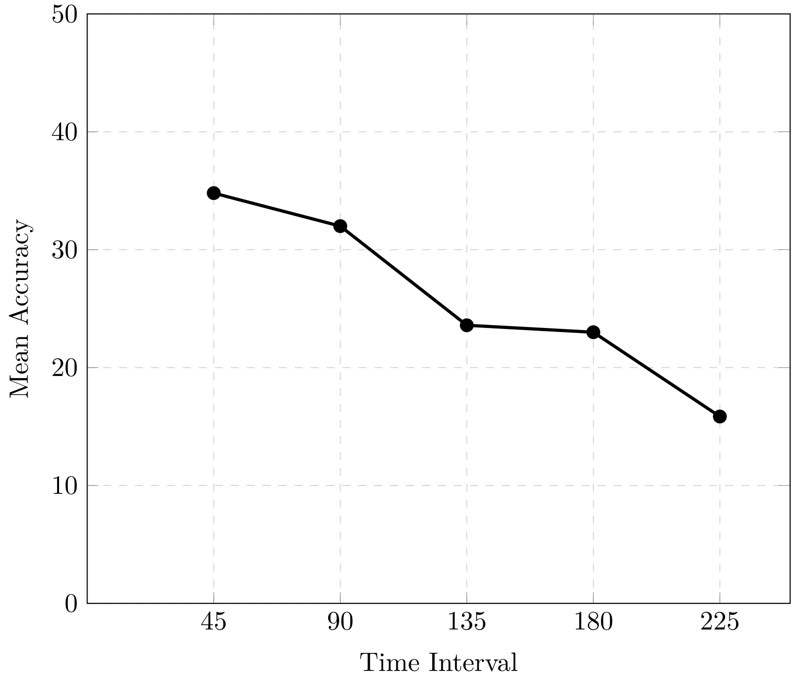


Figure 4.5: The impact of the time interval on prediction accuracy.

The Impact of the Number of Training Examples

The idea of multiple training examples in this attack is to reduce the impact of the changes in the web page caused by time and different localization of the exit relay, by using the average feature vector of multiple visits. However, in classifiers with a higher number of training examples, the mean time interval is higher. The accuracy of the experiments with higher time intervals therefore need to be compensated relative to the impact of the time interval, to give an accurate representation of the impact the number of training examples has on the accuracy.

The compensated accuracy can be found in Table 4.4. The table shows that the number of training examples has little impact on this particular attack, since the standard deviation between the different number of training examples only amounted to 0.58%. This does, however, not signify that the number of training examples *never* has an impact on the result. Chapter 5 presents a more complicated classifier which utilizes feature vectors of higher dimensions, that benefits from using several training examples.

No. Examples	Accuracy
1	34.24
2	35.19
3	34.78
4	35.81
5	34.77

Table 4.4: Mean accuracy compensated for the time interval in the utilized training examples.

The Impact of the Page to be Classified

The accuracy of the attack when classifying the individual web pages varied greatly. Two pages were correctly predicted on the first try in all 171 experiments (`bet365.com` and `bbc.co.uk`) while 27 of the pages were correctly predicted in all 171 experiments by the third prediction. The content of these high-accuracy pages is mostly comprised of search engines and news, content that does not change much in a time period of 225 minutes. On the other hand, some pages were never correctly predicted and the video streaming service `vimeo.com` needed an average of 38 predictions before a correct prediction was made, something which is not much better than guessing at random (i.e. 50 guesses on average). The typical content of web pages that were hard to predict includes video streaming, image services and pages in Chinese.

In the following chapter, this website fingerprinting attack is extended to a web *site* fingerprinting attack based on, not only the number of Tor cells, but also the browsing pattern of the users browsing the web sites.

Chapter 5

A Web *Site* Fingerprinting Attack Based on Browsing Patterns

This chapter proposes a new type of website fingerprinting attack. The attacks discussed in the previous chapter try to identify a single web *page*, not a web *site* which, in most cases, is the most interesting piece of information. If the attacker has access to the encrypted traffic from a visit to a web site, more information than the data from the original loading of a single web page is available. If the victim has browsed the same web site for a certain period, the attacker will often be able to infer additional side-channel information, such as the time between clicks. This is because Tor is a low latency service, and in such a service, the data rate will normally be higher after the user clicks a link. The proposed attack attempts to benefit from this. A proof of concept implementation of the attack is experimented with to provide support for the attack proposal. The source code of the attack presented here, as well as the feature vectors used for the experiments, is publicly available from the author's GitHub account [Han15].

Section 5.1 provides a clarification of the terms website fingerprinting and web *site* fingerprinting in relation to web pages, sites and domains. Following this, related work is summarized in section 5.2. The methodology used is detailed in section 5.3, which describes the experimental setup, the data collection method, the data processing method and the methods of classification in the closed and open world models. The closed and open world experimental results are presented in section 5.4 and section 5.5 respectively, before possible countermeasures are discussed in section 5.6.

5.1 Website Fingerprinting: Page, Site, Domain?

The "website fingerprinting" name can be somewhat confusing in this context. The original purpose of website fingerprinting is to attempt to identify single *pages*, which is why a more fitting name for it would be web *page* fingerprinting. However, the name is mostly agreed upon in the research community, thus no attempt is made to rename it in this thesis. Throughout the thesis, website fingerprinting refers to the method of predicting web *pages* from encrypted traffic, while web *site* fingerprinting refers to the method of predicting web *sites* from encrypted traffic.

A web *site* is defined as the entire collection of linked content under a single domain. For example, consider the domain `example.org`, which includes the following three pages: `example.org/1.html`, `example.org/2.html` and `example.org/3.html`. These are three unique web pages, but they are all part of the `example.org` web *site*.

The attack proposed here is thus a web *site* fingerprinting attack as it tries to identify the site from the encrypted traffic, irrelevant of the sub-pages that are visited within the same domain.

5.2 Related Work

As far as the author can see, there has been no previous research attempting to disclose a web site from encrypted traffic by using the victim's browsing patterns.

Cai et. al proposed a web *site* fingerprinting attack by extending their web *page* classifier [CZJJ12]. Their attack utilized Hidden Markov Models (HMMs) by using the structure of the linked documents on a given site together with the probable paths the user will take when visiting the site. A HMM is a statistical Markov model based on the work of Baum et al. [BP66], in which the transition functions between states are inferred by the training data. In the work of Cai et. al, each web page in the domain of a web site represents a single HMM state [CZJJ12].

Cai et al. made HMMs for two web sites, `facebook.com` and `imdb.com` and constructed a classifier trained with the front pages of the Alexa top 99 web sites in addition to the pages included in the HMMs. This can be considered an open world model of size $|M| = 101$ with $|\Phi| = 101$ monitored sites and $|\Gamma| = 2$ sites marked as censored.

When detecting website visits to Facebook, they achieved a true positive rate of 100% and a false positive rate of 0%. However, the browsing data was artificially made, based directly on the HMM in addition to never renewing the Tor circuit in the data collection phase. Thus, this particular result should, in my opinion, be viewed with some skepticism. The IMDb HMM was made with data from an empirical study and is thus closer to reality. It achieved a true positive rate of 94.4% and a false positive rate of 7.9%.

5.3 Methodology

5.3.1 Experimental Setup

The experimental setup of this attack is mostly the same as the one used for the basic website fingerprinting attack proposed in the previous chapter. Table 4.3 thus serves to describe both attacks.

The setup for the two attacks is, however, different regarding one crucial point: the `pcapy` module, which is used to load binary packet data. By default, the `pcapy` module only supports millisecond accuracy for the timestamps, which is not sufficient for this attack. Therefore, the source code of `pcapy` was modified to facilitate nanosecond accuracy.

5.3.2 Data Collection

A total of six popular web sites of various content were chosen as the monitored sites. These are listed in Table 5.1. Algorithm 5.1 describes how the traffic from browsing a single web site is acquired. Each web site is automatically opened, with no prior notice of which site is going to open. It is then kept open for exactly two minutes before it is automatically closed. In this two-minute interval, the testers are able to browse the web site as they please through a normal Firefox browser window.

`google.com`
`wikipedia.org`
`cbsnews.com`
`nrk.no`
`vimeo.com`
`youtube.com`

Table 5.1: Monitored sites Φ.

The test group responsible for browsing the web sites are comprised of six students at NTNU with both technical and non-technical backgrounds. The testers were asked to browse whatever site appeared as they felt was natural for them. The only requests made of them were that they should attempt to stay within the same domain and elude entering any personal information or user credentials.

A two-minute browsing interval was captured for each of the six monitored web sites a total of six times each. For the open world model experiments, the 109 most popular sites from Alexa top 500 were reduced to 31 sites after removing sites with explicit content, sites that require the user to log in to function normally, sites in other languages than English and sites that block Tor traffic. A single two-minute browsing interval capture was collected for each of these sites, resulting in $|M| = 37$ and $|\Phi| = 6$. Ideally, the open world model would contain more web sites to give a closer representation of how it would have been in a real-world attack, but due to time-consuming data collection, the selection had to be quite limited. When capturing the browsing, all scripts were allowed to make the sites function as close to normal as possible.

Algorithm 5.1 Data collection for a single web site.

```
Input: $s$                              ▷ The web site to collect.
1: Load $s$
2: Capture 120 seconds                  ▷ Let user browse for two minutes.
3: Pause 2 seconds                      ▷ To avoid interference between captures.
Output: $pcap$   ▷ Packet capture file of two minutes worth of browsing site $s$.
```

5.3.3 Data Processing

The extraction of Tor cells from the packet capture byte stream was done in the same fashion as in the basic website fingerprinting attack presented in chapter 4.

The feature vector used in this attack contains six features: the total number of cells on the uplink; the total number of cells on the downlink; the average number of cells on uplink per burst; the average number of cells on downlink per burst; the total number of bursts and the average inter-burst time.

A *burst* is defined as a collection of Tor cells close to each other in the time dimension. The set of extracted Tor cells is divided into bursts by two conditions: the distance in time to the previous and the next burst must be above a supplied *burst tolerance threshold* and the resulting burst must contain at least two Tor cells. Exceptions from these conditions are the first burst, and, if the traffic contains more than a single burst, the last burst. The algorithm for constructing the feature vector from a packet capture file is presented in a simplified version in Algorithm 5.2.

5.3.4 Closed World Classification

The data collection and processing resulted in 36 examples, $n = 6$ for each page. As in the basic website fingerprinting attack, the number of training examples ranges from $|T| \in [1, 5]$ and the leave-p-out cross-validation scheme is applied, resulting in a total of 171 predictions.

When a classifier has been trained, Algorithm 5.3 can be used to predict a web site from a feature vector. The *Vote* method corresponds to the hypothesis set inferred by training the classifier. This set includes a total of five functions, each giving a vote that indicates how well the supplied feature vector matches with each web site. The web site achieving the highest cumulative vote from these five functions will be the output of the hypothesis set (i.e. the predicted web site). The vote functions are further detailed in the following pages.

Algorithm 5.2 Simplified data processing of a single packet capture file.

```
Input: $pcap$                    ▷ Packet capture file of two minutes worth of browsing site $s$.
Input: $b = 2.5$                 ▷ Burst tolerance threshold (in seconds)
 1: $TC \leftarrow$ extractCells($pcap$)        ▷ $TC$ = List of extracted Tor cells.
 2: $\vec{\mathbf{f}}_s \leftarrow [0, 0, 0, 0, 0, 0]$
 3: $\vec{\mathbf{b}}_s \leftarrow []$                                    ▷ Burst vector.
 4: for all $cell \in TC$ do
 5:     if $cell$ on uplink then
 6:         $\vec{\mathbf{f}}_s[0] \leftarrow \vec{\mathbf{f}}_s[0] + 1$
 7:     else
 8:         $\vec{\mathbf{f}}_s[1] \leftarrow \vec{\mathbf{f}}_s[1] + 1$
 9:     end if
10:     if $|ts - cell.ts| > b$ then
11:         $\vec{\mathbf{b}}_s \leftarrow \vec{\mathbf{b}}_s +$ newBurst($cell$)
12:     end if
13:     $ts \leftarrow cell.ts$
14: end for
15: $\vec{\mathbf{b}}_s \leftarrow$ discardBurstsOfSizeOne($\vec{\mathbf{b}}_s$)
16: $\vec{\mathbf{f}}_s[2] \leftarrow$ averageCellsOnUplink($\vec{\mathbf{b}}_s$)
17: $\vec{\mathbf{f}}_s[3] \leftarrow$ averageCellsOnDownlink($\vec{\mathbf{b}}_s$)
18: $\vec{\mathbf{f}}_s[4] \leftarrow |\vec{\mathbf{b}}_s|$                        ▷ Number of bursts.
19: $\vec{\mathbf{f}}_s[5] \leftarrow$ averageInterBurstTime($\vec{\mathbf{b}}_s$)
Output: $\vec{\mathbf{f}}_s$                           ▷ Feature vector for site $s$.
```

Algorithm 5.3 Web site prediction with closed world classifier.

```
Input: $C$                                         ▷ Trained classifier.
Input: $\vec{\mathbf{f}}_p$
1: $s_{pred} \leftarrow$ none                              ▷ Predicted site.
2: $v_{max} \leftarrow -\infty$
3: for all $(\vec{\mathbf{f}}_c, s) \in C$ do
4:     $v \leftarrow$ vote($\vec{\mathbf{f}}_p, \vec{\mathbf{f}}_c$)
5:     if $v > v_{max}$ then
6:         $v_{max} \leftarrow v$
7:         $s_{pred} \leftarrow s$
8:     end if
9: end for
Output: $s_{pred}$                              ▷ The predicted web site.
```

User Inter-Activity

The inter-activity of the web site was measured by attempting to estimate how often an item on the web site is clicked. The function is based on the average time between bursts and the number of bursts which effectively gives the same information. The function gives a vote ranging from -2 for a total mismatch to 8 for a perfect match. The user inter-activity function is given the most weight (i.e. the highest possible vote) of all the functions in the hypothesis set.

Let $\overline{t}_c$ and $\overline{n}_c$ be a given web site contained in the classifier's average time between bursts (in seconds) and average number of bursts respectively, with a standard deviation of σ_t and σ_n. Furthermore, let t_p and n_p be the inter-burst time and number of bursts of the input feature vector, $\vec{\mathbf{f}}_p$. The user inter-activity vote for said site is then calculated as follows:

$$v_{uia} = v_t + v_n$$

$$v_t = \begin{cases} 4.0, & |\overline{t}_c - t_p| = 0, \sigma_t = 0 \\ 1.0, & |\overline{t}_c - t_p| < 4 \\ -1.0, & otherwise \end{cases} \qquad v_n = \begin{cases} 4.0, & |\overline{n}_c - n_p| <= 0.5, \sigma_n <= 0.5 \\ 1.0, & |\overline{n}_c - n_p| < 5 \\ -1.0, & otherwise \end{cases}$$

Uplink-Downlink Cell Ratio

The Tor cell ratio votes give a projection of how much data is uploaded versus how much data is downloaded. The total up-down cell ratio function gives a vote of -1 for a mismatch and 1 for a match. If the web site contained in the classifier's mean ratio is $\overline{r}_c$ and the ratio of the input feature vector is r_p, the vote is calculated as follows:

$$v_r = \begin{cases} 1.0, & |\overline{r}_c - r_p| <= \frac{5}{9} \cdot \overline{r}_c \\ -1.0, & otherwise \end{cases}$$

Uplink-Downlink Cell Ratio Per Burst

Similar to the total cell ratio function, the cell ratio per burst function gives a vote of -1 for a mismatch and 1 for a match. The conditions for a match are slightly different and the vote is calculated as follows:

$$v_{br} = \begin{cases} 1.0, & |\overline{br}_c - br_p| <= \frac{2}{7} \cdot \overline{br}_c \\ -1.0, & otherwise \end{cases}$$

Total Number of Cells

The total number of cells function gives a projection of how much data is exchanged and is based on the basic website fingerprinting hypothesis set presented in the previous chapter. The number of cells uploaded and downloaded is treated as a point in a two-dimensional space and the distances to each of the sites in the classifier are calculated. The distances are ranked from shortest to greatest and the vote is based on this rank. Let $r \in [0, n-1]$ be the rank with $n = |\Phi|$ monitored sites in the classifier. The vote is then calculated with the function:

$$v_d(r, n) = 2 - \frac{2 \cdot r}{n}$$

Number of Cells Per Burst

The number of cells per burst function gives a projection of how much data is exchanged in each burst and is calculated in the same way as the total number of cells.

5.3.5 Open World Classification

The data collection and processing yields six examples of each of the $|\Phi| = 6$ monitored sites and one example of each of the $|M| - |\Phi| = 31$ unmonitored sites, in total 67 examples. Experiments are done with a varying number of sites marked as censored, $|\Gamma| \in [1, 6]$. For each set of censored pages, the number of training examples ranges from $|T| = 1$ to $|T| = 5$ and the leave-p-out cross-validation scheme is applied for each combination of Γ and $|T|$. As all open world website fingerprinting classifiers, the classifiers are trained only with the monitored sites. The total number of classifiers trained is thus:

$$k = 6 \cdot \sum_{p=1}^{p=5} \binom{|\Phi|}{|\Phi| - p} = 378$$

All the examples not used for training a given classifier are used as testing examples, the feature vectors to predict $\vec{\mathbf{f}}_p$, for the classifier. This gives $6 \cdot |\Phi| + |M| - |\Phi| - 6 \cdot p$ for each classifier trained with $|\Phi| - p$ examples. The total number of web sites predicted across all experiments is thus:

$$6 \cdot \sum_{p=1}^{p=5} \binom{|\Phi|}{|\Phi| - p} \cdot (6 \cdot |\Phi| + |M| - |\Phi| - 6 \cdot p) = 18,228$$

The hypothesis set inferred by the open world classifier uses the same vote function as the one used in the closed world. Let v_{max} and Δv be the value of the highest cumulative vote and the difference between the highest and second highest cumulative vote respectively. Furthermore, let s_{vmax} be the web site contained in the classifier that got the highest cumulative vote when matched with the feature vector. A feature vector supplied to the open world hypothesis set will then be classified as a censored site if, and only if, the following requirements are met.

- $s_{vmax} \in \Gamma$
- $v_{max} > v_{threshold}$
- $\Delta v > \Delta v_{threshold}$

The value of these thresholds will be a metric on how sure the hypothesis set must be that the supplied feature vector belongs to a censored site for it to actually classify the feature vector as a censored site. If the thresholds are increased, both the false positive and true positive rates will generally decrease. In the experimental results presented in section 5.5, the thresholds were set to $v_{max} = 7$ and $\Delta v = 1.5$ unless otherwise stated.

5.4 Closed World Experimental Results

Of the 171 predictions done on the closed world model, an average prediction accuracy of 85.58% was achieved over all experiments. The detailed numbers behind the results presented here can be found in Appendix C.1.

5.4.1 Number of Predictions

As presented in Figure 5.1, as much as 97.08% of the predictions were successful by the second prediction. However, a few of the collected feature vectors did not perform well when given as input to the hypothesis set and four guesses were needed to achieve an accuracy of 100%.

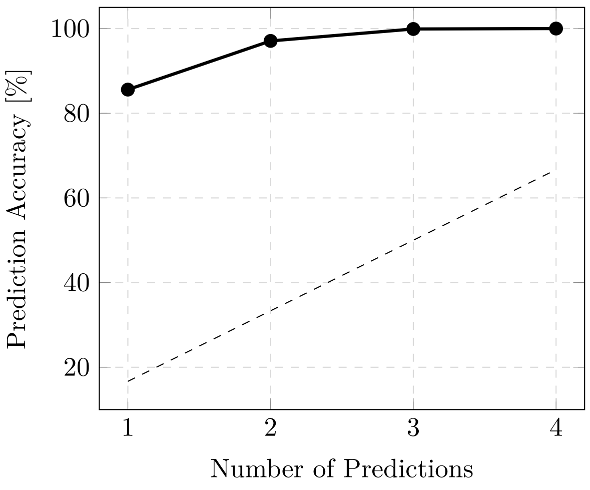


Figure 5.1: Cumulative prediction accuracy for an increasing number of predictions. The dashed line represents the mean accuracy if the guesses were done at random.

5.4.2 Accuracy for Each Site

There was a significant difference in prediction accuracy between the different sites, $s \in \Phi$, as is evident from Figure 5.2. The best performing web site was `vimeo.com`, the exact opposite from the basic website fingerprinting attack where `vimeo.com` needed an average of 38 predictions (see 4.3.2).

The reason why the video streaming sites `vimeo.com` and `youtube.com` perform so well in these experiments is twofold. Firstly, most of the time spent on a site of such content is used to stream videos, resulting in few bursts with a large number of cells in each burst. This separates pure video streaming sites from sites with other content types. Secondly, it is evident from the results that one can relatively easily separate between the two video streaming sites included in the experiment. When analyzing the feature vectors, the two sites sent a similar amount of data, but while `vimeo.com` continuously sent data to the user, resulting in a single large burst, `youtube.com` generally included a few bursts. Thus, the second part of the reason why the video streaming sites in the experiment performed well is credited to the two sites' distinct methods of sending data.

Feature vectors corresponding to `wikipedia.org` gave the lowest prediction accuracy. Since most of the individual pages contained in the domain use the exact same template, the feature vectors are similar when it comes to the average cells per burst. However, the number of bursts and the average inter-burst time, features giving a projection of the user inter-activity, varied greatly. This signifies that the amount of time spent on each page by the users in the test group was highly differentiated.

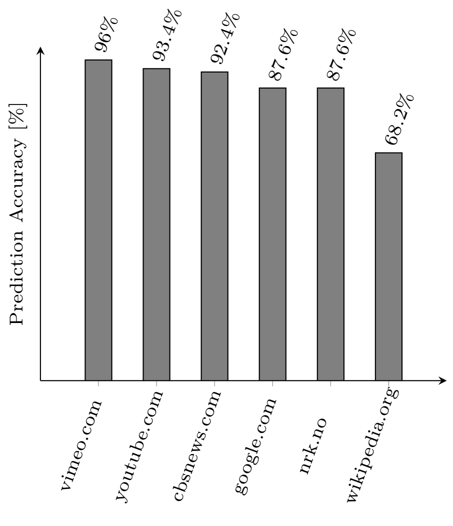


Figure 5.2: Average prediction accuracy for each site $s \in \Phi$.

5.4.3 Varying the Number of Training Examples

The impact that the number of training examples for each site, $|T|$, has on the prediction accuracy is shown in Figure 5.3.

While the impact of the number of training examples in the basic website fingerprinting attack was negligible (see section 4.3.2), it seems to have had an impact in this web *site* fingerprinting attack. In this attack, a higher number of training examples did not only have the ability to reduce the impact of the time interval between capturing the traffic and redirection based on the localization of the exit relay, but also to reduce the impact of one or two web site visits that produced a feature vector deviating from the norm.

It is evident from Figure 5.3 that increasing the number of training examples further than two gives a more or less steady increase in prediction accuracy, and it is expected that the prediction accuracy can be further improved by using even more training examples to train the classifier.

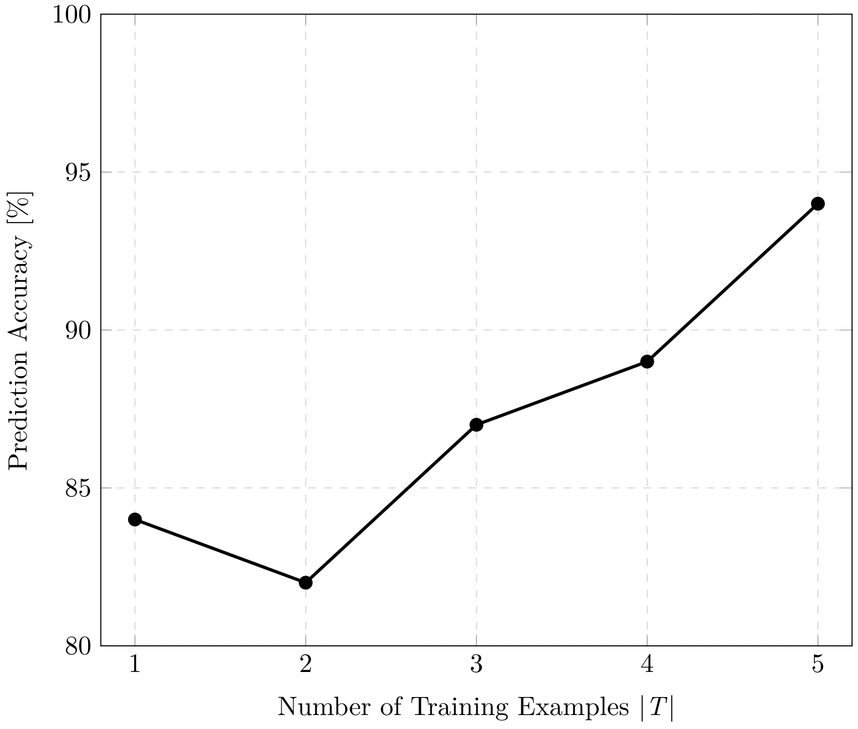


Figure 5.3: Prediction accuracy for an increasing number of training examples.

5.5 Open World Experimental Results

The experiments were done with a number of sites marked as censored ranging from one site to all the monitored sites, a total of six. More sites marked as censored resulted in a higher false positive rate and lower accuracy. Across a total of 18,228 predictions, an average prediction accuracy of 79.97% was achieved. The detailed numbers behind the results presented here can be found in Appendix C.2.

5.5.1 Comparing the Results to Related Work

The web *site* fingerprinting attack proposed by Cai et al., discussed in 5.2, uses an entirely different classification method. Still, since the goal is the same, comparing the results from this attack to theirs can be useful.

Because of the time-consuming data collection phase in this attack, both the model size and the number of training instances are smaller than their equivalent in Cai et al.'s publication. For the comparison, an experiment is run with the two sites `cbsnews.com` and `vimeo.com` marked as censored and a classifier trained by 5 training examples for each monitored site, $s \in \Phi$. The Facebook results from the publication by Cai et al. are not included as they, as attackers, had advantages deemed unrealistic when producing these results (see section 5.2 for details on this). Instead, only the IMDb results are included in the comparison. The comparison, which includes some variables in addition to the true and false positive rates is presented in Table 5.2.

	This Attack	**Cai et al.**
$\|\Gamma\|$	2	2
$\|T\|$	5	25
$\|M\|$	37	101
$\|\Phi\|$	6	101
TPR	0.917	0.944
FPR	0.0095	0.079

Table 5.2: Comparison of variables, true positive rate and false positive rate of this attack and the attack by Cai et al [CZJJ12].

The attack presented here outperforms the attack presented by Cai et al. in terms of FPR, but has a slightly lower performance in terms of TPR. By adjusting the thresholds responsible for deciding if a feature vector should be classified as a censored site or not, it is possible to achieve a TPR and a corresponding Negative Predictive Value (NPV) of 1.0 *or* a FPR and corresponding Positive Predictive Value (PPV) of 0.0 and 1.0. Results from adjusting the threshold values in this experiment are shown in Table 5.3. The choice of setting the threshold values to 1.5 and 7 was made to achieve an acceptable true positive rate, but to keep the false positive rate low.

The false positive rate have an immense impact on whether the attack can be used in a real-world scenario. The importance of keeping the false positive rate low is discussed in section 6.2.

$\Delta v_{threshold}$	$v_{threshold}$	**ACC**	**TPR**	**FPR**	**PPV**	**NPV**
0.8	6	0.978	1.0	0.024	0.706	1.0
1.0	7	0.982	0.923	0.019	0.75	0.995
1.5	7	0.987	0.917	0.0095	0.846	0.995
1.5	8	0.991	0.833	0.0	1.0	0.991

Table 5.3: Results from adjusting the threshold values.

5.5.2 Varying the Number of Marked Sites

The experiments were done with an increasing number of censored sites, Γ, where new sites were included in Γ in the following order: `cbsnews.com`; `vimeo.com`; `youtube.com`; `google.com`; `wikipedia.org`; `nrk.no`.

The performance of the classifier with a varying number of sites marked as censored is shown in Figure 5.4. It is clear from the figure that the classifier performs quite well with a number of censored sites $|\Gamma| \in [1, 3]$. However, it is also evident that keeping the TPR at an acceptable level with more sites marked as censored has a huge impact on the FPR. With all the monitored sites marked as censored (i.e. $\Gamma = \Phi$), the false positive rate reaches 73.2%, a rate so high that the attack will be useless in a real-world scenario.

If the number of sites marked as censored are to be increased in this attack, the thresholds need to be adjusted. Increasing the thresholds will result in a decrease in both the true positive rate and the false positive rate. In these experiments, it is possible to reduce the FPR to a reasonable level through increasing the thresholds, albeit at a big cost to the TPR. Adjusting the thresholds to $\Delta v_{threshold} = 4.5$, $v_{threshold} = 10$ in an experiment with $|\Gamma| = 6$, the classifier was able to achieve a prediction accuracy of 86.04% with a true positive rate of 27.78% and a false positive rate of 2.69%.

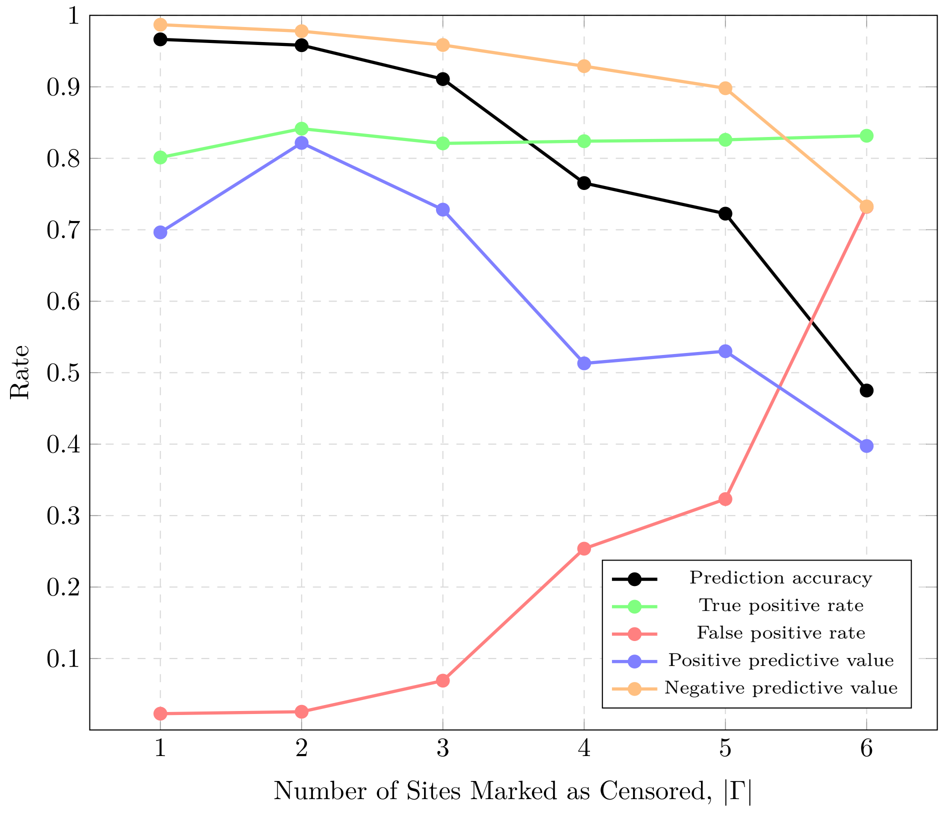


Figure 5.4: Performance of the classifier with an increasing number of sites marked as censored, Γ with thresholds $\Delta v_{threshold} = 1.5$ and $v_{threshold} = 7$.

5.5.3 Varying the Number of Training Examples

Experiments were done with a varying number of training examples, $|T| \in [1, 5]$. Because of the misfit with the thresholds set when $|\Gamma| > 3$, the only results presented here are those of $|\Gamma| \in [1, 3]$. The classifier's performance with an increasing number of training examples for each site is shown in Figure 5.5. The figure shows that the trend where more training examples imply improved performance is present in the open world, as well as in the closed world model (see section 5.4.3).

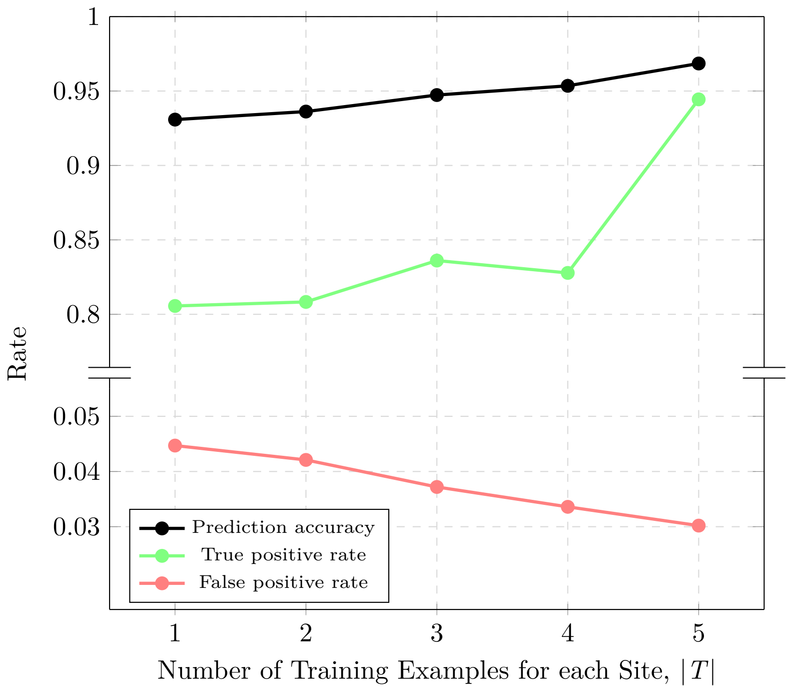


Figure 5.5: Performance of the classifier with a varying number of training examples for each site. Note that the y-scale varies between the false positive rate plot and the others.

5.6 Countermeasures

When a new attack is designed, possible countermeasures should be kept in mind. A brief discussion of viable and not so viable countermeasures is done below. The countermeasures proposed are not tested extensively, but are rather simulated by altering the feature vectors of the web sites experimented on to what they most likely would be if the countermeasures were implemented. The detailed results from the countermeasure simulations can be seen in Appendix D.

5.6.1 Absolute Countermeasure

As the attack presented in this chapter can be considered a side-channel attack, a complete obfuscation of all the side-channel information will render the attack useless. To achieve this, the following additions can be made, either in the source code of Tor or as a layer put on top of Tor.

Constant or Random Inter-Burst Time

Addition: The server *and* the client must send the data in bursts where the inter-burst time either is constant or random and completely independent of the content being sent.

Cost: Depending on how this addition is implemented, it will either slow down the web sites with high data rates considerably or send large amounts of dummy data for the web sites with low data rates.

Result: This addition will obfuscate the user inter-activity.

Constant or Random Data Size

Addition: The server *and* the client must pad the data in each burst to a total number of Tor cells that is either constant or random and completely independent of the original burst size.

Cost: This addition will send large amounts of dummy data in each burst for sites with a relatively low data rate.

Results: This addition will obfuscate all other features of the feature vector used in the web *site* fingerprinting attack.

These countermeasures are infeasible to implement if the low latency requirement in Tor (see section 2.2.2) is to be maintained at the same time as Tor clients are able to browse most existing Clearnet web sites. However, a "light" version of these countermeasures could be introduced to reduce the performance of the attack.

5.6.2 Increasing the Tor Cell Size

If the size of each Tor cell is increased, four out of six features of the feature vector will contain less information, namely the total number of cells on the uplink and downlink and the average number of cells on the uplink and downlink per burst.

This countermeasure was simulated by dividing the number of cells in each burst by an integer. To test the effect of this countermeasure, the same experiment as in section 5.5.1 was run (i.e. with $|\Gamma| = 2$ and $|T| = 5$). The result of this is shown in Figure 5.6 and as can be read from the figure, this countermeasure lowers the performance of the attack, especially in the form of an increased false positive rate (with 512 byte cells, the FPR is 0.95% and with 4092 byte cells, the FPR reaches 19.05%, an increase of over 2000%).

The cost of this countermeasure is relatively small. Depending on what size the Tor cells are set to, it may slightly increase the latency as more data must be buffered at

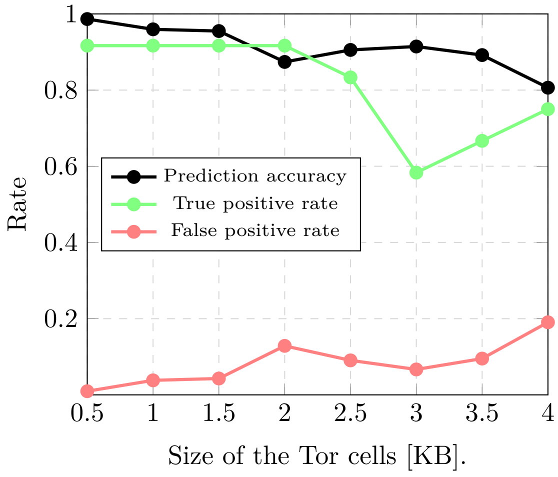


Figure 5.6: Performance of an experiment with $|\Gamma| = 2$ and $|T| = 5$ with an increasing size of the Tor cells.

the exit relay and client before it can be sent. Additionally, when the exit relay does not have enough data to complete a cell, it has to be padded with dummy data, thus introducing a higher overhead.

5.6.3 Obfuscating the User Inter-Activity

The implementation of a countermeasure to partially obfuscate the user inter-activity, while still fulfilling the low latency requirement in Tor, is somewhat more challenging. A possible countermeasure can be to exchange bursts every so often, independent of whether the server or client has anything to send or not. However, these bursts must not be easy to distinguish from the real bursts.

This countermeasure was simulated by editing the feature vector to what it would have been like if data was buffered in the client and server and sent in bursts every *nth* second. If neither the client nor the server has any bursts buffered, a dummy burst consisting of a pseudo-random number of cells within the range of the minimum number of cells across all earlier bursts +/- 10 cells.

In a simulation of this countermeasure where n was set to five, the TPR fell from 91.67% to 75.0% and the FPR increased from 0.95% to 6.19%.

The cost of this countermeasure is not only potentially quite large in terms of higher latency and overhead traffic, but will also open for easier memory based DoS attacks as the exit relay must be able to buffer a maximum of n seconds worth of data for all open circuits.

Chapter 6

Discussion

6.1 Exit Relay and Tor Traffic

O.1: Operate a Tor exit relay

This objective was fulfilled by keeping the exit relay connected to the Tor Network for close to 18 days. As a result of contact with a Botnet, the relay was taken down for a period of three days.

R.1: What applications are responsible for the Tor exit traffic?

In the period the relay was running, in excess of 0.8 terabytes of traffic data went through the relay. Of this traffic, HTTP was responsible for 55.69%, BitTorrent for 25.4%, SSL for 9.93% and other applications for 8.98%. This gives a fairly accurate representation of what applications are responsible for the exit traffic on all relays using a similar exit policy in the Tor Network.

R.2: To what extent can BitTorrent traffic in Tor be minimized while still offering a decent number of services?

Traffic to the ports corresponding to most popular services were allowed to be routed through the exit relay with the exception of a few ports such as the port used for SSH and popular ports used by P2P file sharing applications. The BitTorrent traffic share amounted to 25.4%, a reduction compared to earlier research. However, it is not clear how much of this reduction that can be credited to the exit policy or how much that can be credited the decline in the popularity of P2P file sharing applications.

To be able to answer this question for certain, it is necessary to run several high-bandwidth exit relays in parallel for an extended period of time, each with different exit policies restricting the allowed ports to a varying degree.

6.2 Website Fingerprinting

O.2: Review website fingerprinting

This objective was fulfilled through the review in chapter 4. The most effective website fingerprinting attacks on Tor achieved a prediction accuracy of 83.7-91% in a closed world of size 100 and 70% in a closed world of size 800 [CZJJ12, WG13]. The open world website fingerprinting attacks reviewed achieved a TPR ranging from 56% to 95% with FPRs ranging from 0.05% to 1.94% [PNZE11, WG13, WCN+14]. A basic website fingerprinting attack was also proposed, which achieved a prediction accuracy of up to 57% in a closed world of size 100.

R.3: Are state of the art website fingerprinting attacks feasible in a real-world scenario?

The prediction accuracy of the attacks reviewed seem impressive, but the accuracies presented are done in controlled environments and would most likely decrease if the attacks were to be applied to a real-world scenario.

6.2.1 The Closed World Model

The closed world models used in the research reviewed give interesting theoretical results, but can not easily be extended to a real-world scenario. For the closed world model to work in a real-world scenario, the attacker must include every possible web page the victim might visit in the model, including essentially every existing web page on both the Clearnet and the Onion Network. Assume that the Clearnet contains 1,000,000 web sites (a gross understatement), each containing 100 web pages on average. Furthermore, assume that it takes three seconds on average to load each page through the Tor Network (another understatement). In such a world, the data collection phase alone will take over nine years if done page by page. Even if this process had been highly parallelized, it is infeasible that a classifier can be trained in time for it to be used in an attack.

6.2.2 The Open World Model Size

The open world model research reviewed gives a result that is closer to what it would be in a real-world scenario. However, the size of the experimental data sets used are much smaller than what they would have been in a real-world scenario, where the model will include every existing web page, $M = \Omega$.

Wang et al. used an open world model of size $|M| = 864$ when achieving a TPR of 95% and a FPR of 1.94% and an open world model of size $|M| = 5,000$ when achieving a TPR of 76-85% and a FPR of 0.1-0.6%. These are good results, but because of the limited world size, one can not know how well these attacks would perform in a real-world scenario where $M = \Omega$.

The results by Panchenko et al. is a closer approximation to what would be the case in a real-world scenario as they used an open world model of size $|M| = 1,000,000$. Their classifier achieved a TPR of 56-73% and a FPR of 0.05-0.89%.

6.2.3 False Positives

At first glance, a false positive rate ranging from 0.05% to 0.89% may seem negligible, but this is not the case. To give some insight in the impact false positives can have on an applied website fingerprinting attack, a scenario is constructed. Note that the following scenario is fictional using the performance of the website fingerprinting attack proposed by Panchenko et al. [PNZE11].

Consider Panchenko's website fingerprinting attack used in a real-world scenario to uncover visits to censored pages, $p \in \Gamma$. The following is true for this scenario:

- The open world model, M, contains all existing web pages, i.e. $M = \Omega$.
- A trained classifier, C, is able to predict web pages with the performance as in the publication by Panchenko et al. thus resulting in a TPR of 73.0% and a FPR of 0.05% [PNZE11].
- The loading of 1,000,000 web page visits are recorded, resulting in $|\{\vec{\mathbf{f}}_p\}| = 1,000,000$ feature vectors to be predicted.
- Of the 1,000,000 web page loads, 100 of them corresponds to one of the censored pages $p \in \Gamma$.

The following is defined to be valid throughout the rest of the chapter:

$c =$ Feature vector $\vec{\mathbf{f}}_p$ corresponds to a censored page $p \in \Gamma$

$q =$ Feature vector $\vec{\mathbf{f}}_p$ is predicted as a censored page $p \in \Gamma$ by the classifier C

From the scenario description, the following probabilities are obtained, where c^c denotes the complement of c:

$P(c) = \frac{100}{10^6} = 0.0001, \quad P(c^c) = 1 - P(c) = 0.9999$

$P(q \mid c) = \text{TPR} = 0.73, \quad P(q \mid c^c) = \text{FPR} = 0.0005$

From Bayes' theorem:

$$P(c^c \mid q) = \frac{P(q \mid c^c) \cdot P(c^c)}{P(q)} = \frac{P(q \mid c^c) \cdot P(c^c)}{P(q \mid c^c) \cdot P(c^c) + P(q \mid c) \cdot P(c)}$$
$$= \frac{5 \cdot 10^{-4} \cdot 0.9999}{5 \cdot 10^{-4} \cdot 0.9999 + 0.73 \cdot 10^{-4}} = \underline{0.8726}$$

In this scenario, 73 of the 100 feature vectors corresponding to censored pages are predicted as censored pages on average, which is not bad at all. However, the probability that a feature vector predicted as a censored page does *not* correspond to a censored page is 87.26%. In other words, ~573 feature vectors are predicted as censored pages on average, but 500 of these are false positives.

Under the assumption that visits to censored pages are relatively rare, false positive incidents are more probable than true positive incidents. This kind of result is often referred to as the "false positive paradox" and has been pointed out in many scientific publications, e.g. in Rheinfurth and Howell's "Probability and Statistics in Aerospace Engineering" [RH98].

6.2.4 The Feasibility of Website Fingerprinting Attacks in a Real-World Scenario

From the above critique of website fingerprinting attacks, one may think that such attacks are useless in a real-world scenario. On the contrary, I would argue that state of the art website fingerprinting attacks have realistic applications with today's accuracy. If the attack is to be used to identify Tor users visiting censored pages, a single positive from a single user's captured traffic is not enough to say with confidence that this user has visited a censored page. However, each single positive incident for a given user increases the chance that the user has visited at least one censored page at least once.

Another, fictional scenario is constructed to argue that state of the art website fingerprinting attacks may be applied in a real-world scenario. The following is true for this scenario:

- The open world model, M, contains all existing web pages, i.e. $M = \Omega$.
- A trained classifier, C, is able to predict web pages with the performance as in the publication by Panchenko et al. thus resulting in a TPR of 73.0% and a FPR of 0.05% [PNZE11].
- The Tor traffic of 100,000 Tor users loading 10 web pages each is recorded, resulting in $|\{\vec{\mathbf{f}}_p\}| = 1,000,000$ feature vectors to be predicted.

- 20 of the 100,000 Tor users have visited 5 censored pages $p \in \Gamma$ and 5 uncensored pages $p \notin \Gamma$ each. The remaining 99,980 users have not visited any censored pages.

The following stochastic variables are defined:

$X =$ The user has visited censored pages $p \in \Gamma$

$Y =$ The classifier predicts that the user has visited censored pages $p \in \Gamma$

In this scenario, X is a stochastic variable with six possible values $\in [0, 5]$ and Y is a stochastic variable with eleven possible values $\in [0, 10]$. From the scenario description, the following probabilities are obtained:

$P(c) = \frac{100}{10^6} = 10^{-4}$, $\quad P(c^c) = 1 - P(c) = 0.9999$

$P(q \mid c) = \text{TPR} = 0.73$, $\quad P(q \mid c^c) = \text{FPR} = 5 \cdot 10^{-4}$

$P(X = 0) = \frac{99{,}980}{10^5} = 0.9998$, $\quad P(X > 0) = P(X = 5) = \frac{20}{10^5} = 0.0002$

The probability that the classifier predicts that a user who has visited *no* censored pages, i.e. $X = 0$, has visited exactly n censored pages is:

$$P(Y = n \mid X = 0) = \binom{10}{n} \cdot P(q \mid c^c)^n \cdot (1 - P(q \mid c^c))^{10-n}$$

The probability that the classifier predicts that exactly n of the censored pages $\forall p \in \Gamma$ that a user who has visited censored pages, i.e. $X > 0$, are censored pages is:

$$P(Y = n \mid X > 0 \cap \forall p \in \Gamma) = \binom{5}{n} \cdot P(q \mid c)^n \cdot (1 - P(q \mid c))^{5-n}$$

The probability that the classifier predicts that exactly n of the uncensored pages $\forall p \notin \Gamma$ that a user who has visited censored pages, i.e. $X > 0$, are censored pages is:

$$P(Y = n \mid X > 0 \cap \forall p \notin \Gamma) = \binom{5}{n} \cdot P(q \mid c^c)^n \cdot (1 - P(q \mid c^c))^{5-n}$$

From this, the probability that the classifier predicts that exactly n of the pages that a user who has visited censored pages, $X > 0$, are censored pages is:

$$P(Y = n \mid X > 0) = \begin{cases} \sum\limits_{i=0}^{i=n} P(Y = i \mid X > 0 \cap \forall p \in \Gamma) \cdot P(Y = n - i \mid X > 0 \cap \forall p \notin \Gamma) & n \in [0, 5] \\ \sum\limits_{i=n-5}^{i=5} P(Y = i \mid X > 0 \cap \forall p \in \Gamma) \cdot P(Y = n - i \mid X > 0 \cap \forall p \notin \Gamma) & n \in [6, 10] \end{cases}$$

Finally, the probabilities that the classifier predicts that a given user has visited *at least n* censored pages are:

$$P(Y \geq n \mid X = 0) = \sum_{k=n}^{k=10} P(Y = k \mid X = 0)$$

$$P(Y \geq n \mid X > 0) = \sum_{k=n}^{k=10} P(Y = k \mid X > 0)$$

Now, assume that a given user is suspected of having visited a censored page if, and only if, the classifier predicts that at least two of the pages the user has visited are censored pages. By applying the equations, the following is obtained:

$$\begin{aligned} P(\text{user suspected} \mid X = 0) &= P(Y \geq 2 \mid X = 0) \\ &= \sum_{k=2}^{k=10} \binom{10}{k} \cdot 0.0005^{k} \cdot 0.9995^{10-k} = \underline{1.122 \cdot 10^{-5}} \end{aligned}$$

$$\begin{aligned} P(\text{user suspected} \mid X > 0) &= P(Y \geq 2 \mid X > 0) \\ &= \sum_{k=2}^{k=5} \left(\sum_{i=0}^{i=k} \binom{5}{i} \cdot 0.73^{i} \cdot 0.27^{5-i} \cdot \binom{5}{k-i} \cdot 0.0005^{k-i} \cdot 0.9995^{5-k-i} \right) \\ &+ \sum_{k=6}^{k=10} \left(\sum_{i=k-5}^{i=5} \binom{5}{i} \cdot 0.73^{i} \cdot 0.27^{5-i} \cdot \binom{5}{k-i} \cdot 0.0005^{k-i} \cdot 0.9995^{5-k+i} \right) \\ &= \underline{0.97922} \end{aligned}$$

In other words, the website fingerprinting attack in this fictitious scenario result in an expected number of users rightfully suspected of having visited a censored page $E(\text{users rightfully suspected}) = 19.584$ out of 20. Similarly, the expected number of users who have *not* visited any censored pages, but are falsely suspected of doing so is $E(\text{users wrongfully suspected}) = 1.1213$ out of 99,980.

As a measure to further reduce the chance that an innocent user is suspected of having visited censored pages, a requirement of $Y \geq 3$ can be enforced, resulting in an expected number of users rightfully suspected $E(\text{users rightfully suspected}) = 17.491$ and an expected number of users wrongfully suspected $E(\text{users wrongfully suspected}) = 0.0014958$. Yet another thing that should be considered is that a classifier predicting an uncensored page to be a censored page, may continue to do so for the same, uncensored page. Thus, if a user visits the same, uncensored page repeatedly and all these visits are predicted as a censored page, the user may still be wrongfully suspected of having visited censored pages. By using website fingerprinting alone, one can thus not be truly certain whether a given user has visited a censored page, but it can rather be used as a means to detect suspicious activity.

As have been demonstrated, the chance of a false positive paradox in website fingerprinting can be reduced if it is assumed that users visiting pages marked as censored, $p \in \Gamma$ do so repeatedly. Additionally, users should only be suspected of having visited censored pages if multiple visits to censored pages are predicted by the classifier. Note that it can not be certain whether a website fingerprinting attack scales when every existing web page is included in the model. However, it is assumed that the performance of Panchenko et al.'s classifier gives a good approximation as the 1,000,000 most popular pages used in the experiments [PNZE11] accounts for close to all existing web traffic. Given these assumptions, assumptions that are quite likely valid in the real world, it is feasible that state of the art website fingerprinting attacks can be applied to a real-world scenario.

6.3 Web *Site* Fingerprinting

O.3: Propose a side-channel attack on Tor

This objective was fulfilled with the web *site* fingerprinting attack proposed in chapter 5. A proof of concept implementation of the web *site* fingerprinting attack proposed was used to substantiate the attack with experimental measurements on a limited data set. The average prediction accuracy achieved was 85.58% within a closed world model and 79.97% within an open world model of size 37.

In an experiment where two sites were marked as censored and where the classifier was trained with five training examples for each site (i.e. $|\Gamma| = 2$ and $|T| = 5$), a TPR of 91.7% and a corresponding FPR of 0.95% was achieved.

R.4: Can the browsing pattern of Tor users be used to train a website fingerprinting classifier to identify web sites?

The results from the experiments show promise, but because of the limited data set experimented upon, more research on the area is needed before these methods can be applied to a real-world web *site* fingerprinting attack.

6.3.1 Data Set Sizes

As discussed in section 6.2, the open world model should ideally contain millions of unmonitored sites to give a reasonable approximation to a real-world scenario.

Because of the time-consuming data collection method used, the data sets experimented upon were small. If experiments were done with larger data sets, the TPR is expected to stay more or less the same while the FPR is expected to increase.

6.3.2 Data Collection Method

There are some considerations that are disregarded in the experiments presented in this thesis. These include the difficulty of separating different web site visits in the data stream and interference from multiple, simultaneous web site visits.

Detecting when the data in an encrypted traffic stream changes from belonging to one web site to belonging to another is an unexplored problem. It is, however, possible that similar methods to those used in the proposed attack could be used as a means to do this as well. Features such as the burst pattern deviating strongly from the previous burst pattern can be used as an indicator that the traffic now corresponds to a different web site.

In the proposed attack, the possibility of simultaneous traffic from multiple web sites is not taken into account. The attack can probably take a certain amount of such interference traffic while still achieving decent prediction accuracies for some sites, but identifying multiple sites with intertwined traffic is not possible with this attack. Consider the traffic generated from continuously streaming videos from `vimeo.com` while simultaneously browsing articles on `wikipedia.org`. The classifier presented in this thesis will then identify the traffic as belonging to `vimeo.com` in most cases, but the traffic will never be identified as belonging to `wikipedia.org`.

A major drawback of the web *site* fingerprinting attack proposed in this thesis is the time-consuming data collection. This implies that if the attack is applied in a real-world scenario, both the number of monitored sites, Φ and the number of training examples used to train the classifier for each monitored web site would be limited. The fact that the data collection is time-consuming does not limit the open world size in a real-world scenario (the open world size will, in fact, include all existing web sites in a real-world scenario), but will rather limit the open world size when experimenting on the attack to determine the performance.

Even though the prediction accuracy of the proposed attack would likely be lower if more sites were included in the model, the results are promising and it can be argued that the browsing patterns of Tor users can be used in a web *site* fingerprinting attack. Since the classifier proposed here utilizes features that have not been present in earlier website fingerprinting attacks, there is a possibility that it can be combined with other attacks, such as the one proposed by Cai et al. using HMMs, to further increase the performance.

Chapter 7

Concluding Remarks and Future Work

Through this thesis, I have presented a first-hand experience of operating a Tor exit relay within the network domain of a university campus. The results of the analysis of traffic passing through the relay indicate that that the applications responsible for most of the exit traffic on the relay, running a variant of the reduced exit policy, are HTTP with 55.59% and BitTorrent with 25.4% of the traffic. Furthermore, the analysis indicates that the reduced exit policy results in less BitTorrent traffic in comparison to a policy where all ports are open. Additionally, the reduced exit policy resulted in a smaller BitTorrent traffic share in comparison to a study done in 2008. However, to what extent this reduction is due to the reduction in the popularity of P2P file sharing applications was not determined.

From a review of a collection of website fingerprinting attacks on Tor, the thesis argues that, under the assumption that certain users visit censored pages repeatedly, deploying state of the art website fingerprinting attacks in a real-world scenario is feasible. A typical scenario in which website fingerprinting attacks can be used is by law enforcement to determine, with a certain confidence, whether a given Tor user is visiting web pages of illegal content. Additionally, a basic website fingerprinting attack was proposed, achieving an accuracy of up to 57% using a closed world model of size 100. Through experiments done on the basic website fingerprinting attack, I have demonstrated that the time interval in which the feature vectors are collected has a significant impact on the performance of the attack.

The thesis has also proposed a novel web *site* fingerprinting attack, utilizing not only the number of Tor cells as features, but also the browsing pattern of the users. A limited data amount was gathered and experiments were done on this data through a proof of concept implementation of the proposed attack. In one of the experiments with two sites marked as censored, the attack achieved a TPR of 91.7% and a corresponding FPR of 0.95%. Based on these results, the thesis argues that the users' browsing pattern can be used in a web *site* fingerprinting attack. These results are promising for the feasibility of an extension of a web *site* fingerprinting attack to a real-world scenario in the future, but more research is needed in this area to determine whether web *site* fingerprinting attacks in the Tor Network are feasible.

Tor is a good option for users wanting to use the Internet anonymously. However, no Tor user can be 100 percent certain that their anonymity is completely protected against side-channel attacks.

7.1 Recommendations for Future Work

7.1.1 Traffic Analysis on Exit Relays Running in Parallel with Different Exit Policies

The intention of such an analysis would be to determine, without any ambiguity, the effect that the exit policy has on the application distribution of exit traffic in the Tor Network. It would be particularly interesting to determine the effect the exit policy has on the amount of P2P file sharing traffic.

Several high-bandwidth Tor exit relays should be run in parallel over an extended period of time. The experiment should include at least three exit relays, one with an exit policy allowing all types of traffic, one with the default exit policy and one with a variant of the reduced exit policy.

7.1.2 Website Fingerprinting on Hidden Services

As far as the author can see, there has been no research on whether website fingerprinting is applicable on hidden services or not. The traffic exchanged when Tor users visit hidden services should not have any traits that complicate the website fingerprinting attack further in comparison to traffic from web pages reachable through the Clearnet.

One of the reasons why this has not yet been attempted is probably because the content of many hidden services is illegal. Therefore, simply visiting a page belonging to one of these services would be a crime. On the other hand, applying the website fingerprinting attack on hidden services may be valuable exactly for this reason, as there is a possibility that it can be used as a tool by law enforcement to identify visitors to pages with illegal content.

7.1.3 Web *Site* Fingerprinting Attacks

There are still many unmapped areas of research in the problem domain of web *site* fingerprinting. The most important area is perhaps the model size, aiming to observe how such an attack scales when including more web sites in the open world model.

Another interesting research idea is to combine the features from the attack proposed in this thesis with the HMM concept designed by Cai et al. [CZJJ12] to see if this could provide improvements to the performance of web *site* fingerprinting attacks.

Finally, research on the challenges of web *site* fingerprinting is needed if it were to be used in a real-world scenario. This includes, not only whether such an attack scales well, but also how much interference such an attack can take while still being reasonably effective (e.g. how the attack performs if the traffic from multiple web site visits is intertwined) and to what extent visits to different web sites can be separated in a continuous, encrypted data stream.

7.1.4 Reduce the Efficiency of Side-Channel Attacks in Tor

The recent work on AS-aware Tor clients [SNZ+15] shows promise in regards to the problem of protection against end-to-end correlation attacks in Tor. More work is needed in designing the next generation of anonymity network which is able to resist timing attacks mounted by a global, passive adversary.

In the domain of website fingerprinting, defenses have been proposed [CZJJ12, WCN+14], but for the defenses to significantly reduce the performance of website fingerprinting attacks, a massive bandwidth overhead was necessary. Therefore, I would recommend further research on designing and implementing a Tor client able to reduce the performance of passive, client-side, side-channel attacks such as website fingerprinting aiming to be as efficient as possible in terms of latency and bandwidth increase.

References

[AMK10] Chaabane Abdelberi, Pere Manils, and Mohamed Ali Kâafar. “Digging into Anonymous Traffic: A Deep Analysis of the Tor Anonymizing Network”. In *Proceedings of the Fourth International Conference on Network and System Security (NSS ’10), September 1-3, Melbourne, Victoria, Australia*, pages 167–174, 2010.

[BLJL05] George Dean Bissias, Marc Liberatore, David Jensen, and Brian Neil Levine. “Privacy Vulnerabilities in Encrypted HTTP Streams”. In *Proceedings of the 5th International Workshop on Privacy Enhancing Technologies (PET ’05), May 30-June 1, Cavtat, Croatia, Revised Selected Papers*, pages 1–11, 2005.

[BMG$^+$07] Kevin S. Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas C. Sicker. “Low-Resource Routing Attacks Against Tor”. In *Proceedings of the 2007 ACM Workshop on Privacy in the Electronic Society (WPES ’07), October 29, Alexandria, VA, USA*, pages 11–20, 2007.

[BP66] Leonard E. Baum and Ted Petrie. “Statistical Inference for Probabilistic Functions of Finite State Markov Chains”. *The Annals of Mathematical Statistics*, 37(6):1554–1563, 1966.

[Cha81] David Chaum. “Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms”. *Commun. ACM*, 24(2):84–88, 1981.

[Cot94] Lance Cottrell. “Mixmaster and Remailer Attacks”. http://mixmaster.sourceforge.net, 1994. Accessed: May 2015.

[CZJJ12] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. “Touching from a Distance: Website Fingerprinting Attacks and Defenses”. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS ’12), October 16-18, Raleigh, NC, USA*, pages 605–616, 2012.

[DDM03] George Danezis, Roger Dingledine, and Nick Mathewson. “Mixminion: Design of a Type III Anonymous Remailer Protocol”. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy (S&P ’03), May 11-14, Berkeley, CA, USA*, pages 2–15, 2003.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. “Tor: The Second-Generation Onion Router”. In *Proceedings of the 13th USENIX Security Symposium (USENIX Security ’04), August 9-13, San Diego, CA, USA*, pages 303–320, 2004.

[Dou02] John R. Douceur. “The Sybil Attack”. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS ’02), March 7-8, Cambridge, MA, USA, Revised Papers*, pages 251–260, 2002.

[Gra94] Volker Grassmuck. “Don’t Try to Control the Network Because it’s Impossible Anyway: Interview with Johan Helsingius on Anonymous Remailers”. *IC Magazine, NTT Publishing*, December 1994.

[GRS96] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. “Hiding Routing Information”. In *Proceedings of the 1st ACM International Workshop on Information Hiding (IH ’96), May 30 - June 1, Cambridge, U.K.*, pages 137–150, 1996.

[Han15] Christian August Holm Hansen. “Website Fingerprinting Attacks on Tor Client Anonymity”. GitHub repository https://github.com/chhans/tor-automation, 2015. Accessed: June 2015.

[Hin02] Andrew Hintz. “Fingerprinting Websites Using Traffic Analysis”. In *Proceedings of the 2nd International Workshop on Privacy Enhancing Technologies (PET ’02), April 14-15, San Francisco, CA, USA, Revised Papers*, pages 171–178, 2002.

[HVC10] Nicholas Hopper, Eugene Y. Vasserman, and Eric Chan-Tin. “How Much Anonymity does Network Latency Leak?”. *ACM Trans. Inf. Syst. Secur.*, 13(2), 2010.

[HWF09] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. “Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier”. In *Proceedings of the 1st ACM Cloud Computing Security Workshop (CCSW ’09), November 13, Chicago, IL, USA*, pages 31–42, 2009.

[JTJS14] Rob Jansen, Florian Tschorsch, Aaron Johnson, and Björn Scheuermann. “The Sniper Attack: Anonymously Deanonymizing and Disabling the Tor Network”. In *Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS ’14), February 23-26, San Diego, California, USA*, 2014.

[LL06] Marc Liberatore and Brian Neil Levine. “Inferring the Source of Encrypted HTTP Connections”. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS ’06), October 30 - November 3, Alexandria, VA, USA*, pages 255–263, 2006.

[MBF$^+$02] Ratul Mahajan, Steven M. Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker. “Controlling High Bandwidth Aggregates in the Network”. *Computer Communication Review*, 32(3):62–73, 2002.

[MBG$^+$08] Damon McCoy, Kevin S. Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas C. Sicker. "Shining Light in Dark Places: Understanding the Tor Network". In *Proceedings of the 8th International Symposium on Privacy Enhancing Technologies (PETS '08), July 23-25, Leuven, Belgium*, pages 63–76, 2008.

[MRT12] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *"Foundations of Machine Learning"*. Adaptive computation and machine learning. MIT Press, 2012.

[MZ07] Steven J. Murdoch and Piotr Zielinski. "Sampled Traffic Analysis by Internet-Exchange-Level Adversaries". In *Proceedings of the 7th International Symposium on Privacy Enhancing Technologies (PET '07), June 20-22, Ottawa, Canada, Revised Selected Papers*, pages 167–183, 2007.

[NSA07] NSA. "Tor Stinks", 2007. Presentation given at CT SIGDEV in 2012 (Counter-Terrorism Signals Intelligence Development). Made available through the NSA document leaks by Edward Snowden in October 2013. https://edwardsnowden.com/wp-content/uploads/2013/10/tor-stinks-presentation.pdf. Accessed: May 2015.

[PNZE11] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. "Website Fingerprinting in Onion Routing Based Anonymization Networks". In *Proceedings of the 10th annual ACM workshop on Privacy in the Electronic Society (WPES '11), October 17, Chicago, IL, USA*, pages 103–114, 2011.

[PYFZ08] Ryan Pries, Wei Yu, Xinwen Fu, and Wei Zhao. "A New Replay Attack Against Anonymous Communication Networks". In *Proceedings of IEEE International Conference on Communications (ICC '08), May 19-23, Beijing, China*, pages 1578–1582, 2008.

[RH98] MH Rheinfurth and Leonard W Howell. "Probability and Statistics in Aerospace Engineering". *Marshall Space Flight Center, Alabama, NASA/TP-1998-20719*, 1998.

[San14] Sandvine Network Demographics. "Global Internet Phenomena Report". Technical report, 2008-2014. Known as the Global Broadband Phenomena in the years 2008-2010 where it was published once a year. Published twice a year from 2011-2014.

[Sha93] Jun Shao. "Linear Model Selection by Cross-Validation". *Journal of the American Statistical Association*, 88(422):486–494, 1993.

[SNZ$^+$15] Oleksii Starov, Rishab Nithyanand, Adva Zair, Phillipa Gill, and Michael Schapira. "Measuring and Mitigating AS-Level Adversaries Against Tor". arXiv:1505.05173v4 [cs.CR], June 2015.

[WCN$^+$14] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. "Effective Attacks and Provable Defenses for Website Fingerprinting". In *Proceedings of the 23rd USENIX Security Symposium (USENIX Security '14), August 20-22, San Diego, CA, USA*, pages 143–157, 2014.

[WG13] Tao Wang and Ian Goldberg. “Improved Website Fingerprinting on Tor”. In *Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society (WPES ’13), November 4, Berlin, Germany*, pages 201–212, 2013.

[WKM$^+$14] Philipp Winter, Richard Köwer, Martin Mulazzani, Markus Huber, Sebastian Schrittwieser, Stefan Lindskog, and Edgar R. Weippl. “Spoiled Onions: Exposing Malicious Tor Exit Relays”. In *Proceedings of the 14th International Security Symposium on Privacy Enhancing Technologies (PETS ’14), July 16-18, Amsterdam, The Netherlands*, pages 304–331, 2014.

[WS96] David Wagner and Bruce Schneier. “Analysis of the SSL 3.0 protocol”. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce (EC ’96), November 18-21, Oakland, CA, USA*, pages 29–40, 1996.

Appendix A

Relay Policy

Below is the contents of the Tor configuration file. This describes, among other things, the ports used for incoming traffic, the bandwidth limit and the exit policy. The bandwidth limit is set to 24 Mbps, but bursts up to 40 Mbps are allowed. Furthermore, some IP addresses are blocked in the exit policy to disallow access to the campus intranet and reserved addresses. The exit policy is adapted from the *reduced exit policy* recommended by the Tor Project. Changes include, among other things, the blocking of port 22 to avoid SSH bruteforcing, something which can be the source of abuse complaints.

```
ORPort 9001
DirPort 80
DirPortFrontPage /home/exitrelayuser/chroot/src/index.html
Nickname torntnu
ContactInfo chhans@stud.ntnu.no
RelayBandwidthRate 3 MBytes      # 24 Mbps
RelayBandwidthBurst 5 MBytes     # 40 Mbps

# Blocked IP addresses in exit policy
ExitPolicy reject 129.241.0.0/16:*  # Intranet
ExitPolicy reject 192.168.0.0/16:*  # Reserved for private nets
ExitPolicy reject 10.0.0.0/8:*      # Reserved for private nets
ExitPolicy reject 172.16.0.0/12:*   # Reserved for private nets
ExitPolicy reject 0.0.0.0/8:*       # Reserved self-identification
ExitPolicy reject 169.254.0.0/16:*  # Invalid IP from DHCP

# Accepted ports in exit policy
ExitPolicy accept *:21          # FTP
ExitPolicy accept *:23          # telnet
ExitPolicy accept *:43          # WHOIS
ExitPolicy accept *:53          # DNS
ExitPolicy accept *:79-81       # finger, HTTP
ExitPolicy accept *:88          # kerberos
ExitPolicy accept *:110         # POP3
ExitPolicy accept *:143         # IMAP
ExitPolicy accept *:194         # IRC
ExitPolicy accept *:220         # IMAP3
ExitPolicy accept *:389          # LDAP
ExitPolicy accept *:443          # HTTPS
ExitPolicy accept *:464          # kpasswd
ExitPolicy accept *:531          # IRC/AIM
ExitPolicy accept *:543-544      # Kerberos
ExitPolicy accept *:554          # RTSP
ExitPolicy accept *:563          # NNTP over SSL
ExitPolicy accept *:636          # LDAP over SSL
ExitPolicy accept *:706          # SILC
ExitPolicy accept *:749          # kerberos
ExitPolicy accept *:873          # rsync
ExitPolicy accept *:902-904      # VMware
ExitPolicy accept *:981          # Remote HTTPS firewall management
ExitPolicy accept *:989-995      # Various protocols over SSL
ExitPolicy accept *:1194         # OpenVPN
ExitPolicy accept *:1220         # QT Server Admin
ExitPolicy accept *:1293         # PKT-KRB-IPSec
ExitPolicy accept *:1500         # VLSI License Manager
ExitPolicy accept *:1533         # Sametime
ExitPolicy accept *:1677         # GroupWise
ExitPolicy accept *:1723         # PPTP
ExitPolicy accept *:1755         # RTSP
ExitPolicy accept *:1863         # MSNP
ExitPolicy accept *:2082         # Infowave Mobility Server
ExitPolicy accept *:2083         # Secure Radius Service (radsec)
ExitPolicy accept *:2086-2087    # GNUnet, ELI
ExitPolicy accept *:2095-2096    # NBX
ExitPolicy accept *:2102-2104    # Zephyr
ExitPolicy accept *:3128         # SQUID
ExitPolicy accept *:3389         # MS WBT
ExitPolicy accept *:3690         # SVN
ExitPolicy accept *:4321         # RWHOIS
ExitPolicy accept *:4643         # Virtuozzo
ExitPolicy accept *:5050         # MMCC
ExitPolicy accept *:5190         # ICQ
ExitPolicy accept *:5222-5223    # XMPP, XMPP over SSL
ExitPolicy accept *:5228         # Android Market
ExitPolicy accept *:5900         # VNC
ExitPolicy accept *:6660-6669    # IRC
ExitPolicy accept *:6679         # IRC SSL
ExitPolicy accept *:6697         # IRC SSL
ExitPolicy accept *:8000         # iRDMI
ExitPolicy accept *:8008         # HTTP alternate
ExitPolicy accept *:8074         # Gadu-Gadu
ExitPolicy accept *:8080         # HTTP Proxies
ExitPolicy accept *:8082         # HTTPS Electrum Bitcoin port
ExitPolicy accept *:8087-8088    # Simplify Media SPP Protocol
ExitPolicy accept *:8332-8333    # Bitcoin
ExitPolicy accept *:8443         # PCsync HTTPS
ExitPolicy accept *:8888         # HTTP Proxies, NewsEDGE
ExitPolicy accept *:9418         # git
ExitPolicy accept *:9999-10000   # NDMP
ExitPolicy accept *:11371        # OpenPGP hkp
ExitPolicy accept *:19294        # Google Voice TCP
ExitPolicy accept *:19638        # Ensim control panel
ExitPolicy accept *:50002        # Electrum Bitcoin SSL
ExitPolicy accept *:64738        # Mumble

# Reject all other ports
ExitPolicy reject *:*
```

Appendix B

Basic Website Fingerprinting Results

Let the column headers signify their respective definition below. The results for the basic website fingerprinting attack is then given in Table B.1, B.2 and B.3.

- $\mathbf{\Delta t}$: the maximum time interval between the capture time of any two packet capture files used in the experiment.
- **n**: the number of predictions made in the experiment.
- %: the mean accuracy over the predictions in the experiment.
- σ: the standard deviation of the prediction accuracy in the experiment.
- $|T|$: the number of training examples used for each page in the experiment.
- **c**: the compensation factor for the prediction accuracy of experiments done with a varying number of training examples based on $\mathbf{\Delta t}$.
- $\%_c$: the compensated mean accuracy over the predictions in the experiment.

Δ**t**	**n**	%	σ
45	5	34.80	15.94
90	16	32.00	14.27
135	33	23.59	9.36
180	54	23.00	10.52
225	63	15.85	5.16

Table B.1: Results based on the time interval.

$\|T\|$	**n**	%	σ
1	15	29.33	14.78
2	60	23.82	11.71
3	60	20.37	8.54
4	30	18.93	6.90
5	6	15.83	4.67

Table B.2: Results based on the number of training examples.

Δ**t**	$\|T\|$	**n**	%	**c**	$\%_c$
45	1	5	34.80	1.00	34.80
90	1	4	34.00	1.09	36.98
135	1	3	24.00	1.48	35.40
180	1	2	22.00	1.51	33.28
225	1	1	14.00	2.20	30.75
90	2	12	30.00	1.09	32.63
135	2	18	24.28	1.48	35.81
180	2	18	24.33	1.51	36.81
225	2	12	16.17	2.20	35.51
135	3	12	22.50	1.48	33.19
180	3	24	23.38	1.51	35.36
225	3	24	16.29	2.20	35.78
180	4	10	22.30	1.51	33.74
225	4	20	17.25	2.20	37.88
225	5	6	15.83	2.20	34.77

Table B.3: Results of the experiments with a varying number of training examples compensated for the time-interval.

Appendix C

Web *Site* Fingerprinting Results

C.1 Results with the Closed World Model

The average prediction accuracy for each web site is presented in Table C.1, the cumulative prediction accuracy for a different number of guesses is presented in C.2 and the prediction accuracy for a different number of training examples is presented in C.3.

Site	**Accuracy**
cbsnews.com	92.4%
wikipedia.org	68.2%
youtube.com	93.4%
google.com	87.6%
vimeo.com	96.0%
nrk.no	87.6%

Table C.1: Prediction accuracy for each site $s \in \Phi$.

Guesses	**Accuracy**
1	85.58%
2	97.08%
3	98.93%
4	99.90%
5	100.0%

Table C.2: Cumulative prediction accuracy for an increasing number of guesses.

Training examples	**Predictions**	**Accuracy**
5	6	94.0%
4	30	89.0%
3	60	87.0%
2	60	82.0%
1	15	84.0%
	Total:	85.58%

Table C.3: Prediction accuracy for a different number of training examples.

C.2 Results with the Open World Model

Let the column headers of Table C.5 signify their respective definitions in Table C.4. The results for the pattern-based web *site* fingerprinting attack is then given in Table C.5.

Header	**Formula**	**Description**
TP	-	The number of true positives.
FP	-	The number of false positives.
TN	-	The number of true negatives.
FN	-	The number of false negatives.
ACC	$\frac{\mathbf{TP}+\mathbf{TN}}{\mathbf{TP}+\mathbf{TN}+\mathbf{FP}+\mathbf{FN}}$	The prediction accuracy.
TPR	$\frac{\mathbf{TP}}{\mathbf{TP}+\mathbf{FN}}$	The true positive rate.
FPR	$\frac{\mathbf{FP}}{\mathbf{FP}+\mathbf{TN}}$	The false positive rate.
PPV	$\frac{\mathbf{TP}}{\mathbf{TP}+\mathbf{FP}}$	The positive predictive value.
NPV	$\frac{\mathbf{TN}}{\mathbf{TN}+\mathbf{FN}}$	The negative predictive value.

Table C.4: Description of abbreviations in Table C.5 and their corresponding formulas.

$\|\Gamma\|$	$\|T\|$	**TP**	**FP**	**TN**	**FN**	**ACC**	**TPR**	**FPR**	**PPV**	**NPV**
1	1	22	11	325	8	0.9481	0.7333	0.0327	0.6667	0.9760
1	2	47	22	743	13	0.9576	0.7833	0.0288	0.6812	0.9828
1	3	50	20	900	10	0.9694	0.8333	0.0217	0.7143	0.9890
1	4	24	10	605	6	0.9752	0.8000	0.0163	0.7059	0.9902
1	5	6	2	214	0	0.9910	1.0000	0.0093	0.7500	1.0000
2	1	50	11	295	10	0.9426	0.8333	0.0359	0.8197	0.9672
2	2	101	22	683	19	0.9503	0.8417	0.0312	0.8211	0.9729
2	3	101	21	839	19	0.9592	0.8417	0.0244	0.8279	0.9779
2	4	50	12	573	10	0.9659	0.8333	0.0205	0.8065	0.9828
2	5	11	2	208	1	0.9865	0.9167	0.0095	0.8462	0.9952
3	1	73	19	257	17	0.9016	0.8111	0.0688	0.7935	0.9380
3	2	143	45	600	37	0.9006	0.7944	0.0698	0.7606	0.9419
3	3	150	55	745	30	0.9133	0.8333	0.0688	0.7317	0.9613
3	4	75	37	518	15	0.9194	0.8333	0.0667	0.6696	0.9719
3	5	17	15	189	1	0.9279	0.9444	0.0735	0.5313	0.9947
4	1	93	72	174	27	0.7295	0.7750	0.2927	0.5636	0.8657
4	2	192	154	431	48	0.7552	0.8000	0.2632	0.5549	0.8998
4	3	205	189	551	35	0.7714	0.8542	0.2554	0.5203	0.9403
4	4	101	122	403	19	0.7814	0.8417	0.2324	0.4529	0.9550
4	5	22	45	153	2	0.7883	0.9167	0.2273	0.3284	0.9871
5	1	112	78	138	38	0.6831	0.7467	0.3611	0.5895	0.7841
5	2	241	173	352	59	0.7552	0.8000	0.2632	0.5549	0.8998
5	3	258	225	455	42	0.7276	0.8600	0.3309	0.5342	0.9155
5	4	129	148	347	21	0.7380	0.8600	0.2990	0.4657	0.9429
5	5	28	57	135	2	0.7342	0.9333	0.2969	0.3294	0.9854
6	1	139	122	64	41	0.5546	0.7722	0.6559	0.5326	0.6095
6	2	291	334	131	69	0.7188	0.8033	0.3295	0.5821	0.8564
6	3	311	472	148	49	0.4684	0.8639	0.7613	0.3972	0.7513
6	4	154	342	123	26	0.4295	0.8556	0.7355	0.3105	0.8255
6	5	33	137	49	3	0.3694	0.9167	0.7366	0.1941	0.9423
Total:		3229	2974	11348	677	0.7997	0.8267	0.2077	0.5206	0.9437

Table C.5: Results of browsing-pattern-based web *site* fingerprinting attack with the open world model.

Appendix D

Web *Site* Fingerprinting Countermeasure Simulation Results

The results from simulating the countermeasures are presented in Table D.1. The top half of the table corresponds to the countermeasure where the Tor cell size is increased and the bottom half corresponds to the countermeasure where the exit relay and client buffers data to exchange it evrey *nth* second.

Cell Size [B]	**ACC**	**TPR**	**FPR**
512	0.9865	0.9167	0.0095
1024	0.9595	0.9167	0.0381
1536	0.9550	0.9167	0.0429
2048	0.8739	0.9167	0.1286
2560	0.9054	0.8333	0.0905
3072	0.9144	0.5833	0.0667
3584	0.8919	0.6667	0.0952
4096	0.8063	0.7500	0.1905
n	**ACC**	**TPR**	**FPR**
2	0.9279	0.6667	0.0571
3	0.9234	0.6667	0.0619
4	0.9234	0.75	0.0667
5	0.9279	0.75	0.0619

Table D.1: Results of simulating countermeasures.