Measuring and circumventing Internet censorship

Philipp Winter

Karlstad University

Department of Mathematics
and Computer Science

Thesis submitted for the degree
of Doctor of Philosophy

November 2014

Widmung

Diese Arbeit ist meinen Eltern, Daniela und Erwin, gewidmet. Ohne ihre Unterstützung wäre all dies nicht möglich gewesen.

Dedication

This thesis is dedicated to my parents, Daniela and Erwin. Without their support, all this would not have been possible.

Abstract

An ever increasing amount of governments, organisations, and companies employ Internet censorship in order to filter the free flow of information. These efforts are supported by an equally increasing number of companies focusing on the development of filtering equipment. Only what these entities consider right can pass the filters. This practice constitutes a violation of the Universal Declaration of Human Rights and hampers progress.

This thesis contributes novel techniques to measure and to circumvent Internet censorship. In particular, we 1) analyse how the Great Firewall of China is blocking the Tor network by using active probing techniques as well as side channel measurements, we 2) propose a concept to involve users in the process of censorship analysis, we 3) discuss the aptitude of a globally-deployed network measurement platform for censorship analysis, and we 4) propose a novel circumvention protocol. We attach particular importance to practicality and usability. Most of the techniques proposed in this thesis were implemented and some of them were deployed and are used on a daily basis.

We demonstrate that the measurement techniques proposed in this thesis are practical and useful by applying them in order to shed light on previously undocumented cases of Internet censorship. We employed our techniques in three countries and were able to expose previously unknown censorship techniques and cooperation between a corporation and a government for the sake of censorship. We also implemented a circumvention protocol which was subsequently deployed and is used to evade the Great Firewall of China.

Acknowledgements

First, I want to thank my advisers Stefan Lindskog and Simone Fischer-Hübner for their support, guidance, and patience over the last three years. I am aware that many of my decisions and requests made their job much harder but they never complained.

Next, I want to thank my collaborators Collin Anderson, Jed Crandall, Martin Drašar, Roya Ensafi, Jürgen Fuß, Markus Huber, Richard Köwer, Stefan Lindskog, Abdullah Mueen, Martin Mulazzani, Tobias Pulls, Sebastian Schrittwieser, Jan Vykopal, and Edgar Weippl. I also want to thank the Citizen Lab at the University of Toronto and Jed Crandall's group at the University of New Mexico for having me as a visiting researcher.

I also want to thank my friends and family for their moral support; in particular, all members of the PriSec research group, Harald Lampesberger, and Roya Ensafi. I am also grateful to all members of the Tor Project whose almost boundless passion and dedication served as great inspiration to me.

Finally, I want to thank Internetfonden whose generous research grants funded a significant part of my research over the last three years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In 2012, Reporters Without Borders published a report which identifies twelve "enemies of the Internet" [1]. These twelve enemies are in fact countries; namely Burma, China, Cuba, Iran, North Korea, Saudi Arabia, Syria, Turkmenistan, Uzbekistan, and Vietnam. What these countries have in common is their tight grip on national communication infrastructure. The report mentions Bahrain's arrest of bloggers, Iran's launch of its "national Internet" and Uzbekistan's Internet monitoring, just to name a few examples. Furthermore, the report identifies 14 countries under surveillance. This list also contains Australia and France which shows that surveillance is a global phenomena which also includes the Western world. This fact is further supported by recent revelations concerning a large Western intelligence agency [2].

The difference between Internet censorship and surveillance appears significant from a policy point of view. Censorship, which is frequently conducted by repressive regimes and dictatorships, is typically associated with web sites blocks, arrests and harassment of bloggers, and the deletion of regime-critical content. Surveillance, on the other hand, is often seen as a necessary part of democratic countries. It is typically conducted by intelligence agencies with the goal to thwart criminals and terrorists. From a technical point of view, however, Internet surveillance can quickly turn into censorship: Both rely on special equipment which is deployed in national communication infrastructure. Often, the only difference is merely a set of configuration options. Surveillance equipment can be turned into censorship equipment within a matter of minutes. Furthermore, Reporters Without Borders' report identified two countries which made the unfortunate step from a surveillance to a censorship country: Bahrain and Belarus.

Censorship comes in many shapes. It can range from the widespread arrest of political opponents to the more subtle self-censorship which is caused by fear and is self-imposed by affected individuals. One technological aspect of censorship, Internet censorship, is merely a symptom of severe social and political problems and technology alone is unlikely to solve these problems. Technology can, however, be a useful tool towards solving these problems. This technological aspect of censorship is the content of this thesis.

1.1 Internet censorship

From a technical point of view, we can describe the problem of Internet censorship as a client intending to fetch blocked information which is located outside the censoring network. The censor, however, controls a subset of the network path between the client and the information it seeks to obtain. This control is then exercised to selectively block the client's network traffic. Note that this scenario is merely a simple example. In reality, Internet censorship comes in many flavours. The censor can be a nation-state government, an organisation, a company, or even an educational institution. Similarly, the blocked destination can be a web page, another user, an instant messaging service, or a search engine. What all these cases have in common is an entity—the censor—which seeks to block certain kinds of communication it considers harmful.

Internet censorship can be divided into censorship *circumvention* and *analysis*. While circumvention seeks to evade censorship systems, analysis aims to understand how the censor operates and what it blocks. We now attempt to characterise the problem of censorship circumvention.

A user who finds herself in a censored country and wants to circumvent the national firewall has to obtain circumvention software first. Once she downloaded and installed the circumvention software, the tool might have to perform a number of additional steps before it is ready to work. For example, some circumvention tools might first have to discover proxies over which the user's Internet traffic is subsequently relayed. All these preliminary steps are typically summarised as the *bootstrapping problem*.

Once a tool is fully bootstrapped, it is ready to transmit network traffic. However, the destination which receives this very network traffic can be blocked by the censor. We call this the *endpoint blocking problem*. This is a common problem for circumvention tools such as virtual private network (VPN) systems, which rely on the Internet protocol (IP) addresses of proxies remaining unblocked. A censor who is able to enumerate all these proxies is able to block them and render the circumvention tool useless.

If a circumvention tool is able to bootstrap and communicate with its intended destination, it still has to send network packets. A censor is able to inspect the content and shape of these network packets in order to block them. As a result, circumvention tools must find a way to either blend in with unblocked traffic or shape their traffic in a way that it is hard to identify. We call this problem the *traffic obfuscation problem*.

Note that in order for a circumvention tool to be effective in the face of many adversaries, it has to solve all three problems. It needs to be able to bootstrap, be resistant to endpoint blocking, and provide traffic obfuscation.

1.2 The Tor network

The Tor network [3] is a low-latency overlay network, that anonymises transmission control protocol (TCP) streams. Since its design in 2004, the Tor network emerged as the most popular low-latency anonymity network on the Internet and counts more than one million users every day. The four main technical components of the Tor network are relays, bridges, authorities, and clients, which are all explained in the following.

Relays are servers whose purpose is to accept incoming traffic and forward it to the next hop. As illustrated in Figure 1.1, there are three types of relays. Entry guards are accepting connections from Tor clients. Middle relays are situated in between entry guards and exit relays. Exit relays constitute the last hop in Tor's default of three-hop circuits.

Directory authorities, not illustrated in Figure 1.1, are a static set of nine servers which keep track of all Tor relays and summarise them in a document

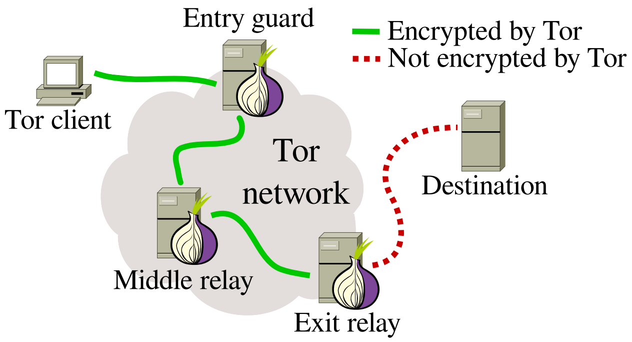


Figure 1.1: The structure of the Tor anonymity network. Clients establish three-hop circuits which anonymise traffic on its way to the destination.

that is called the network consensus. Every hour, all authorities vote on the next network consensus which is then made available to Tor clients. Directory authorities as well as their mirrors are contacted by clients while they are bootstrapping a connection to the Tor network.

Bridges are Tor relays which are not listed in the network consensus. They are meant for censorship circumvention and are distributed to users who find themselves unable to connect to Tor relays or directory authorities which are easy to blacklist on the IP layer. While bridges can also be blacklisted on the IP layer, the Tor Project implements strategies such as rate limiting to make it harder to harvest bridge IP addresses on a large scale.

Lastly, *Tor clients* connect to the Tor network in order to anonymise some of their network traffic. They randomly select three-hop circuits with every hop being a Tor relay. As mentioned earlier, clients regularly download the network consensus in order to learn about all currently running Tor relays. As of September 2014, the network contains approximately 6,000 relays and counts more than two million daily clients.

For this thesis, only the first hop in the Tor circuits is relevant, i.e., the path from the Tor client to the entry guard or the bridge. This path is where censors are typically active and attempting to block access to the network.

1.3 Contributions

This thesis addresses two research challenges, namely measuring and circumventing Internet censorship. As for measuring censorship, we seek to propose novel and practical measurement techniques which allow us to expose Internet censorship in areas of the Internet which are known to be difficult to measure. As for circumventing censorship, we propose the design and implementation of lightweight and usable tools which facilitate censorship circumvention in the face of a powerful adversary who tries to prevent us from doing so. To this end, we make the following contributions to these research challenges.

1. *An understanding of how the GFW is blocking the Tor network*
   Chapter 2 and 3 discuss a variety of networking experiments which were designed to shed light on how the Great Firewall of China (GFW) operates. Our results are novel and greatly extend preliminary research in this area.

2. *A lightweight circumvention tool to evade the GFW*
   Chapter 2 further proposes a lightweight tool which enables server-side evasion of the GFW's fingerprinting. The tool rewrites a server's TCP window size and can be run by bridge operators to prevent active probing attacks. This concept is novel and has not been proposed before.

3. *A design for a lightweight censorship analyser for Tor*
   Chapter 4 proposes a design for a lightweight censorship analyser for the Tor network. The analyser is meant to assist the Tor developers in debugging censorship incidents. The main contribution is that this is the first analyser which is "crowdsourced", meaning that it is supposed to be run by ordinary computer users.

4. *The concept of and experience with global censorship analysis*
   Chapter 5 shows how the RIPE Atlas platform can be used to detect some forms of censorship. Our case studies show that, despite being limited in their flexibility, censorship incidents can be detected and analysed from a great multitude of distributed vantages points, which advances the state of the art.

5. *The design of a blocking-resistant transport protocol*
   Chapter 6 discusses design and implementation of a blocking-resistant transport protocol. The protocol proposes two active probing-resistant authentication mechanisms and techniques to change the transported protocol's flow signature. Finally, the chapter contains an initial evaluation of the protocol. This chapter breaks new ground for circumvention protocols because in contrast to previously proposed systems, our protocol is polymorphic.

6. *Deployed software*
   Chapter 2 and 6 discuss software prototypes for server-side circumvention as well as a prototype of ScrambleSuit, our blocking-resistant transport protocol. All the code is available under a free license at http://www.cs.kau.se/philwint/.

1.4 Thesis structure

The remainder of this thesis is structured as follows. Note that all chapters are based on previously published papers (in particular, paper items P.1, P.2, P.5, P.7 and P.10), which are all referenced in Appendix A.

Chapter 2 discusses how the GFW is blocking the Tor anonymity network. The presented experiments are based on active probing; vantage points inside China were used to systematically determine what byte patterns trigger the Great Firewall, how it scans Tor bridges, and how its blocking mechanisms can be circumvented. The chapter is based on Paper P.10.

While Chapter 2 analyses the GFW using active probing, Chapter 3 takes a different approach. Instead of controlling vantage points in China, we use side channel techniques, which allow us to test the connectivity between Tor relays and randomly selected computers inside China. Using these side channels, this chapter provides a spatiotemporal analysis of the GFW's censorship capabilities. Our results confirm the findings of Chapter 2 and present additional findings. The chapter is based on Paper P.1.

The analysis techniques of both, Chapter 2 and Chapter 3 rely on a researcher controlling or conscripting machines in order to analyse Internet censorship. Chapter 4 presents an alternative approach that involves censored users, i.e., users inside a censored network are given the opportunity to run a lightweight tool which conducts a number of network tests and communicates the results to a researcher who is then able to analyse the data. The chapter is based on Paper P.7.

Chapter 5 discusses timely, global, and coarse-grained censorship analysis which is in stark contrast to the previous chapters which presented techniques for scope-constrained and fine-grained censorship analysis. In this chapter, we also present two case studies which demonstrate the practical value of our approach. The chapter is based on Paper P.2.

So far, all chapters focused on censorship analysis. Chapter 6 presents a censorship-resistant protocol, which attempts to defend against the GFW's active probing attacks that were presented in Chapter 2. The circumvention protocol employs a shared secret which is distributed out-of-band and uses polymorphic techniques to provide limited resistance against traffic analysis attacks. The chapter is based on Paper P.5.

Related work is covered in Chapter 7 and the thesis is concluded with a summary and an outlook in Chapter 8.

Chapter 2

How the GFW is blocking Tor

2.1 Introduction

On October 4, 2011 a user reported to the Tor bug tracker that unpublished bridges stop working after only a few minutes when used from within China [4]. Bridges are unpublished Tor relays and their very purpose is to help censored users connect to the Tor network if the “main entrance” is blocked [5]. The bug report indicated that the GFW has been enhanced with the potentiality of dynamically blocking Tor.

This censorship attempt is by no means China’s first attempt to block Tor. In the past, there have been efforts to block the website [6], the public Tor network [7, 8] and bridges [9]. According to a report [6], these blocks were implemented by simple IP address blacklisting and hyper text transfer protocol (HTTP) header filtering. All these blocking attempts had in common that they were straightforward and inflexible.

In contrast to the above mentioned censorship attempts, the currently observable block appears to be much more flexible and sophisticated. The GFW blocks bridges dynamically without simply enumerating their IP addresses and blacklisting them (cf. [10]).

In this chapter, we try to deepen the understanding of the infrastructure used by the GFW to block the Tor anonymity network. Our contributions are as follows.

- We reveal how users within China are prevented from connecting to the Tor network.
- We conjecture how China's Tor blocking infrastructure is designed.
- We discuss and propose circumvention techniques.

We also point out that censorship is a quickly moving target. Our results are only valid at the time of writing and might—and probably will—be subject to change.[1] Nevertheless, we believe that a detailed understanding of the GFW's current capabilities, a "censorship snapshot", is important for future circumvention work.

2.2 Experimental setup

During the process of preparing and running our experiments we took special care to not violate any ethical standards and laws. In addition, all our experiments were in accordance with the terms of service of our service providers. In order to ensure reproducibility and encourage further research, we publish our gathered data and developed code.[2] The data includes Chinese IP addresses which were found to conduct active probing of our bridge. We carefully configured our Tor bridges to remain unpublished and we always picked randomly chosen and unregistered ports to listen to so that we can be reasonably sure that the data is free from legitimate Tor users.

2.2.1 Vantage points

In order to ensure a high degree of confidence in our results, we used different vantage points. We had a relay in Russia, bridges in Singapore and Sweden,

[1] The data was mostly gathered in March 2012.

[2] The code is available at http://www.cs.kau.se/philwint/gfw/.

and clients in China. There is no technical reason why we chose Russia, Singapore, and Sweden.

Bridge in Singapore: A large part of our experiments was conducted with our Tor bridge located in Singapore. The bridge was running inside the Amazon EC2 cloud [11, 12]. The OpenNet Initiative describes Singapore as a country conducting minimal Internet filtering involving only pornography [13]. Hence, we assume that our experimental results were not interfered with by Internet filtering in Singapore.

Bridge in Sweden: To reproduce experiments, we set up Tor bridges located at Karlstad University in Sweden. Internet filtering for these bridges was limited to well-known malware ports, so we can rule out filtering mechanisms interfering with our results.

Relay in Russia: A public Tor relay located in a Russian data centre was used to investigate the type of blocking, public Tor relays are undergoing. The relay served as a middle relay, meaning that it is not selected as the first hop in a Tor circuit and it does not see the exit traffic of users.

Clients in China: To avoid biased results, we used two types of vantage points inside China: open SOCKS proxies and a virtual private system (VPS). We compiled a list of public Chinese SOCKS proxies by searching Google. We were able to find a total of 32 SOCKS proxies which were distributed amongst 12 distinct autonomous systems. We connected to these SOCKS proxies from computers outside of China and used them to repeat certain experiments on a smaller scale to rule out artefacts limited to our VPS.

Our second vantage point and primary experimental machine is a VPS we rented. The VPS ran GNU/Linux and resided in the autonomous system number (ASN) 4808. We had full root access to the VPS which made it possible for us to sniff traffic and conduct experiments below the application layer. Most of our experiments were conducted from our VPS whereas the SOCKS proxies' primary use was to reproduce results.

2.2.2 Shortcomings

Active analysis of a censorship system can easily attract the censor's attention if no special care is taken to "stay under the radar". Due to the fact that China is a sophisticated censor with the potential power to actively falsify measurement results, we have to point out potential shortcomings in our experimental setup.

We have no reliable information about the owners of our public SOCKS proxies. Whois lookups did not yield anything suspicious but the information in the records can be spoofed. It might even be possible that the SOCKS proxies are operated by Chinese authorities. Second, our VPS was located in a data centre where Tor client connections typically do not originate. We also had no information about whether our service provider conducts Internet filtering and the type or extent thereof.

2.3 Analysis

2.3.1 How are bridges and relays blocked?

The first step in bootstrapping a Tor connection requires connecting to the directory authorities to download the consensus which contains all public relays. We noticed that seven of all eight directory authorities are blocked *on the IP layer*.[3] These machines responded neither to TCP, nor to Internet control message protocol (ICMP) packets. One authority turned out to be reachable and it was possible for us to download the consensus. We have no explanation why this particular machine was unblocked.

After the consensus has been downloaded, clients can start creating circuits. Using our Russian relay, we found out that when a client in China connects to a relay, the GFW lets the TCP SYN pass but drops the SYN/ACK sent by the bridge to the client (cf. [14]). The same happens when a client tries to connect to a blocked bridge. However, clients are still able to connect to different TCP ports as well as ping the bridge. We believe that the reason for the GFW blocking relays and bridges by IP:port tuples rather than by

[3] Note that in 2012, the Tor network had only eight directory authorities.

IP addresses is to minimise collateral damage.

2.3.2 How long do bridges remain blocked?

To answer this question, we started two Tor bridges on our machine in Singapore. Both Tor processes were private bridges and listening on TCP port 27418 and 23941, respectively. Both ports were chosen randomly.

In the next step, we made the GFW block both IP:port tuples by initiating Tor connections to them from our VPS in China. After both tuples were blocked, we set up iptables [15] rules on our machine in Singapore to whitelist our VPS in China to port 23941 and drop all other connections to the same port. That way, the tuple appeared unreachable to the GFW but not to our Chinese VPS. Port 27418 remained unchanged and hence reachable to the GFW. We then started monitoring the reachability of both Tor processes by continuously trying to connect to them using telnet from our VPS.

After approximately 12 hours, the Tor process behind port 23941 (which appeared to be unreachable to the GFW) became reachable again whereas connections to port 27418 still timed out and continued to do so. In our iptables logs we could find numerous connection attempts originating from Chinese scanners. This observation suggests that once a Tor bridge has been blocked, it only remains blocked if Chinese scanners are able to *continuously connect* to the bridge. If they cannot, the block is *removed*. We believe that this is an important aspect of the GFW because if blocked tuples were only added and never removed, it would be possible to overwhelm the GFW by continuously adding new tuples.

2.3.3 Is the public network reachable?

To verify how many public relays are reachable from within China, we downloaded the consensus published at February 23, 2012 at 08:00 universal time, coordinated (UTC). At the time, the consensus contained descriptors for a total of 2,819 relays. Then, from our Chinese VPS we tried to establish a TCP connection to the Tor port of every single relay. If we were able to successfully establish a TCP connection we labelled the relay as reachable, otherwise unreachable.

We found that our VPS could successfully establish TCP connections to 47 out of all 2,819 (1.6%) public relays. We manually inspected the descriptors of the 47 relays but could not find any common property which could have been responsible for the relays being unblocked. We checked the availability of the reachable relays again after a period of three days. Only one out of the original 47 unblocked relays was still reachable.

2.3.4 Where does the fingerprinting happen?

We want to gain a better understanding of where the Chinese deep packet inspection (DPI) boxes are looking for the Tor fingerprint. In particular, we tried to investigate whether the DPI boxes also analyse domestic and ingress traffic.

We used six open Chinese SOCKS proxies (in ASN 4134, 4837, 9808 and 17968) as well as six PlanetLab nodes (in ASN 4538 and 23910) to investigate domestic fingerprinting. We simulated the initiation of a Tor connection multiple times to randomly chosen TCP ports on our VPS, but could not attract any active scans.

Previous research confirmed that HTTP keyword filtering done by the GFW is bidirectional [16], i.e., keywords are scanned in ingress as well as in egress traffic. We wanted to find out whether this holds true for the Tor DPI infrastructure too. To verify that, we tried to initiate Tor connections to our Chinese VPS from our vantage points in Sweden, Russia, and Singapore. Despite multiple attempts we were not able to attract a single scan.

The above mentioned results indicate that Tor fingerprinting is probably *not done in domestic traffic* and only with traffic going from *inside China to the outside world*. We believe that there are two reasons for that. First, fingerprinting domestic traffic in addition to international traffic would dramatically increase the amount of data to analyse since domestic traffic is believed to be the largest fraction of Chinese traffic [17]. Second, at the time of this writing there are no relays in China so there is no need to fingerprint domestic or ingress traffic. Tor usage in China means being able to connect to the outside world.

2.3.5 Where are the scanners coming from?

To get extensive data for answering this question, we continuously attracted scanners over a period of 17 days ranging from March 6, 2012 to March 23, 2012. We attracted scanners by simulating Tor connections from within China to our bridge in Singapore.[4] To simulate a Tor connection, we developed a small tool whose sole purpose was to send the Tor transport layer security (TLS) client hello to the bridge and terminate after receiving the response. We could also have used the original Tor client to do so, but our tool was much more lightweight which was helpful given our resource-constrained VPS. After every Tor connection simulation, our program remained inactive for a randomly chosen value between 9 and 14 minutes. The experiment yielded *3,295 scans of our bridge.* Our findings described below are based on this data.

Scanner IP address distribution: We are interested in the scanners' IP address distribution, i.e., how often can we find a particular IP address in our logs? Our data exhibits two surprising characteristics:

1. More than half of all connections—1,680 of 3,295 (51%)—were initiated by *a single* IP address, namely 202.108.181.70.

2. The second half of all addresses is almost *uniformly distributed.* Among all 1,615 remaining addresses, 1,584 (98%) were unique.

The IP address 202.108.181.70 clearly stands out from all observed scanners. Aside from its heavy activity we could not observe any other peculiarities in its scanning behaviour. The whois record of the address states a company named "Beijing Guanda Technology Co.Ltd" as owner:

```
inetnum:        202.108.181.0 - 202.108.181.255
netname:        BJ-GD-TECH-CO
descr:          Beijing Guanda Technology Co.Ltd
country:        CN
admin-c:        CH455-AP
```

[4] We reproduced this experiment with a bridge in Sweden and with open Chinese SOCKS proxies. Our findings were the same.

```
tech-c:          SY21-AP
mnt-by:          MAINT-CNCGROUP-BJ
changed:         suny@publicf.bta.net.cn 20020524
status:          ASSIGNED NON-PORTABLE
source:          APNIC
```

We could only find a company named “Guanda Technology Amusement Equipment Co., Ltd” on the Internet. It is not clear whether this is the same company. However, as explained below, we have reason to believe that the scanners’ IP addresses are spoofed by the GFW so the owner of the IP address, assuming that it even exists, might not be aware of the scanning activity.

Whois and reverse domain name system (DNS) lookups of all the seemingly random IP addresses suggested that the IP addresses were coming from Internet service provider (ISP) pools. For example, all valid reverse DNS lookups contained either the strings “adsl” or “dynamic”.

Autonomous system origin: We used the IP address to ASN mapping service provided by Team Cymru [18] to get the ASN for every observed scanner. The result reveals that all scanners come from *one of three* autonomous systems (ASs)[5] with the respective percentage in parantheses:

AS4837: China Unicom Backbone (65.7%)

AS4134: China Telecom Backbone (30.5%)

AS17622: China Unicom Guangzhou network (3.8%)

AS4134 is owned by China Telecom while AS4837 and AS17622 is owned by China Unicom. AS4134 and AS4837 are the two largest ASs in China [19] and play a crucial role in the country-wide censorship as pointed out by Xu, Mao, and Halderman [20]. Furthermore, Roberts et al. [19] showed that China’s AS-level structure is *far from uniform* with a significant fraction of the country’s traffic being routed through AS4134 or AS4837.

[5] Recent research efforts by Roberts et al. showed that China operates 177 autonomous systems [19].

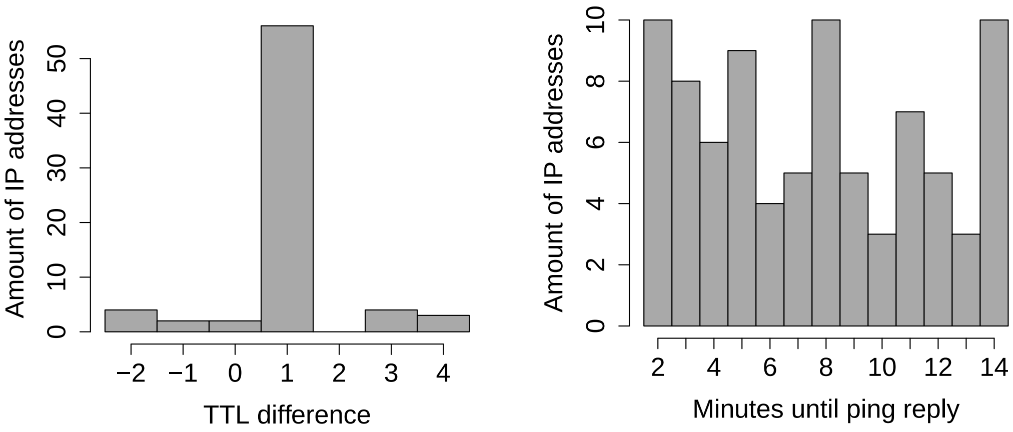
IP address spoofing: During manual tests, we noticed that sometimes shortly after a scan, the respective IP address starts replying to pings,[6] but with a *different IP time-to-live value (TTL)* than during the scan. In order to have more data for our analysis, we implemented a script which automatically collects additional data as soon as a scanner connects and again some minutes afterwards. In particular, the script 1) runs TCP, user datagram protocol (UDP), and ICMP traceroutes immediately after a scan and again 15 minutes later; 2) continuously pings the scanning IP address for 15 minutes; and 3) captures all network traffic during these 15 minutes using tcpdump.

Between March 21 and 26, 2012, we started an independent experiment to attract scanners and let our script gather the above mentioned data. We caused a total of 429 scans coming from 427 unique IP addresses. From all 429 scans, we then extracted all connections where the continuous 15 minutes ping resulted in at least one ping reply. This process yielded a subset of 85 connections which corresponds to approximately 20% of all observed connections. We analysed the 85 connections by computing the amount of minutes until the respective IP address started replying to our ping requests and the IP TTL difference (new TTL – old TTL) between packets during the scan and ping replies.

The results are shown in Figure 2.1(a) and 2.1(b). Figure 2.1(b) illustrates how long it took for the hosts to start replying to the ping requests. No clear pattern is visible. Figure 2.1(a) depicts all IP TTL differences after the scan (when the host starts replying to ping packets) and during the scan. We had 14 outliers with a TTL difference of 65 and 192 but did not list them in the barplot. It is clearly noticeable that the difference was mostly one, meaning that after the scan, the TTL was *by one more than during the scan.*

One explanation for the changing TTL—but definitely not the only one—is that the GFW could be *spoofing IP addresses.* The firewall could be abusing several IP address pools intended for Internet users to allocate short-lived IP addresses for the purpose of scanning.

[6] Note that this is never the case during or immediately after scans. All ICMP packets are being dropped.



(a) The IP TTL difference between after and during the scan.

(b) The amount of minutes until the hosts started replying to pings.

Figure 2.1: IP TTL difference (a) and duration until ping replies (b).

2.3.6 When do the scanners connect?

Figure 2.2 visualises when the scanners in our data set connected. The y axis depicts the minutes of the respective hour. Contrary to December 2011, when Wilde ran his experiments, the scanners seemed to use a broader time interval to launch the scans. In addition, the data contains two time intervals which are free from scanning. These intervals lasted from March 8, 2012 at around 16:30 to March 9, 2012 at 09:00 and from March 14, 2012 at 09:30 to March 16, 2012 at 3:30 UTC. We have no explanation why the GFW did not conduct scanning during that time.

Closer manual analysis yielded that the data exhibits a *diurnal pattern*. In order to make the pattern visible, we processed the data as four distinct time series with every 15 minutes interval forming one time series, respectively. We smoothed the time series' data points using simple exponential smoothing with a smoothing factor $\alpha = 0.05$. The result—a subset of the data ranging from March 16, 2012 to March 23, 2012—is shown in Figure 2.3. Each of the four diagrams represents one of the 15 minutes intervals. The diagrams show that depending on the time of the day, on average, scanners connect either close to the respective 15 minutes multiple or a little bit later.

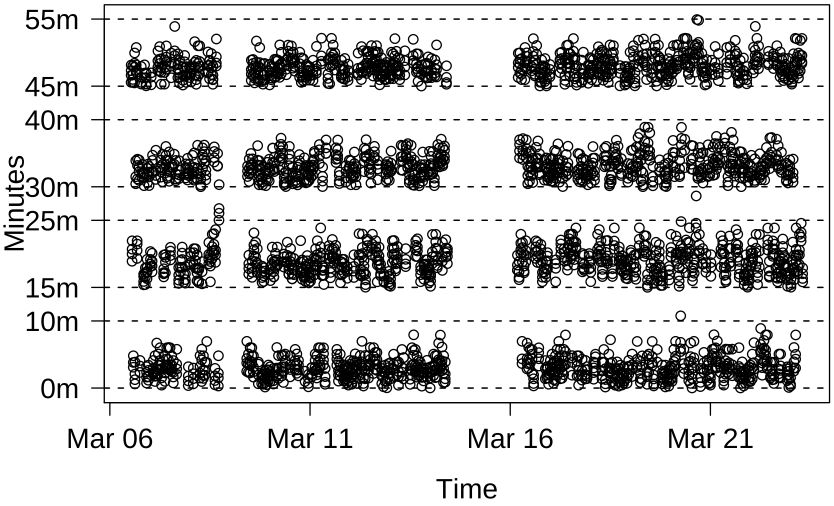


Figure 2.2: Time points when scanners were found connecting to our bridge. Every point represents one Tor connection which was established by an active prober in China.

We conjecture that the GFW maintains *scanning queues*. When the DPI boxes discover a potential Tor connection, the IP:port tuple of the suspected bridge is added to a queue. Every 15 minutes, these queues are processed and all IP:port tuples in the queue are being scanned. We believe that during the day, the GFW needs more time to process the queues since there are probably more users trying to connect to the Tor network which leads to more scans.

2.3.7 Blocking malfunction

During our experiments, we noticed several outages of active probing. This lack of probing made it possible for us to successfully initiate Tor connections without causing bridges to get scanned and blocked. The Tor bridge usage statistics between January and June 2012 contain several usage spikes which confirm outages in the blocking infrastructure [21]. Another downtime was observed by Wilde [22].

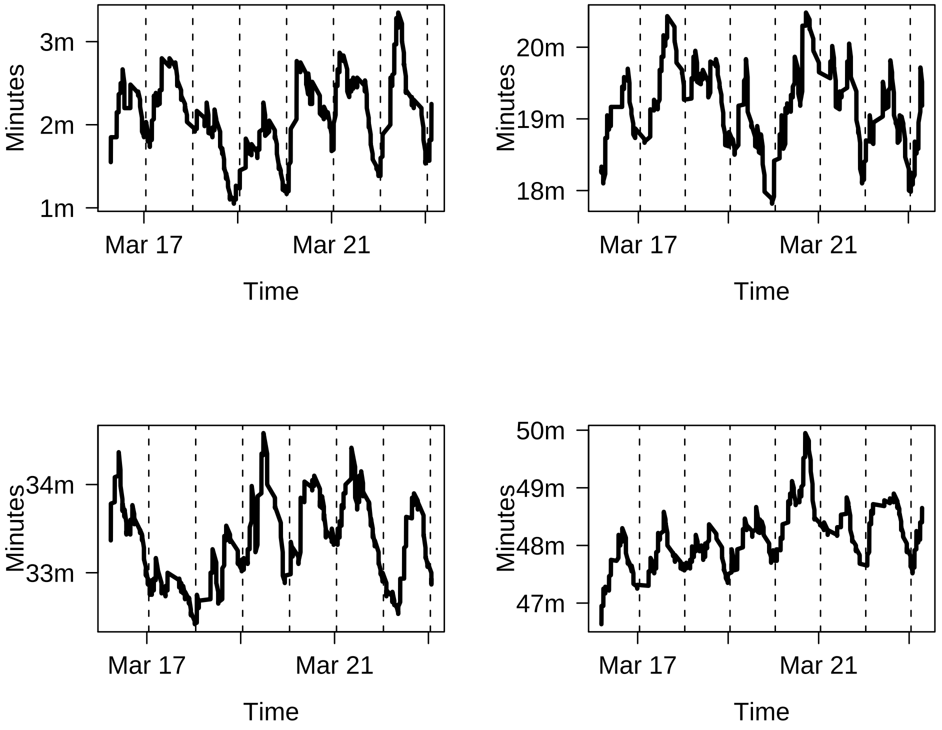


Figure 2.3: Diurnal scanning connection pattern. It can be seen that during the day, more active probing is happening than during the night.

2.4 Circumvention

While there is a large body of work dedicated to censorship circumvention systems (cf. Section 7.2 and [23]), we limit this section to the discussion of obfsproxy [24] and propose a novel way to evade the GFW's DPI boxes.

2.4.1 Obfsproxy

The Tor Project is developing a tool called obfsproxy [24]. The tool runs independently of Tor and is obfuscating the network traffic it receives from the Tor process. As long as both, the bridge and the client are running the tool, the Tor traffic transmitted between them can be obfuscated so that the Chinese DPI boxes are no longer able to identify the TLS cipher list. Obfsproxy implements a *pluggable transport layer* which means that modules can be written that support different types of obfuscation.[7] Chapter 6 proposes such a module.

2.4.2 Packet fragmentation

A network intrusion detection system (NIDS) evasion technique originally discussed by Ptacek and Newsham in 1998 [26] is packet fragmentation which exploits the fact that some network intrusion detection systems do not conduct packet reassembly. Crandall et al. also considered fragmentation to evade the GFW's keyword-based detection [27].

To find out how the GFW handles fragmented Tor traffic, we used the tool fragroute [28] to enforce packet fragmentation on our VPS in China. We configured fragroute to split the TCP stream to segments of 16 bytes each. In our test, it took five TCP segments to transmit the fragmented cipher list to our bridge. Despite initiating several fragmented Tor connections, we never observed any active probing and could use Tor without interference. This experiment indicates that the GFW does not conduct packet reassembly. A similar observation was made by Park and Crandall [29]. However, client-side fragmentation is an impractical solution given that this method must be supported by *all* connecting Chinese users. A single client who does not use fragmentation triggers active probing which then leads to the block of the respective bridge. Another disadvantage is the significant protocol overhead due to the smaller TCP segments which leads to a decrease in throughput.

Due to these shortcomings, we propose an evasion technique based on *server-side packet fragmentation*. In essence, our technique manipulates the TCP window size. The purpose of TCP's window size is to provide flow control. Both communicating parties dynamically adapt and communicate their window size to let the other party know how many bytes they are willing to accept at the moment. For example, if the receiving application struggles with processing incoming bytes at the rate at which they are being received, the receiver's TCP stack shrinks the window size which causes the throughput to drop. Our evasion technique makes the server reduce its window size in

[7] An overview of currently developed modules can be found online [25].

the beginning of the Tor connection. In particular, we want to split the Tor client's cipher list inside the TLS client hello into two parts. If the server's announced window size right before receiving the first TCP segment is small enough, the client uses two TCP segments to transmit the TLS client hello. In addition to being diminished, the server's window size must be randomised. Small TCP windows are atypical and would give the censor the opportunity to actively look for such small windows. By randomly selecting the window size from a small set of possible values, we make this task more difficult. We empirically determined the set of feasible values to be {60..90}.

We implemented our approach in a C tool called brdgrd which encompasses only 300 lines of code.[8] The tool must be run on a Tor bridge and makes use of the libnetfilter_queue application programming interface (API) [30] which makes it possible to copy network packets from kernel space to user space. User space applications can then pass a verdict, i.e., decide if the packet should be forwarded or dropped by the kernel. If the packet should be forwarded, the application is further able to modify the packet, copy it back to the kernel, and instruct the kernel to forward the modified packet. We make use of this technique to rewrite the TCP window size. Using iptables, we copy only the bridge's TCP SYN/ACK segments to user space which keeps the performance impact at a minimum. All other TCP segments, including those transporting bulk data, are directly forwarded.

Our server-side fragmentation tool has the advantage that it is easy to deploy and it only interferes with Tor connections by rewriting the announced TCP window during the TCP handshake. Hence, there are virtually no performance implications. In fact, the bridges's kernel is not even aware of the fact that its window size was manipulated.

2.4.3 Practical experience

After implementing brdgrd, we tested it on several Tor bridges which were located outside China. We found that we were able to repeatedly connect to these bridges and establish Tor connections. We then began the deployment of brdgrd to several high-performance Tor bridges. These bridges were man-

[8] The tool is available on our project website http://www.cs.kau.se/philwint/gfw/.

ually distributed to users in China. For many months, brdgrd proved to be successful in evading the GFW's fingerprinting of Tor connections.

Approximately one year later, in 2013, reports emerged that bridges running brdgrd were now also subject to blocking shortly after a user connected to it. We manually investigated these incidents and discovered that bridges were indeed actively probed despite the cipher list being split into two TCP segments. While it is difficult to attribute this change to a particular event, it is possible that the GFW started to conduct TCP stream reassembly which would render fragmentation useless.

brdgrd was never meant to be a sustainable resource in the arms race that is censorship circumvention. Nevertheless, development and deployment was straightforward and it prevented Tor bridges from getting blocked while obfsproxy was still under development. We believe that at the core of the arms race, economic principles are at work (cf. [31]). If a particular circumvention technique should be deployed or not depends on the cost of developing it and on the censor's cost when blocking it.

2.5 Conclusion

In this chapter, we showed how access to Tor is being denied in China and we conjectured how the blocking infrastructure is designed. In addition, we discussed countermeasures intended to "unblock" the Tor network. Our findings include that the Great Firewall of China might spoof IP addresses for the purpose of probing Tor bridges and that domestic as well as ingress traffic does not seem to be subject to Tor fingerprinting. We also showed that the firewall used to be easily circumvented by fragmented packets.

Chapter 3

Large-scale spatiotemporal analysis of the GFW

3.1 Introduction

As mentioned earlier, the GFW has a tight grip on several layers of the TCP/IP model and is known to block IP addresses [32], TCP ports [32], DNS requests [33–35], HTTP requests [16, 27, 29], circumvention tools, and even social networking sites [36]. The previous chapter showed that Tor, once having had 30,000 users solely from China, now is largely inaccessible from within China's borders as illustrated in Figure 3.1.

The amount of users trying to connect to the Tor network indicates that there is a strong need for practical and scalable circumvention tools. Censorship circumvention, however, builds on *censorship analysis*. A solid understanding of censorship systems is necessary in order to design sound and sustainable circumvention systems. However, it is difficult to analyse Internet censorship without controlling either the censored source machine or its—typically uncensored—communication destination. This problem is usually tackled by obtaining access to censored source machines (as it was done in the previous chapter), finding open proxies, renting virtual systems, or by cooperating with volunteers inside the censoring country. In the absence of these possibilities, censorship analysis has to resort to observing traffic on the server's side and inferring what the client is seeing.

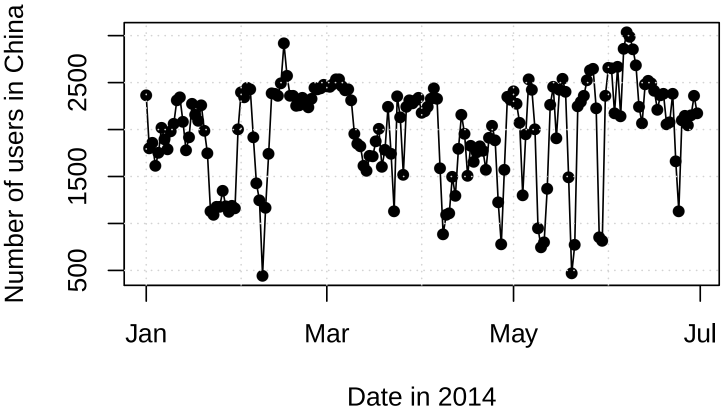


Figure 3.1: The approximate amount of directly connecting Tor users (as opposed to connecting over bridges) for the first months of 2014. While the number of users varies, it rarely exceeds 3,000.

This chapter tries to fill this gap by presenting and evaluating network measurement techniques which can be used to expose censorship while controlling *neither the source nor the destination machine*. This puts our work in stark contrast to previous work which had to rely on proxies or volunteers, both of which provide limited coverage of the censor's networks. By being mostly independent of source and destination machines, we are able to shed light on entirely unexplored areas of the Internet. We evaluate our techniques by applying them to the Tor anonymity network, thereby handing the Tor Project practical tools to measure the reachability of their network. Such tools are needed because bridges are frequently blocked in China without the bridge operators or the Tor Project noticing [37]. Our work makes it possible to test the reachability of these bridges without having a vantage point in China. As a result, the Tor Project is able to learn which subset of bridges is still reachable and hence undiscovered by the GFW. This knowledge facilitates the optimisation of bridge distribution [38], e.g., bridges blocked in China are only given out to users outside China.

Our techniques are currently limited to testing *basic IP connectivity*. Thus, we can only detect censorship on lower layers of the network stack, i.e., before a TCP connection is even established. This kind of low-level censorship is very important to the censors, however. For example, while social media controls on domestic sites in China, such as Weibo, can be very sophisticated, users would simply use alternatives such as Facebook if the low-level IP address blocking were not in place to prevent this. Also, DPI does not scale as well in terms of raw traffic as does lower-level filtering. Nevertheless, we acknowledge that our techniques are not applicable if censors only make use of DPI to block Tor as it was or is done by Ethiopia, Kazakhstan, and Syria [39].

We are interested not only in finding patterns in the GFW's *failures*, but also in gaining a better understanding of how the GFW is *architected* within China's backbone and provincial networks and whether previously observed details of its *implementation* are observed throughout the country. To this end, we focus our efforts on testing the following hypotheses that will illuminate the GFW's architecture and implementation. All hypotheses are with respect to the filtering of TCP/IP packets based on IP addresses and port numbers.

Hypothesis 1. In general, from any client to any destination if a SYN packet is filtered by the GFW then a RST with the same source, destination, and port numbers will also be filtered. For brevity, we refer to this hypothesis as "*RSTs are treated the same as SYNs*".

Hypothesis 2. There are no conspicuous geographic patterns in the GFW's failures. In other words, failures can occur in any part of the country. For brevity, we refer to this hypothesis as "*No geographic patterns in failures*".

Hypothesis 3. In general, the GFW blocks connections to Tor relays by dropping SYN/ACK segments with IP address and port information that matches known Tor relays. Other types of filtering seen for Tor relays in China (e.g., dropping SYN segments) are a negligible fraction of the censorship. For brevity, we refer to this hypothesis as "*server-to-client blocking*".

Hypothesis 4. At least some of the failures of the GFW are persistent, meaning that the client and server are able to communicate throughout the day. Note that this could also be due to intentionally whitelisted destinations, but in this chapter we refer to all cases where clients in China can access Tor relays as "failures". For brevity, we refer to this hypothesis as "*some failures are persistent*".

Hypothesis 5. At least some of the failures of the GFW exhibit diurnal patterns, where a client and blacklisted server can communicate at some times of the day but not others. For brevity, we refer to this hypothesis as "*some failures have diurnal patterns*".

Hypothesis 6. In general, packets that are subject to censorship traverse at least one or two hops, and sometimes more, into China before they are dropped by the GFW. For brevity, we refer to this hypothesis as "*blocking is in the backbone*".

By testing the above hypotheses, we further increase the public's knowledge about the GFW and by presenting and evaluating our measurement techniques, we equip circumvention system developers with a set of tools to analyse and debug censorship incidents. In summary, this chapter makes the following contributions:

- We describe the first real-world application of the hybrid idle scan [14, 40] to a large-scale Internet measurement problem, in which we measure the connectivity between the Tor anonymity network and clients in China over a period of four weeks.
- We present and evaluate a novel side channel based on the Linux kernel's SYN backlog which enables indirect detection of packet loss.
- We increase the community's understanding of how the GFW is architected and how its blocking of the Tor network looks from different clients all over China.

3.2 Networking background

The research questions we seek to answer require high geographical diversity of clients in China. Typically, such a study would only be possible if we could find and control vantage points in all of China's provinces. Instead, we exploit side channels allowing us to detect intentional packet dropping—without controlling the two affected machines. In particular, we use *hybrid idle scans* (see Section 3.2.3) and *SYN backlog scans* (see Section 3.2.1). The idea behind these side channels as well as their prerequisites are discussed in this section.

3.2.1 Side channels in Linux's SYN backlog

A performance optimisation in the Linux kernel's SYN backlog can be used to detect intentional packet dropping. Half-open TCP connections of network applications are queued in the kernel's *SYN backlog* whose size defaults to 256. These half-open connections then turn into fully established TCP connections once the server's SYN/ACK was acknowledged by the client. If a proper response is not received for an entry in the SYN backlog, it will retransmit the SYN/ACK several times. However, if the SYN/ACK and its respective retransmissions are never acknowledged by the client, the half-open connection is removed from the backlog. When under heavy load or under attack, a server's backlog might fill faster than it can be processed. This causes attempted TCP connections to not be fully handled while pending TCP connections time out. The Linux kernel mitigates this problem by *pruning* an application's SYN backlog. If the backlog becomes more than half full, the kernel begins to reduce the number of SYN/ACK retransmissions for all pending connections [41]. As a result, half-open connections will time out earlier which should bring the SYN backlog back into uncritical state. We show that the Linux kernel's pruning mechanism—by design a *shared resource*—opens a side channel which can be used to measure intentional packet drops targeting a server. This is possible without controlling said server.

Our key insight is that we can remotely measure the approximate size of a server's SYN backlog by sending SYN segments and counting the number of corresponding SYN/ACK retransmissions. Starting with version number 2.2, the Linux kernel retransmits unacknowledged SYN/ACK segments five times [42]. As a result, we expect to receive the full number of five retransmissions when querying a service whose SYN backlog is less than half full. If, on the other hand, the backlog becomes more than half full, we will observe less than five retransmissions. When applied to the problem of intentional packet dropping, this allows us to infer whether a firewall blocks TCP connections by dropping the client's SYN or the server's SYN/ACK segment.

It is worth mentioning that a server's backlog state can also be inferred by coercing it into using *SYN cookies* [43]. A server using SYN cookies reveals that its SYN backlog is completely full. However, this measurement technique is effectively a SYN flood and TCP connections which were established using SYN cookies suffer from reduced throughput due to the lack of flow control and window scaling. In contrast to triggering SYN cookies, our technique has no negative impact on servers or other clients' connections, when applied carefully.

3.2.2 The global IP identifier

The IP identification number (IPID) is a unique number assigned to IP packets in case they are fragmented along a path. The receiving party is able to reassemble the fragmented packets by looking at their IPID field. Most modern TCP/IP stacks increment the IPID field per connection or randomise it, as opposed to *globally incrementing* it. A machine with a globally incrementing IPID keeps a global counter that is incremented by 1 for every packet the machine sends, regardless of the destination IP address. Being a *shared resource*, the IPID can be used by a measurement machine talking to a remote machine to estimate how many packets the remote machine has sent to other machines. Throughout this chapter, we refer to machines with globally incrementing IPIDs as simply machines with "global IPIDs".

3.2.3 Hybrid idle scan

Ensafi et al. [14, 40] discovered a new method for remotely detecting intentional packet drops on the Internet via side channel inferences. Their technique can discover packet drops (e.g., caused by censorship) between two remote machines, as well as infer in which direction the packet drops are occurring. The only major requirements for their approach are a client with a global IPID and a target server with an open port. Access to the

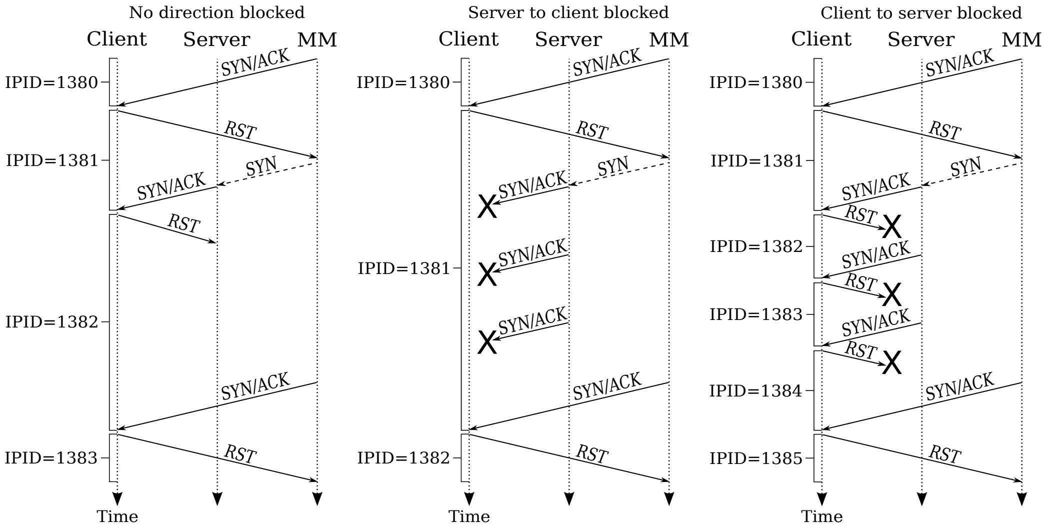


Figure 3.2: Three different cases of packet dropping that our method can detect. MM is our measurement machine.

client or the server is not required. Conceptually, the hybrid idle scan technique can turn approximately 1% of the total IPv4 address space [14] into conscripted measurement machines that can be used as vantage points to measure IP address-based censorship—without having root access on those machines. This is why we employ the hybrid idle scan technique for our geographic study of how Tor is blocked in China.

As shown in Figure 3.3, the hybrid idle scan implementation queries the IPID of the client to create a time series. By sending SYN/ACKs from the measurement machine and receiving RST responses, the IPID of the client can be recorded. The time series is used to compare a base case (when no traffic is being generated other than noise) to a period of time when the server is sending SYN/ACKs to the client (because of our forged SYNs). Recall that the hybrid idle scan assumes that the client's IPID is global and the server has an open port. By comparing two phases, one phase where no SYN packets are sent to the server and one phase where SYN packets are sent to the server with the return IP address spoofed to appear to be from the client, the hybrid idle scan technique can detect *three different cases* (plus an error case), shown in Figure 3.2, with respect to IP packets being dropped by the network in between the client and the server:

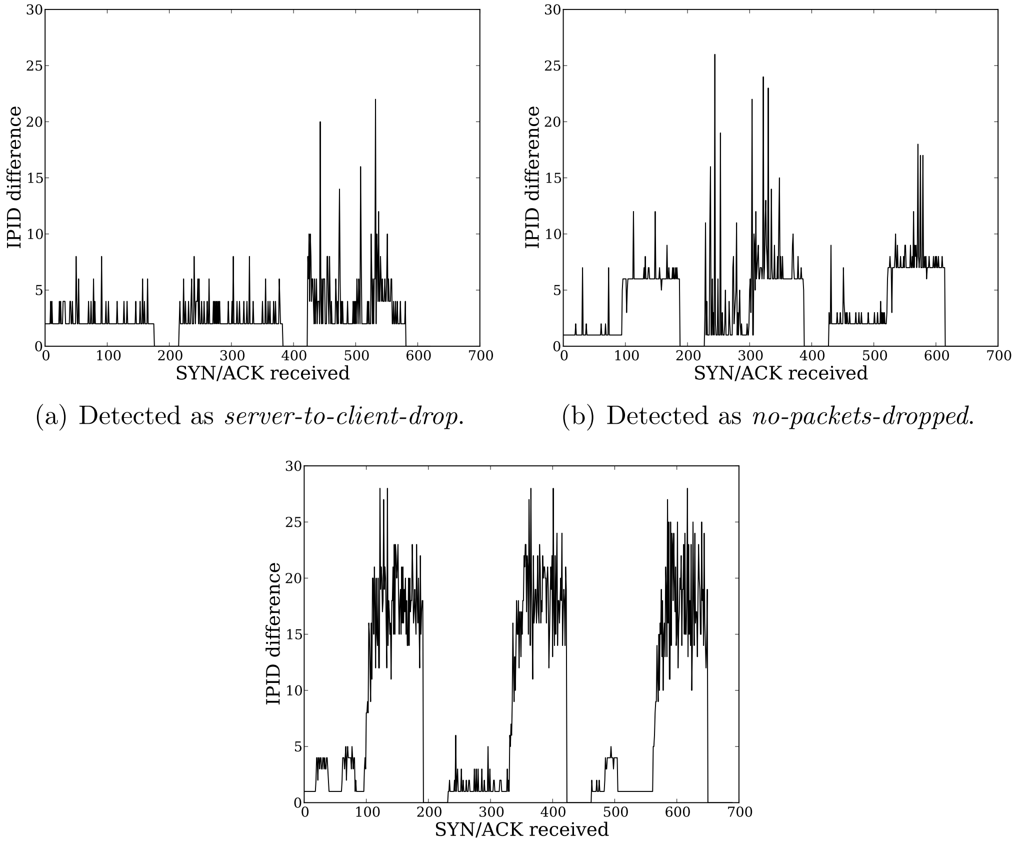
1. **Server-to-client-dropped:** SYN/ACKs are dropped in transit from the server to the client causing the client's IPID to not increase at all (except for noise). See Figure 3.3(a).

2. **No-packets-dropped:** If no intentional packet dropping is happening, the client's IPID will go up by exactly one. See Figure 3.3(b). This happens because the server's SYN/ACK is unsolicited and answered by the client with a RST segment causing the server to remove the entry from its SYN backlog and not retransmit the SYN/ACK.

3. **Client-to-server-dropped:** The RST responses sent by the client to the server are dropped in transit. In this case, the server will continue to retransmit SYN/ACKs and the client's IPID will get incremented by the total number of (re)transmitted SYN/ACKs, which is typically three to six. See Figure 3.3(c). This may indicate null routing, the simplest method for blacklisting an IP address.

4. **Error:** A measurement error happens if networking errors occur during the experiment, the IPID is found to not be global throughout the experiment, a model is fit to the data but does not match any of the three non-error cases above, the data is too noisy and intervention analysis fails because we are not able to fit a model to the data, and/or other errors.

ARMA models are used to distinguish these cases. This overcomes autocorrelated noise in IPID values (e.g., due to packet loss, packet delay, or other traffic that the client is receiving). More details about the ARMA modelling are described by Ensafi et al. [14, 40].



(a) Detected as *server-to-client-drop*.

(b) Detected as *no-packets-dropped*.

(c) Detected as *client-to-server-drop*.

Figure 3.3: Each subfigure illustrates a time series based on IPID differences for a specific blocking case. Despite high amounts of noise, our ARMA modelling can still detect the blocking case correctly.

3.3 Experimental methodology

3.3.1 Encountered challenges

Over the course of running our experiments and analysing our data, we faced a number of challenges which we discuss here.

Churn in the Tor network: While the size of the Tor network does not vary considerably over a short period of time, the network's *churn rate* can render longitudinal studies difficult. For example, the median size of Tor's network consensus (i.e., the number of Tor relays in the network) in March 2014 was 5,286. In total, however, March has seen 13,343 *unique relays*—many of which were online for only hours. To minimise the chance of selecting unstable Tor relays for longitudinal studies, only relays having earned the "Stable" flag should be considered [44, §3.4.2]. Furthermore, the relay descriptor archives could be examined to calculate a relay's reachability over time [45]. We selected only Tor relays that had an uptime of at least five days, and filtered out all data points where a node appeared to have left the network. After having run our experiments, we removed one Tor relay in Argentina from our data because its Tor and web ports switched during our experiments.

Geolocation of routers: For geolocating routers, we used MaxMind's GeoIP2 City database [46]. As of April 2014, this database lacks accurate geolocation information for backbone routers in China. While provincial routers can typically be mapped to their province based on whois records, backbone routers are all mapped to the same bogus location at latitude 35 and longitude 105 which resides in an unpopulated area in central China. We also used MaxMind for geolocating clients, for which it is fairly accurate. For the location of routers, we used a combination of *whois information* and *round-trip delays* per hop. We discarded hops in our data that have whois records from China but are actually in Hong Kong or Pasadena, CA (where ChinaNet has a Point of Presence).

Diurnal patterns: For most measurements in this chapter, we measured once per hour throughout the day. This avoids bias and distortion. For example, if we measured one set of clients in the morning and one set at night, differences between the two sets of clients may be due to different traffic patterns at the different times of day and not a property of the different set of clients. Thus we always randomise the order of our experiments when possible and repeat all measurements every hour for at least one full day.

3.3.2 Experimental design and setup

Over the course of our experiments, we made use of three sets of Linux-based measurement machines in the U.S., China, and Europe. These three sets of machines correspond to the three main datasets that we collected.

Machines in the U.S.: The three machines used for our hybrid idle scans (see Section 3.2.3) and SYN backlog scans (see Section 3.2.1) were located at our university campus (UC). All machines had a direct link to a research network which is free from packet filtering and does not conduct egress filtering to block spoofed return IP addresses. Furthermore, the UC measurement machines have IP addresses that are not bound to any interfaces in order to eliminate unsolicited network packets. For example, a measurement machine's kernel should never send a RST when it receives a SYN/ACK. The data set collected using the hybrid idle scan from these machines is a large-scale geographic pairing of many clients (in China and other countries) with many Tor relays and web servers around the world (mostly outside China). It complements the other data sets discussed below because it gives a complete cross-section of censorship between many clients and many servers. This data will be used to test Hypotheses 2 (*no geographical patterns in failures*) and 4 (*some failures are persistent*).

VPS in China: We rented a VPS in China. The system was located in Beijing (AS 23028) and was used for our SYN backlog scans discussed in Section 3.2.1. Our VPS provider employed a transparent and stateful TCP proxy in front of our VPS which silently dropped unsolicited segments. We carefully implemented our SYN backlog scans so they first established state whenever necessary to be unaffected by the TCP proxy. These SYN backlog scans provide a dataset that speaks to our assumptions about how China blocks Tor. It complements the hybrid idle scan data set because, although the measurements are from a single client in China, it allows us to see exactly how that client experiences the censorship. This data will be used to test

Hypotheses 1 (*RSTs are treated the same as SYNs*) and 3 (*server to client blocking*).

Tor relay in Sweden: We used a long-established Tor relay at Karlstad University in Sweden for our traceroute measurements discussed in Section 3.3.2. The relay has been part of the Tor network for several months, and using our VPS we manually verified it to be blocked in China. This data set shows blocking between one Tor relay and many clients in China. It complements the hybrid idle scan data set because access to the Tor relay allows us to collect more details about the blocking. This data will be used to test Hypotheses 4 (*some failures are persistent*), 5 (*some failures have diurnal patterns*), and 6 (*blocking is in the backbone*).

We now present our probing infrastructure as well as our measurement methodology used to investigate the theories posited in Section 3.1.

Hybrid idle scans: Recall that by using hybrid idle scans, we have more freedom in choosing clients in different regions to test their reachability to different servers. Our goal is to determine blocking of Tor relays (outside of China) from the perspective of a large and geographically diverse set of clients (within China).

We are interested in knowing whether there exist different experiences of the censorship of Tor for different users in different regions. The previous chapter showed that a small fraction of all Tor relays was accessible from a single vantage point in Beijing, but what about the rest of the country? Key questions are: how does the GFW's architecture and China's routing affect censorship in different regions?

IP address selection: We selected clients in China (CN), North America (NA), and Europe (EU). In order to be able to select random IP addresses in China without favouring specific locations—especially large cities featuring a vast number of allocated IP addresses—we divided the map of China into $33 * 65$ cells corresponding to one degree of latitude and longitude. We filled this grid with all IP addresses in MaxMind's database that were documented to be in China. Then, we collected IP addresses by randomly selecting a cell from our grid after checking that they employed global IPIDs. In an analogous manner, clients from the EU and NA were chosen by horizontally scanning these regions. After 24 hours, we gathered a pool of IP addresses that belonged to machines with a global IPID. Then, we continually checked the selected IP addresses for a 24-hour period to discard IP addresses that changed global IPID behaviour, went down, or were too noisy. At the end we had 11 NA, 7 EU, and 161 CN clients to use for our measurements.

Servers were chosen from three groups: Tor relays, Tor directory authorities, and web servers. Tor relays were downloaded from the Tor Status [47] website. We only selected relays with an uptime greater than five days. In order to select Tor relays in geographically diverse regions, we selected 10 Tor relays from Europe, 13 from the United States, 20 from Russia, and 101 from other countries. This way, our selected Tor relays were not biased toward Europe or the U.S., which exhibit more relays per capita than other regions. The 10 Tor authorities were obtained from the Tor source code. Web servers were chosen randomly from Alexa's top 50 websites in China [48]. All web server and Tor relay IP addresses were checked hourly to make sure that they stayed up for at least 24 hours before being selected for our measurement.

The geographic distribution of our Tor relays as well as all clients in China is illustrated in Figure 3.4.

Creating a complete bipartite graph: We used three machines at UC (our university campus) to run the hybrid idle scan experiments. We started the experiments with 180 clients and 176 servers. Each day 20 clients and approximately 20 servers were selected for each of the machines. For 22 hours[1], every hour, we performed the hybrid idle scan for each possible pair of client and server. Every "scan round" performs: 1) two minutes of hybrid idle scans, 2) 30 seconds of sending RSTs to clear the server's backlog, and 3) five seconds of testing the client to assure that they remained online and kept their global IPID. Similar checks are performed to ensure that servers remain online throughout each experiment. At any given time, each IP address (client or server) was involved in only one test. After 27 days, each client's reachability was tested to all servers, i.e., *our clients and servers created a bipartite graph.* For more details about the experiment design refer to Ensafi et al. [14, 40].

Pruning the data: We used the selected IP addresses throughout our experiments. Naturally, some of the hosts went down or were occasionally

[1] Two hours per day were reserved for server data synchronisation.

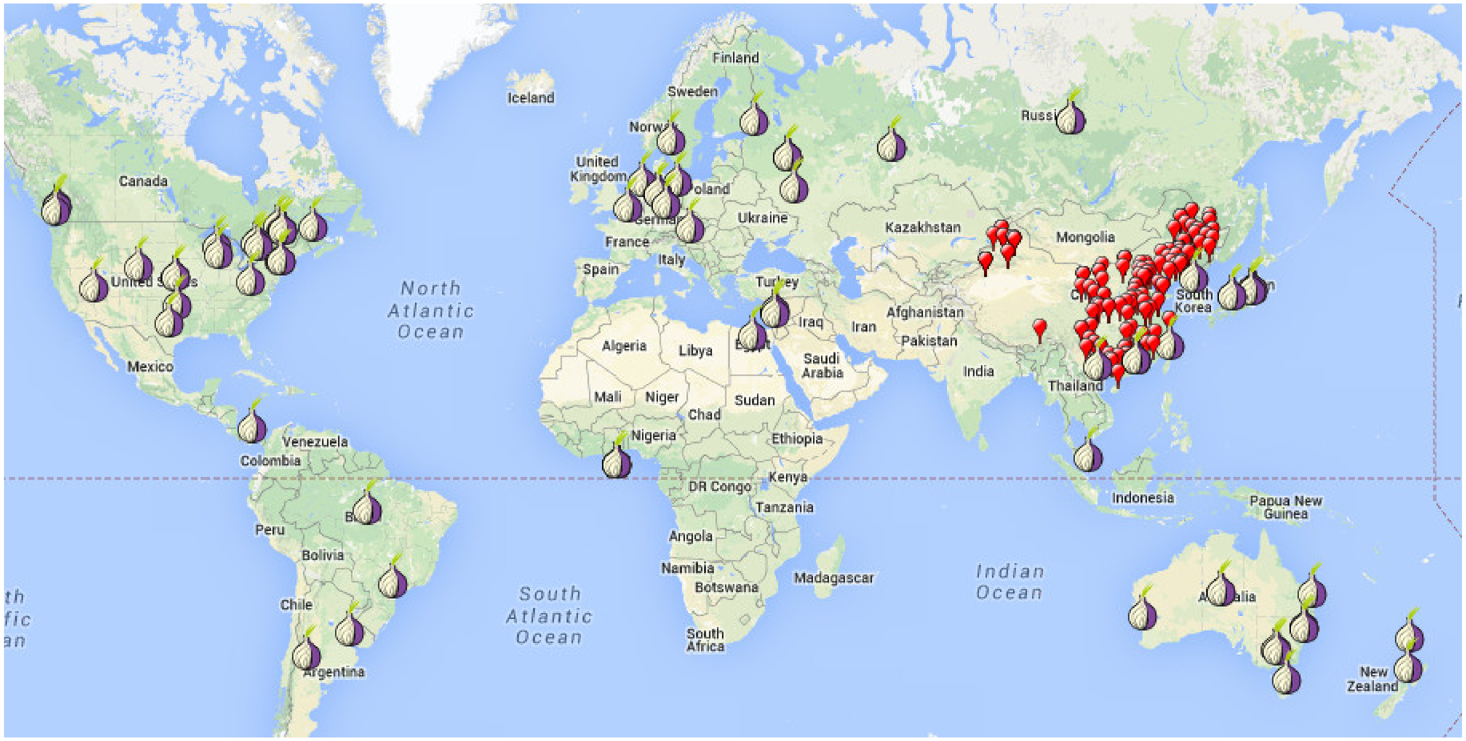


Figure 3.4: The geographic distribution of all tested Tor relays (shown as onions) and of our global IPID clients in China (shown as red marks). Note that outside of Xinjiang the west of China has very little Internet penetration, which is why we have few data points in this region and the distribution is biased towards the eastern parts of China.

too noisy. Also, the host behind an IP address can change, e.g., a client with a global IPID might lose its dynamic host configuration protocol (DHCP) lease and get replaced with a client running a random IPID. To account for these issues, we perform tests throughout our experiments which cull out data points where basic assumptions are not met. For every server involved in the experiment, we had two checks: liveliness and the stable Tor flag test. After each scan, for five seconds we sent five SYN segments per second using UC's unbound IP address. The data point passed the liveliness test only if it retransmits three or more SYN/ACKs. Also, if the server was a Tor relay, we verified that the relay was assigned the "Stable" flag (cf. Section 3.3.1).

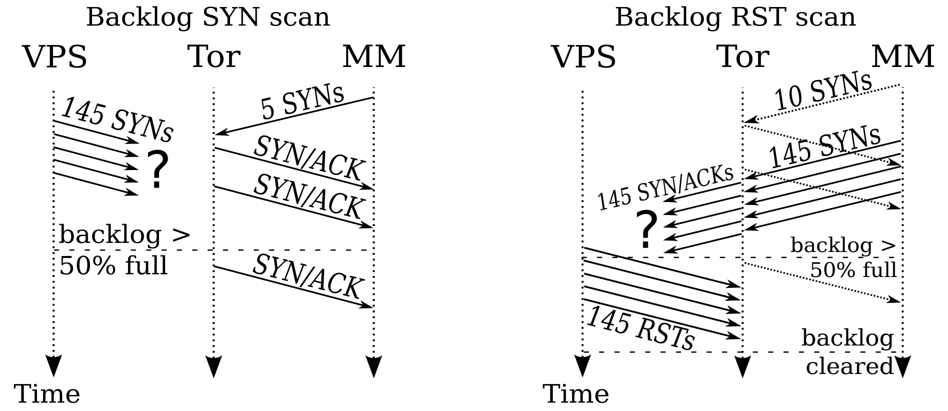
For every client, for five seconds, we sent five SYN/ACKs per second using UC's unbound IP address. We expect the client to respond with RST segments totaling in number to more than half the number of sent SYN/ACKs. If this is the case then the data point passes the client's liveliness test. The results of a scan were allowed into the data set only if both the client and server passed their checks. Note that each data point is one client and one server tested one time in a given hour. There was a several-hour network outage that caused a hole in a portion of one day of our data.

After culling out data that did not meet our basic assumptions, we were left with 36% of the total data collected. This 36% is the data described in Section 3.4 and used for our analysis.

Backlog scans After having presented the underlying side channel in Section 3.2.1, we now discuss the implementation of our two backlog scan types which can answer two questions, 1) "Do SYN segments from China reach a Tor relay?" and 2) "Do RST segments from China reach a Tor relay?". Basically, we answer both questions by first transmitting crafted TCP segments to a relay, thus manipulating its SYN backlog, and then querying its backlog size by counting the relay's SYN/ACK retransmissions. The conceptual implementation of both scan types is illustrated in Figure 3.5.

SYN scan: The SYN scan—depicted in Figure 3.5(a)—is started by MM by sending five SYN segments to Tor in order to infer the relay's backlog size when under stress.[2] After a delay of approximately 500 ms, VPS proceeds by sending 145 SYN segments whose purpose is to fill the relay's backlog by

[2] We transmit five SYN segments rather than just one to account for packet loss.



(a) SYN scan to infer whether SYN segments from VPS reach Tor. “MM” is our measurement machine.

(b) RST scan to infer whether RST segments from VPS reach Tor. “MM” is our measurement machine.

Figure 3.5: The two types of backlog scans we employ. The purpose of these scans is to verify if 1) SYN segments from China reach a Tor relay and if 2) RST segments from China reach a Tor relay.

more than half. Recall that the backlog size defaults to 256, so we only fill the backlog to 59%. That way, we can make the Tor relay's kernel prune MM's SYN segments, thus reducing their retransmissions. Finally, MM knows that VPS's SYNs reached the relay if the number of SYN/ACK retransmissions for its five SYNs is lower than five. Otherwise, VPS's SYNs did not reach the relay. This type of inference is necessary because most of the time, China's GFW drops SYN/ACKs from known Tor relays.

RST scan: Our RST scan incorporates an additional step but is based on the same principle. As illustrated in Figure 3.5(b), MM starts by sending 10 SYN segments whose purpose is, analogous to the SYN scan, to monitor the relay's backlog size. Afterwards, MM proceeds by sending 145 spoofed SYN segments with VPS's source address. Note that we cannot send the SYN segments from VPS as they might be blocked. By sending spoofed SYN segments from an unfiltered network link, we can ensure that the segments reach the Tor relay. Upon receiving the SYN segment burst, the relay replies with SYN/ACK segments which we expect to be dropped by the GFW. In the final step, VPS sends a burst of RST segments to the Tor relay. The RST segments are crafted so that every RST segment corresponds to one of the relay's SYN/ACK segments. The purpose of the RST burst is to terminate all half-open connections, thus clearing the relay's backlog. Based on how many retransmissions we observe for the 10 "probing SYNs", we can infer whether the RST segments were dropped by the GFW or not. Receiving five retransmissions means that the backlog was not cleared and the RST segments were dropped. Receiving less than five retransmissions means that the backlog was successfully cleared and the RST segments were not dropped by the GFW. This kind of inference is necessary because machines outside China cannot measure directly what happens to RST packets sent from China, and machines inside China are very limited in their ability to infer what is happening on blocked IP address/TCP port pairs.

Implementation: We implemented our scans using a collection of bash scripts and a patched version of the tool hping3 [49]. Accurate timing was crucial for our experiments. To keep the clock of our machines synchronised, we used the tool ntp which implements the network time protocol. Recall that the SYN backlog behaviour we are exploiting is limited to Linux kernels (cf. Section 3.2.1). As a result, our scans targeted the subset of 94 out of our 144 Tor relays which are known to run Linux. Tor relays periodically publish their server descriptors—which includes their operating system—to all directory authorities so there is no need for us to guess the operating system of Tor relays.

Pruning the data: By pruning the backlog scan data, we aim to make sure that the relay runs an unmodified Linux TCP/IP stack. After scanning a relay, we send three "baseline SYNs" to it in order to query its original amount of SYN/ACK retransmissions. First, we discard scans in which the relay never sent five SYN/ACK retransmissions, Linux's default value since version 2.2. For example, we found embedded Linux relays which always retransmit SYN/ACK segments four times, regardless of their backlog size. Second, we also discard scans whose SYN/ACK retransmissions do not exhibit Linux's exponential backoff behaviour. Third and finally, we discard scans where the relay was offline or other networking problems occurred. These three pruning steps discarded 774 out of all 2,094 scans (37%).

Traceroutes into China: We want to learn if there are *unfiltered routes* leading into China. To investigate this question, we used our Tor relay in Europe to run traceroutes to numerous destinations in China. After a country-wide scan, we obtained a list of 3,934 IP addresses in China that responded to SYN/ACKs and were distributed geographically in a diverse way, which served as our traceroute destinations. For every IP address, we ran two TCP traceroutes; one whose TCP source port was equal to the filtered Tor port 9001 and one whose TCP port was set to the unused and unfiltered port 9002. The traceroutes had both their SYN and ACK bit set. We used a slightly modified version of the tool hping3 [49] to run the traceroutes as it allowed us to send TCP segments with a source port which is bound by the Tor process.[3] Starting on 4 May 2014, we ran the traceroutes on an hourly basis for two days, resulting in a total of $3,934 \cdot 24 \cdot 2 \cdot 2 = 377,664$ traceroutes. We determined where the traceroutes entered China using whois and round-trip time information. We culled out a small amount of data that did not enter China through a known backbone network, since all such data either appeared to enter China in Pasadena, California (a case we can handle but will require deeper analysis into whois records) or was destined for clients that we determined to actually be in Hong Kong.

3.3.3 Good Internet citizenship

We took several steps to devise our scans to be minimally invasive. First, we set up a web server on our measurement machines whose index page informed visitors about our experiments. The page contained our contact information to provide alarmed network operators with an opportunity to contact us and opt out of our measurements. Furthermore, we carefully designed our measurements so that it is very unlikely that they harmed any computers or networks. Throughout the lifetime of our experiments, we did not receive any complaints. We discuss ethical aspects of our measurements in Section 3.5.3.

[3] We modified the tool to constantly increase the TTL of outgoing TCP segments. The default behaviour is to wait for every hop to reply with a “TTL exceeded” ICMP message.

3.4 Analysis and results

We now analyse the three data sets we gathered; the hybrid idle scans, the backlog scans, as well as the traceroutes into China.

3.4.1 Hybrid idle scans

The hybrid idle scan data was collected from 15 March 2014 to 10 April 2014. One client was removed from the data because we determined that it was in Hong Kong and as a result not subject to the GFW's filtering.

Table 3.1 shows the results of our hybrid idle scans. The column $S \rightarrow C$ is short for **Server-to-client-dropped**, *None* means **No-packets-dropped**, $C \rightarrow S$ means **Client-to-server-dropped**, and *Error* simply means **Error**. In the table's rows, *CN* is short for China, *EU* means Europe, and *NA* means North America. As for the server types, *Tor−Dir* is a Tor directory authority, *Tor−Relay* is a Tor relay, and *Web* is a web server. Our results confirm that, in general, SYN/ACKs entering China from blacklisted IP address/TCP port pairs are blocked. Some web servers were censored, and some Tor nodes were censored outside China. This is to be expected because even in countries that do not perform nation-scale Internet censorship, organisations frequently take steps to filter material such as pornography or file sharing sites. Note that highly popular websites often contain material that is subject to censorship.

The most interesting result from the hybrid idle scans is that the **No-packets-dropped** case was measured all over the country without any noticeable geographic pattern. The geographic distribution of observed **No-packets-dropped** cases is shown in Figure 3.6. The case distribution closely matches the distribution of our clients which, in turn, matches the geographic Internet penetration patterns of China. This means that the failures in China's IP address/TCP port blacklisting mechanisms are not limited to one region or one network block. We provide a more thorough analysis in Section 3.4.2, which confirms Hypothesis 2.

We also observed that in many cases these filtering failures are persistent and *last throughout the day*. We witnessed four client/server pairs where all 22 measurements in a day returned **No-packets-dropped**. We redacted the clients' 16 least significant bits:

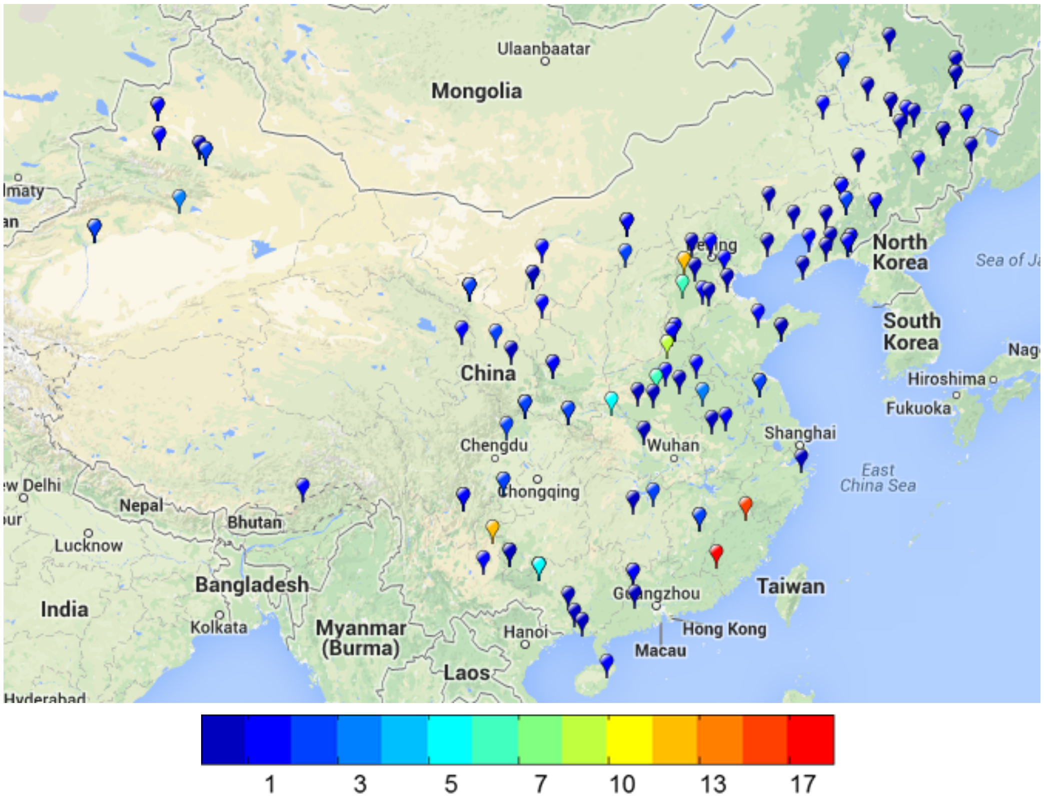


Figure 3.6: The colour temperature for clients corresponds to the number of observed **No-packets-dropped** cases over the entire experiment. No geographic or topological pattern is visible. Instead, the distribution matches the geographic Internet penetration patterns of China.

Table 3.1: Results from the hybrid idle scan measurement study.

Cli	**Srv**	$S \to C$ **(%)**	**None (%)**	$C \to S$ **(%)**	**Error (%)**
CN	Tor relay	116,460 (81.5)	555 (0.3)	786 (0.5)	25,061 (17.5)
CN	Tor dir	8,922 (64.9)	31 (0.2)	2,696 (19.6)	2,097 (15.2)
CN	Web	306 (1.2)	15,663 (62.9)	2,688 (10.8)	6,226 (25.0)
EU	Tor relay	18 (0.2)	8,589 (96.7)	22 (0.2)	245 (2.7)
EU	Tor dir	2 (0.2)	776 (96.7)	0 (0)	24 (2.9)
EU	Web	19 (1.2)	1,333 (86.2)	95 (6.1)	98 (6.3)
NA	Tor relay	45 (0.3)	11,022 (94.4)	33 (0.2)	566 (4.8)
NA	Tor dir	4 (0.3)	1,025 (94.7)	3 (0.2)	50 (4.6)
NA	Web	32 (1.5)	1,794 (85)	98 (4.6)	185 (8.7)

Client 58.193.0.0 (CN) → server 198.96.155.3 (CA)
Client 58.193.0.0 (CN) → server 161.53.116.37 (HR)
Client 58.193.0.0 (CN) → server 128.173.89.245 (US)
Client 121.194.0.0 (CN) → server 198.96.155.3 (CA)

This would give evidence towards Hypothesis 4, but our traceroute results reveal that Chinese Educational and Research Network (CERNET) does not perform the type of blocking we are measuring at all so later in this section we will discuss similar failures in commercial networks. Clients 58.193.0.0 and 121.194.0.0 are part of CERNET. Server 198.96.155.3 is a long-established Tor exit relay at the University of Waterloo. 161.53.116.37 and 128.173.89.245 are Tor relays in Croatia and the U.S., respectively. There were also many instances where client/server pairs showed **Server-to-client-dropped** for most of the day but also showed **No-packets-dropped** once or a handful of times.

3.4.2 Temporal and spatial association

We now seek to answer the question of whether there are any temporal or spatial associations among the **No-packets-dropped** cases observed for Tor relays tested from within China.

Temporal association is shown in Figure 3.7. The probabilities are com-

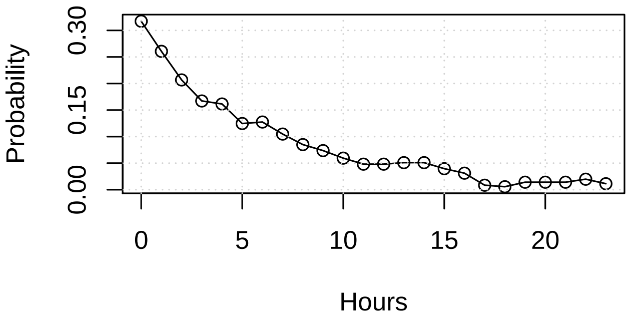


Figure 3.7: The temporal association between cases of **No-packets-dropped**. The x axis shows the amount of hours since the last **No-packets-dropped** case whereas the y axis shows the probability of observing another case of **No-packets-dropped**.

puted by a simple counting technique. We have the hourly count of the number of **No-packets-dropped** cases for each source. For each occurrence of **No-packets-dropped**, we check if there are other **No-packets-dropped** cases in the subsequent hours. We use 151 sources for this calculation, excluding the educational sources, which contained 353 **No-packets-dropped** cases in total. The final probabilities are averaged over all sources. With the increase in the lag amount in the x axis, the probability decreases. This shows that **No-packets-dropped** cases generally happen in bursts of hours.

Spatial association is shown in Figure 3.8. We use the latitude and longitude of the sources as two-dimensional coordinates. The curvature of the earth is ignored while computing the distance between sources. For every source, we find the geographically K-nearest neighbouring sources and average their count. We compute the Pearson's correlation coefficient between the count of **No-packets-dropped** cases for a source and the average of the same for the neighbouring sources. Note that Pearson's correlation has a range of -1.0 to 1.0. Our maximum observed correlation value of 0.26 is, therefore, a very weak positive correlation and supports the fact that there is no significant geographical association between sources and their neighbours. With the increase of the neighbourhood radius, the correlation decreases to below 0.1. Together with the fact that the cases of **No-packets-dropped**

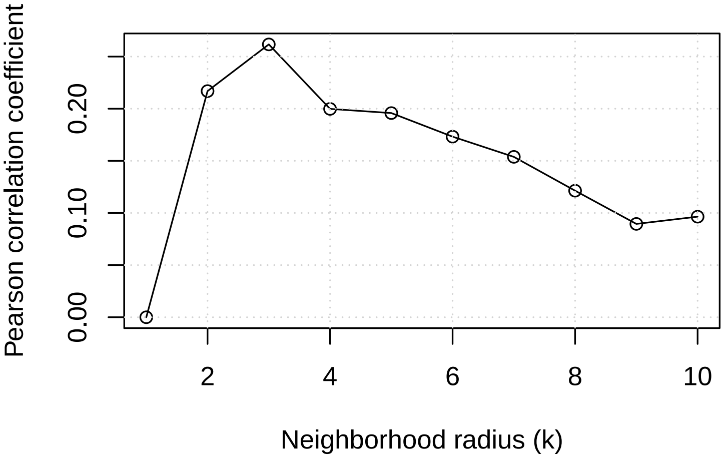


Figure 3.8: Spatial association between clients in China. The x axis shows the neighbourhood radius (k) and the y axis shows the Pearson correlation coefficient.

are distributed fairly evenly in all geographic regions (see Figure 3.6), this is strong support for Hypothesis 2.

3.4.3 SYN backlog scans

We began our backlog scans on 24 March 2014 and ran them twice a day with approximately 12 hours in between the scans until 10 April 2014. We gathered a total of 2,094 scans and after pruning, this effort yielded 1,320 scans (63%).

Out of all 1,320 backlog scans, 33 scans (2.5%) to 12 unique IP addresses contained the respective Tor relay's SYN/ACK segments, indicating that no filtering was happening. Interestingly, 19 of these 33 scans targeted the directory authority 128.31.0.39 on port 9131. Only the RST scan and not the SYN scan yielded SYN/ACKs from the directory authority.

The results in Table 3.2 show that, in general, if a RST packet passes, then a SYN packet also will. This confirms one of the basic assumptions behind the hybrid idle scan, and confirms Hypothesis 1. Also, the fact that most SYNs were allowed to pass through the GFW confirms Hypothesis 3.

Table 3.2: Backlog scan results.

	RST passes	**RST dropped**
SYN passes	666 (80%)	39 (4.7%)
SYN dropped	68 (8.2%)	53 (6.4%)

Table 3.3: The results of our traceroute measurements, which targeted educational networks.

	EDU Randport	**EDU Torport**
Stalled	1,061	1,045
Finished	428	433

3.4.4 Traceroutes

Table 3.3 and 3.4 show the results of our traceroute measurements. In the table, "EDU" indicates that the first hop in China in the traceroute is the educational and research network backbone, CERNET (210.250.0.0/16 or 101.4.112.0/24) or another scientific network which is called CSTNET (159.226.0.0/16). "COM" indicates that the first hop in China was a commercial backbone, one of: CNCGROUP (219.158.0.0/16), China Telecom/CHINANET (202.97.0.0/16), China Mobile Comm. Corp. (211.136.1.0/24 or 221.176.23.0/24), or the China Telecom Next Carrying Network backbone (50.43.0.0/16). All other entry points were thrown out because they were actually in Hong Kong or Pasadena, and that usually indicated that the destination IP address was not in China or non-Chinese routing hops had not been properly culled. "Tor" means that the source port of the SYN/ACKs sent in the traceroute was the Tor port, and "rand" means that the source port was another port that the GFW does not filter. Thus, "Tor" traceroutes should always stop before the destination host if the filtering is effective on that route, and "rand" should reach the destination unless there are other types of filtering in play, such as ICMP filtering or firewalls not related to censorship. The elements in the table are the number of times that a traceroute reached all the way to the destination.

Table 3.4: The results of our traceroute measurements, which targeted commercial networks.

	COM Randport	**COM Torport**
Stalled	111,133	163,095
Finished	53,479	429

Surprisingly, the educational and research networks, in particular CERNET, do not seem to be implementing this type of filtering at all. The "Tor" and "rand" columns are nearly identical for the "EDU" traceroutes. The "COM" traceroutes, however, show that commercial networks are clearly censoring Tor by dropping SYN/ACKs. The "rand" traceroutes reached their destination 53,479 times, while the "Tor" traceroutes aimed at the same destinations only reached the destination end host 429 times. Similar to the hybrid idle scan results, these failures were all over the country and for any destination IP address where at least one failure was observed, the number of failures ranged from 1 to 48 (i.e., all 48 hours of measurements). The number of failures in the most prominent destinations where the traceroute entered China on a commercial background included one instance where 48 failures were observed and two where 47 were observed. This means that sometimes the failures are relatively persistent, confirming Hypothesis 4.

Figure 3.9 shows the amount of hops into China, filtered "Tor" port traceroutes traversed before stalling. For each measurement of each hour of each day, we only add the data to Figure 3.9 if the "rand" traceroute reached the destination and the "Tor" traceroute did not. In most cases, the filtered packets make it two hops into China, confirming Hypothesis 6.

Figure 3.10 shows the number of failures for traceroutes that entered China on the commercial network backbone, per hour. The diurnal patterns apparent in the figure confirm Hypothesis 5. Note that 02:00 UTC is 10:00 (or, 10:00 am) in Beijing.

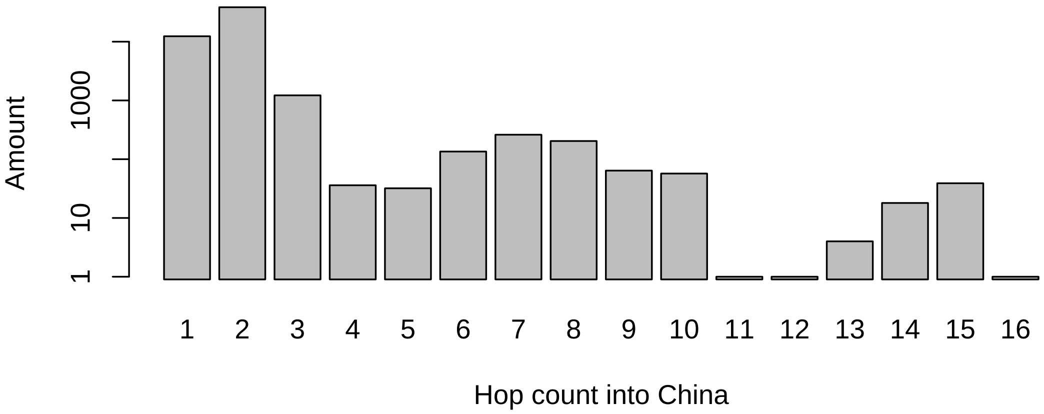


Figure 3.9: The amount of hops (log scale) in China, our filtered traceroutes could traverse. For example, a hop count of five means that a traceroute could successfully reach the fifth router inside China.

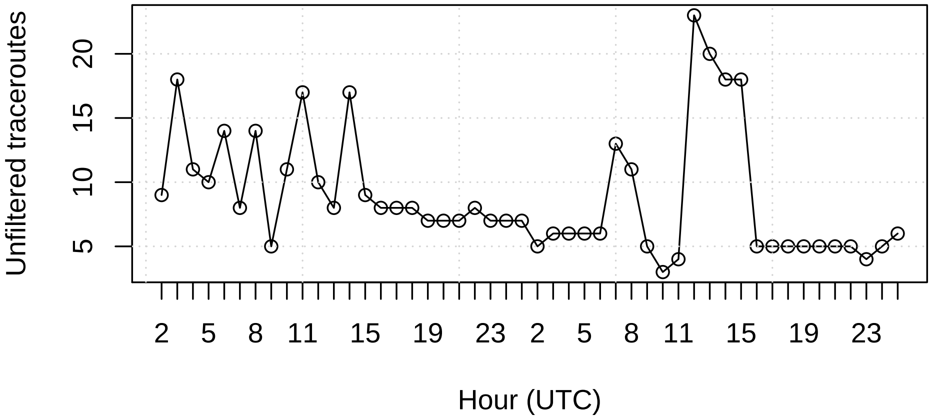


Figure 3.10: The amount of unfiltered traceroutes from our Tor relay to clients in China over time. A diurnal pattern is visible.

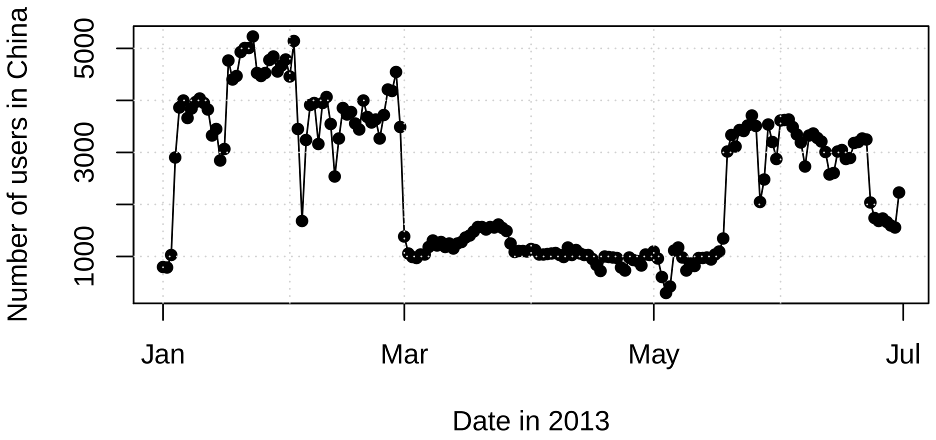


Figure 3.11: The amount of directly connecting Tor users from China over the first seven months of 2013. The diagram shows several spikes and a "valley" in between March and May.

3.5 Discussion

We discuss three different aspects of our work in this section: what we learned about the filtering of Tor in China, what we learned about the architecture of the GFW, and ethical considerations.

3.5.1 Filtering of Tor in China

Our results suggest that the filtering of Tor in China has several interesting aspects, some of which may even be useful for circumvention efforts. We showed that the failures in the filtering occur in every part of the country, and they are sometimes *intermittent* and sometimes *persistent*. A historical example of intermittent failures is illustrated in Figure 3.11. The diagram shows the amount of directly connecting Tor users in China in the first seven months of 2013. A relatively stable "valley" in between March an May is clearly visible. This valley is surrounded by significantly higher usage numbers.

We also showed that this type of filtering does not occur on CERNET, the educational and research backbone of China's Internet. This might suggest that CERNET users can reach the Tor network, or it might suggest that CERNET employs a more sophisticated method for detecting and interfering with connections to the Tor network, perhaps something stateful and based on deep packet inspection.

Our results raise additional questions such as "is it possible to run a Tor relay in China?". In general, the Tor network represents a complete graph. As a result, every relay should be able to connect (and generally maintain connections) to all other relays in the network. Furthermore, relays must be able to connect to the directory authorities in order to upload their server descriptors. If CERNET is indeed whitelisted, a Tor relay inside CERNET might be able to successfully join the Tor network. In addition, Chapter 2 suggested that domestic Tor traffic in China is not subject to blocking [32]. If filtering indeed happens at the Internet exchange point (IXP) level, as suggested by our data, it is not surprising that the GFW is generally unable to filter domestic network traffic as it typically does not reach IXP level[4] and is of significantly higher volume than international traffic. As a result, functioning Tor relays or bridges inside CERNET might be able to connect users in China to the rest of the Tor network.

3.5.2 The architecture of the GFW

Our results also shed light on the architecture of the GFW, at least with respect to the mechanism that blacklists IP address/TCP port pairs. As discussed in Section 3.2, the three theories about how the GFW is architected are that 1) the filtering occurs at choke points where undersea cables enter the country, 2) the filtering occurs in the backbone in large IXPs, and 3) the filtering occurs at a regional level. While our results show some filtering occurring many hops into China and some filtering occurring before packets can even enter China, the majority of the filtering happens about two hops into China (presumably at the large IXP in Beijing). Thus, Hypothesis 6 is most consistent with the theory that the filtering occurs in the backbone. Note that this observation is in accordance with other recent research efforts which focused on the GFW's DNS injection [50]. The small amount of routes

[4] We ignore routing phenomena such as "boomerang routing".

that are filtered at the provincial level, which were also observed by Xu et al. [20], can be explained by the strategy employed by China's formerly second-largest ISP, CNCGROUP, which was recently bought by the largest (CHINANET).

While whitelisting would appear as persistent failures in the filtering and the filtering apparatus getting overloaded with traffic would appear as intermittent failures, the mix of intermittent failures and diurnal patterns with persistent failures suggests that routing is a major reason why the filtering fails. Hypotheses 4 and 5 are most consistent with the theory that the filtering occurs in the backbone, because provincial networks in China are very hierarchical [51] and undersea cables are few in number [52]. Hypothesis 2 is also most consistent with backbone-level filtering for this reason.

3.5.3 Ethical considerations

Our work has two ethical considerations that need to be discussed. First, our SYN backlog scans briefly fill a Tor relay's backlog in order to be able to observe packet drops. A full backlog can prevent a relay from accepting new TCP connections or cause the use of SYN cookies which can lead to reduced throughput. To prevent relays from using SYN cookies, we adapted our scan parameters to minimise the risk of completely filling a relay's SYN backlog. SYN cookies typically do not support scaled flow control windows, which is why we made every effort to avoid them. In general, the rate at which we are sending SYN packets, without intention of completing a connection, is not enough to create a denial-of-service condition on any modern network stack. For an interesting discussion about ethical issues related to port scans in general, we refer the reader to Durumeric et al. [53].

Second, our idle scans create unsolicited traffic between a client and a server. This traffic—which can be observed by the censor—only consists of SYN/ACKs from the server to the client and RSTs from the client to the server. As a result, we are not causing any meaningful communication other than background noise as it is also caused by port scanning activity. While one may conceptualise the hybrid idle scan technique as providing the ability to conscript a client into performing tests for us, in reality the traffic between the server and the client is no different from if the server chose to send SYN/ACKs to the client. Thus, in terms of the traffic that the censor sees, the hybrid idle scan technique is no different from if Tor relay operators performed simple connectivity measurements by directly sending SYN/ACKs.

3.6 Conclusion

In this chapter, we have characterised the mechanism that the Great Firewall of China uses to block the Tor network using a hybrid idle scan that can measure connectivity from the perspective of many clients all over China. We have also presented a novel SYN backlog idle scan that can infer packets received by a server without causing denial of service. These novel Internet measurement techniques open up whole new possibilities in terms of being able to measure the Internet from the perspective of arbitrary clients and servers. This is extremely important when it comes to characterising and documenting Internet censorship around the world, because of the difficulty in finding volunteers geographically dispersed throughout a country.

We also evaluated our techniques which led to several new insights about the inner workings of the Great Firewall. Our data shows that 1) at least several machines inside CERNET (China Education and Research Network) are able to connect to Tor relays, 2) filtering seems to be centralised at the IXP level, and 3) filtering is quite reliable with the Tor network being either almost completely reachable or almost completely blocked in different parts of the country.

Chapter 4

User-aided censorship analysis

4.1 Introduction

Censorship evasion can be seen as a *feedback loop*: circumvention tool developers learn from blocks and adapt their tools whereas censors also learn from circumvention tools and refine their blocking techniques. This feedback loop is, however, *highly asymmetric*, meaning that circumvention tool developers learn significantly less than censors. For example, the Tor project is traditionally very open in its circumvention work [39]. Code, designs, and data are available and discussed in public. Censorship systems such as the GFW, on the other hand, are a black box. Typically, these black boxes are probed by developers and researchers in order to unravel their inner workings. The purpose of the censorship analyser discussed in this chapter is to *reduce this information asymmetry*. That way, we strive to ease censorship analysis and boost circumvention tool development.

The task of censorship analysis requires a way to observe the respective censorship system. Unfortunately, analysis is not always possible from outside the censoring networks. While Chapter 3 proposed side channels to measure packet dropping from outside a censoring network, our techniques are limited to the TCP and IP layer. Given these shortcomings, two popular and alternative analysis strategies are to either 1) obtain network traffic traces for manual inspection and/or to 2) gain access to machines inside the censoring regime. Both of these approaches can turn out to be difficult in

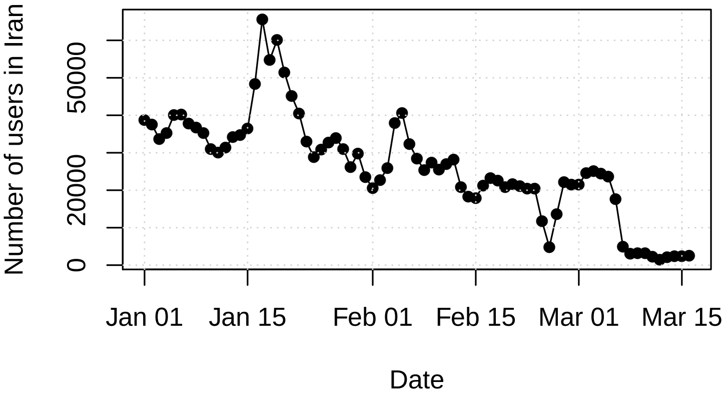


Figure 4.1: The amount of Iranian Tor users since the beginning of 2013 [54]. In February 2013, the country started to deploy a new strategy to block Tor.

practice. Network traces require the cooperation of users in the censoring country and are difficult to anonymise which poses a problem of operational security. Further, remote access to machines is problematic due to the lack of volunteers or appropriate proxies such as PlanetLab [55], open SOCKS proxies or VPSs for rental.

The above should highlight that there is a strong need for a lightweight and easy to use tool which can assist in the process of analysing blocking incidents. In a nutshell, this tool should be run by volunteering users within the censoring country and conduct a number of networking tests in order to gain an understanding of how and if Tor is blocked in a given country or network. The analysis report should then reach the Tor developers in a safe way and thus facilitate circumvention and documentation [39]. The main contribution of this chapter is the design and software architecture for such a lightweight censorship analyser for the Tor anonymity network.

4.2 Requirements and design

At first glance, one would expect a censorship analyser to gather as much data and conduct as many experiments as possible. In practice however, this can be a bad idea since the analyser will be run by non-technical computer users

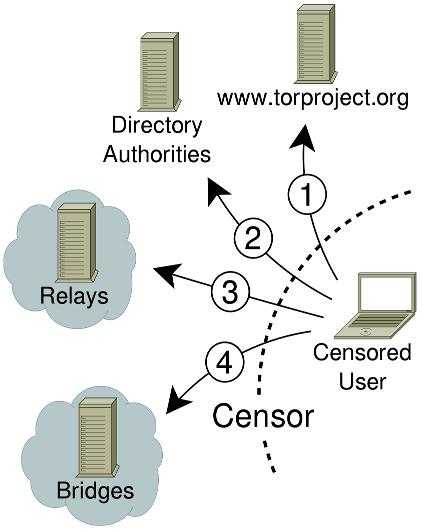


Figure 4.2: The Tor censorship analyser 1) probes the official website, 2) tries to download the consensus, 3) tries to connect to relays, 4) and to bridges.

who might even face repercussions for running such a "noisy" tool. Besides, users should not be linkable to the data, their copy of our tool generated. As a result, it is important to find a balance between *meaningful data* and respecting the *privacy and security* of users volunteering to run the analyser.

In Section 4.2.1 and 4.2.2, we will enumerate a variety of features we consider desirable in a Tor censorship analyser. These features are motivated by prior experience in debugging censorship incidents as well as by preventively probing for what strategies censors might implement next. Naturally, some features are harder to implement than others. Therefore, the list is organised in ascending order based on the difficulty of the respective feature. While these features were designed specifically for Tor, some might be interesting for other circumvention tools as well.

4.2.1 Analysis-centric requirements

Network trace of analysis

Optionally, the analyser should be able to capture a network trace in pcap format. A network trace enables minute inspection of censorship techniques such as spoofed TCP reset segments injected into the user's connection. The trace must be limited to the network traffic generated by our censorship analyser and must not contain network traffic other than that. While clearly useful, pcaps contain IP addresses and can pose a threat to users having strong threat models. Techniques such as Crypto-PAn [56] could be used to pseudonymise IP addresses but packet payload could still contain identifying information. As a result, this feature should be turned off by default. If a user decides to activate it, she should be warned that she gives up a large part of her anonymity by doing so.

Difficult to detect

An effort should be made to avoid obvious network activity which would make it straightforward for censors to isolate users running our analyser. Obfuscation strategies should involve random sleep periods between and during network tests and randomising the order of executed tests. We also note, however, that a fully undetectable analyser would be very difficult to implement; it would include emulation of typical user behaviour as well as steganography to transmit the final report without censors noticing.

Probe the website

The official website, www.torproject.org, is sometimes found to be blocked in censoring countries [39]. With the website being unreachable, users should be prevented from downloading the Tor Browser Bundle (TBB) [57]. To detect website blocks, the analyser should try to download the index page by emulating an HTTP GET request of a modern browser such as Firefox or Chrome. We note that it is important to craft the GET requests to resemble one of these browsers. Otherwise, the GET requests could fail if censors whitelist user agents. If the website is believed to be blocked—because the index page could not be fetched or the TLS connection failed—the following subsequent analysis steps are performed.

First, the domain www.torproject.org must be resolved by querying the user's configured DNS server. The returned IP addresses (if any) are then compared to the expected IP addresses which are hard-coded in our analyser. That way, we can detect DNS poisoning. Other open and "well-behaved" DNS servers such as Google's 8.8.8.8 could further be queried. That way, we hope to be able to detect transparent DNS rewriting assuming that we trust 8.8.8.8 to return the right DNS records.

Second, another website request should be issued for one of the official Tor mirrors. In particular, for mirrors without the strings "tor" or "torproject" in their domain name. While the official website might be blocked, mirrors could very well be reachable.

Third and finally, in a subsequent step the analyser should verify whether a TCP connection to port 443 of www.torproject.org is possible. Besides, a TLS connection with a modified server name identifier (SNI) could unravel censorship based on SNI inspection in the TLS client hello.

The same method should be applied for bridges.torproject.org which is used to distribute bridges (cf. Section 4.2.1) to users who find themselves unable to connect to the public Tor network.

Probe the directory authorities

In March 2013, nine directory authorities served the consensus for the Tor network. These authorities are a natural *choke point*. If a client is unable to contact any of them, a direct connection to the public Tor network can not be bootstrapped. Therefore, the authorities are sometimes blocked on the IP layer (cf. Section 2). To detect such censorship attempts, our analyser should be able to connect to the authorities and try to download the consensus. Should the analyser be unable to contact any of the hard-coded authorities, two subsequent checks should be performed.

First, ICMP echo requests should be sent to the authorities. Most of the authorities reply to ping requests. If no reply is received, this could be an indicator of blocking. Note that this test could easily yield false positives if the user's network disallows ICMP in general.

Second, traceroutes should be run to the potentially blocked authorities. In case of filtering, traceroutes could help narrow down the hops on the network path responsible for the block. Further, traceroutes can be conducted using a variety of protocols such as TCP, UDP, and ICMP to increase the likelihood of this test to succeed.

Relay reachability

After a client successfully downloaded the consensus, the next step in bootstrapping a Tor connection consists of selecting relays which together form a circuit. The first relay in a circuit—the so-called entry guard—must be reachable for the client.

Another straightforward way to block access to the Tor network lies in periodically downloading the consensus and blacklisting all IP addresses found within. In fact, it is sufficient to blacklist only the entry guards. We can easily detect this censorship strategy by trying to initiate a TLS connection to several entry guards. However, it would also be interesting to learn whether connections to pure middle or exit relays succeed. If not, then this could be an indicator that a censor is blindly blacklisting all IP addresses found in the consensus.

Connections to entry guards can fail for a number of different reasons. Perhaps the most popular strategy among censors is to terminate or drop TLS handshakes which appear to be Tor-specific. Over time, Tor's TLS handshake exhibited a number of distinguishing elements such as the client cipher list [32, 58], the server certificates as well as the randomly generated SNIs. TLS-based filtering was or is done in Ethiopia [59], China [32], and Iran [60], just to name a few. To detect Tor-specific TLS filtering, a Tor-specific TLS client hello could be sent to machines which are not Tor relays; for example the host behind mail.google.com. Inferring the exact pattern matched by DPI boxes is a difficult task and was sometimes done manually [58]. An automated way would be highly desirable but is beyond the scope of this chapter.

Bridge reachability

Analogous to ordinary Tor relays, *bridges* should be tested as well. Recall that bridges are relays which are not listed in the public Tor consensus. Instead, they are distributed to censored users out-of-band and serve as "hidden" stepping stones into the Tor network.

Furthermore, by using obfsproxy, it is possible to obfuscate the network traffic exchanged between clients and bridges [24]. In order to detect DPI boxes targeting obfsproxy's pluggable transports, the analyser should be able to conduct handshakes using various pluggable transports and see if they succeed.

Gather debug information

Censorship is typically not homogeneous across a country and can differ on the level of provinces, autonomous systems or even network prefixes [61]. As a result, we are interested in information which can help narrow down the respective censorship infrastructure. Also, this would help ruling out interferences and prevent jumping to wrong conclusions. Of interest would be the following information.

1. What ISP does the user have? Our analyser could obtain the whois record for the user's IP address and discard all IP address material which would otherwise reveal the user's location.

2. What is the autonomous system number? Open IP-to-ASN lookup services could be used [18].

3. Is the user behind a captive portal? This can be difficult to verify and might require a number of heuristics.

4. Is all traffic forced to go through an HTTP proxy?

Anonymising reports

Naturally, reports should not be linkable to users submitting them. We have to distinguish between *report content* and *report submission*.

The actual report is straightforward to anonymise. As an option, we can easily discard identifying information in the report by, e.g., never logging the user's IP address and redacting the first hops in traceroutes. A high degree of anonymity could be achieved by completely discarding all IP addresses, prefixes and location information such as autonomous system numbers and whois records. Network captures in the form of pcap files are very difficult to anonymise and should also be completely discarded if strong anonymity is desired.

Problematic however, is report submission. It is the task of tools such as Tor to provide anonymity on the address layer. Unfortunately, we cannot make use of Tor as the very purpose of our tool is to find out why Tor is unreachable. Users could be advised to generate one-time email addresses for submission over email. That way however, the email provider would learn the user's IP address.[1] This would place a lot of control into the hands of an untrusted third party so another—possibly better—option is automatic submission using an hyper text transfer protocol secure (HTTPS) POST request to infrastructure run by the Tor Project. This method could be easier to block and once again, this cannot disguise the user's IP address, unfortunately.

4.2.2 User-centric requirements

While the analyser's main purpose is to gather censorship-related data, it is important to recognise that it will frequently—if not mostly—be run by users who *lack technical expertise*. These users will not be able to configure the analyser manually or run tools on the command line. This observation motivates the following requirements.

User-friendly output

We emphasise that the primary purpose of the analyser is to gather data which can assist Tor developers. Still, as a side task, the analyser should inform users about the gathered data and results while avoiding too much

[1] The content of the report can be protected when the report is encrypted using the hard-coded OpenPGP public key (cf. Section 4.3.1).

technical jargon. Based on a decision tree inferred from the gathered data, the analyser could inform the user about possible ways to circumvent the respective censoring network.

Cover the tracks

As mentioned above, it is crucial to protect users' privacy. This implies that temporary files or reports must not be stored in non-obvious locations on the user's hard drive. All temporary data must remain in the unpacked directory containing the analyser. That way, a user can easily delete the entire directory and by doing so also delete gathered data.

Nevertheless, it is difficult to hide the fact that a certain program was executed [62]. Even more so on Windows. What can be hidden is the analysis results by simply deleting them after submission.[2] The very existence of our analyser is difficult to disguise, however.

We argue that *usage diversity* can mitigate the threat arising from simply having a copy of our tool. Users could state that they were simply interested in finding out why their TBB would not bootstrap (cf. Section 4.2.2) rather than in assisting in circumventing their national firewall.

Ease of use and informed consent

It is crucial that the analyser is as easy to use as possible. In particular, it should be a self-contained click-and-go executable, just like the TBB. Ease of use also involves the analyser's *bundle size*. Ideally, the analyser would only be a few megabytes in size which would also make is suitable for email distribution over GetTor [63].

Users should be informed in clear words that the analyser is performing network probing and that the gathered data can be submitted afterwards for further inspection. Users should be given the opportunity to abort the process prior to network probing and prior to submission. Note that internationalisation of our analyser will also be necessary since it is unrealistic to expect users to have a decent command of the English language.

[2] Note that simple file removal might not be sufficient when file system forensics is conducted.

4.3 Software architecture

There is no need to reinvent the wheel. The analyser will be implemented as a set of tests for OONI [64]. OONI is a framework for network censorship analysis and provides a Python API which can be used to develop all of the requirements discussed above. In addition, OONI defines a custom YAML format—YAMLOO—for analysis reports. At the time of writing OONI cannot, however, be invoked by running a single executable. As a result, our analyser must be packaged together with OONI to a single Windows executable. This can be done using tools such as py2exe [65]. The following sections discuss a number of architectural considerations.

4.3.1 Communicating results

Eventually, the data produced by a user's analyser—mainly a text file containing YAMLOO data—has to reach the Tor developers. We believe that it would be short-sighted to define a single mechanism for the transmission to happen. This would create a single point of failure which might turn out to be easy to block for censors.

Instead, we envision several reasonable communication channels. Ideally, we would send the report in an anonymous fashion and upload it to a hidden service. But as already discussed in Section 4.2.1, we expect Tor to be censored and not an option for this. A report could be sent manually over email. While this requires user interaction, some sort of email is typically available; even in countries with pervasive censorship. In particular, free services such as GMail are frequently reachable over HTTPS.

Another option would be to automatically transmit the report using an HTTPS POST request to a web server operated by the Tor project. While single web servers are straightforward to block, we could increase collateral damage by running the web server inside a cloud provider, the censor is hopefully unwilling to blacklist. Other communication channels could include instant messaging programs or steganographic publishing systems. Unfortunately, all of these methods have in common that the user's IP address will eventually be known by a provider of some sort.

Furthermore, the report should contain a *message digest* which is calculated over the YAMLOO report. This is particularly important when the report is being sent over email since it makes it possible to detect if the report is incomplete or the user accidentally changed parts of it.

Finally, the analyser should contain a hard-coded OpenPGP public key which can be used to encrypt analysis reports. Users who decide to encrypt the report should still have the opportunity to review a report prior to encryption and submission. While encryption comes at the cost of key management, we believe that the additional management overhead is worth the increase in security in certain situations.

Configurable and testable during build

It should be possible to pass configuration parameters to the analyser during the build process. That is necessary because certain information will be subject to change. This includes hard-coded IP addresses of relays or the web servers hosting the domain www.torproject.org. If we would be unable to change these parameters, a censor could simply whitelist the hard-coded IP addresses and render our analysis results useless.

All the features discussed in Section 4.2 should be testable in an automated way. Otherwise, we might end up shipping code which does not work in real environments or we might not notice if improvements break existing code.

4.4 Discussion

We point out that motivated and strongly equipped censors would be able to identify users trying to run our analyser. Such censors would also be able to actively falsify our analyser's results. Being undetectable would require a substantially more complex concept. However, we believe that most censors are more motivated to spend their resources on actual blocking technology rather than measurement and analysis technology. Of course, this will change if our analyser should turn out to be a very valuable tool in the arms race.

There are two additional important features not discussed here due to page constraints. First, our analyser could be enhanced to be able to infer the patterns used by DPI boxes to identify protocols. According to our own experience, DPI boxes do not always just use simple regular expressions but also context-sensitive languages. This is, however, a difficult problem which might require a client-server architecture which runs a grammatical inference algorithm. Precise knowledge of the patterns used for censorship would enable targeted circumvention and could be fed into tools such as format-transforming encryption [66].

Second and equally difficult, our analyser could have features to detect the type or model of DPI box used for censorship. This would make it possible to expose cases similar to the use of Bluecoat in Syria [67]. While well-designed DPI boxes not exposing their management interface can be difficult to identify, it might be possible to define a number of heuristics to cluster DPI boxes; perhaps based on injected TCP reset segments (cf. [68]).

4.5 Conclusion

In this section, we outlined the design requirements for a censorship analyser for the Tor anonymity network. While the majority of these requirements discuss network probing techniques, we also consider usability aspects. We plan to implement our concept in the next step.

We believe that involving ordinary computer users in the process of censorship analysis can greatly increase the coverage of censorship documentation and shed light on previously unknown types of censorship. Great care must be taken, however, to not exploit users and inform them about the process. Informed consent must be achieved whenever possible. Furthermore, we hope to start a discussion about how to safely integrate non-technical users into the process of censorship measurement.

Chapter 5

Global and longitudinal censorship detection

5.1 Introduction

The previous chapters proposed several methods to detect Internet filtering. Ideally, the analyst has direct control over a censored source host that can perform measurements against an external destination. This is typically not the case, however, so research often has to opportunistically resort to open proxies, the help of volunteers (see Chapter 4), and existing measurement platforms. All of these methods have advantages and disadvantages. Open proxies suffer from low network coverage, are unreliable or questionably reflect typical conditions, and are often limited to TCP streams or HTTP requests. Cooperation with volunteers exposes individuals to potential harm and is time-consuming. Current, specially-designed censorship measurement platforms suffer from limited deployment and insufficient maintenance. Therefore, in order to develop representative and real-time perspectives of interference, we build a prospective mechanism on top of an existing, widely-deployed measurement platform, the RIPE Atlas network [69].

Measurement and analysis of information controls is a non-zero sum development effort. Existing platforms, such as PlanetLab [55], Herdict [70], and OONI [64] are complementary and provide unique perspectives on the diverse forms of interference. We believe that an interference analysis platform based on Atlas can provide an additional perspective to the bigger picture, one whose strengths are wide deployment, rapid results, and the foreshadowing of broader community lessons. Toward these objectives, we make the following contributions.

- We evaluate the aptitude of the RIPE Atlas platform for analysis of information controls and propose an algorithm to balance timeliness, network diversity, and cost, in order to facilitate effective analysis.
- We apply the platform and algorithm for monitoring of ongoing filtering events across different countries, and provide results based on several months of measurements.

5.2 Framework structure

In order to assess Atlas's aptitude as an interference measurement platform, we continue by presenting available data collection mechanisms and our analytical framework.

5.2.1 RIPE Atlas background

Founded in 2010 by RIPE NCC, Atlas [69] is a globally distributed Internet measurement network consisting of physical probes hosted by volunteers. Once a user connects her probe to the network, it can be used by other participants for measurements. So-called *credits* are awarded automatically based on the uptime of contributed probes, which are expended in order to perform custom measurements. Queries to probes can be initialized centrally either over the web frontend, or over a RESTful API.

An ideal measurement platform features high geographic and topological diversity, thereby facilitating measurements in any region where filtering occurs. While Atlas probes are distributed throughout the world, there is a significant bias towards the U.S. and Europe as can be seen in Figure 5.1. As for Atlas's topography, only 68 autonomous systems contain 40% of all Atlas probes with the three most common autonomous system numbers being:

AS7922: 4.4% of probes, owned by Comcast Cable Communications

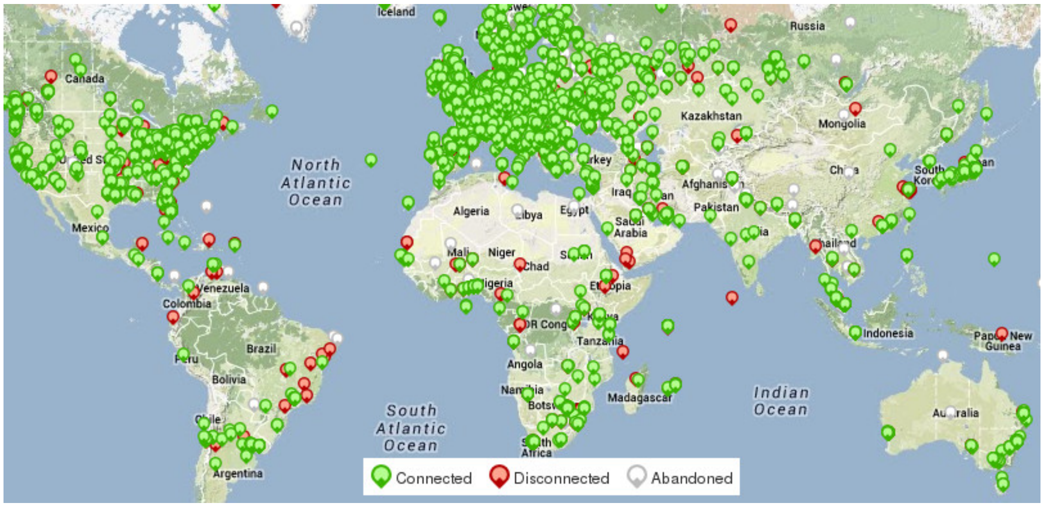


Figure 5.1: The geographic distribution of Atlas probes as of May 2014. Green icons represent active probes whereas red icons represent probes which are currently offline. The distribution is heavily biased towards the U.S. and Europe.

AS3320: 3.2% of probes, owned by Deutsche Telekom

AS6830: 2.8% of probes, owned by Liberty Global Operations

While not optimal, most regions of particular interest still contain at least several probes.

As of May 2014, Atlas allows four types of measurements; ping, traceroute, DNS resolution, and X.509 certificate fetching (henceforth simply called SSLCert). All four measurement types can further be parameterized for more fine-grained control. HTTP requests are not possible at this point due to abuse and security concerns. While Atlas clearly lacks the flexibility of comparable platforms (see Table 5.1), it makes up for it with high diversity, responsiveness, and continued growth. After all, we do not expect Atlas to replace existing platforms, such as OONI, but rather to *complement* them.

Table 5.1: Qualitative comparison between several popular censorship analysis platforms. Platforms are compared by their measurement flexibility, probe coverage, and what they are mainly used for.

Platform	**Flexibility**	**Coverage**	**Main use**
PlanetLab [55]	High	Low/Medium	Network measurements
Atlas [69]	Low	Medium/High	Network measurements
M-Lab [71]	Low	High	Network measurements
Tor [3]	Medium	Medium	Low-latency anonymity
OONI [64]	High	Low	Interference analysis
Herdict [70]	Low	Low/Medium	Interference analysis
OpenNet [13]	Low	Medium	Interference analysis

5.2.2 Atlas's cost model

As previously mentioned, Atlas measurements are paid with platform credits. The exact "price" of a measurement depends on the measurement type, its parameters, and the number of destinations. The credit system works based on a linear cost model. Each user has a credit balance that can be increased steadily by hosting Atlas probes[1] or by receiving credits from other users.

Table 5.2 lists the currently available measurement types as well as their associated costs. While DNS and SSLCert measurements have a fixed cost, ping and traceroutes vary depending on the amount and sizes of packets. Also, one-off measurements cost twice as much as repeated measurements. When scheduling a new experiment, the user first specifies the details (e.g., measurement type as well as measurement parameters). Afterwards, Atlas's web-based frontend calculates the measurement costs on the server side and shows it to the user. Finally, upon completion of the measurement, the respective cost is subtracted from the user's credit balance.

Due do the non-deterministic nature of pings and traceroutes, and measurements in general, we developed a command-line based tool to help users create new measurements and estimate their costs.[2] As input, the tool ex-

[1] As of June 2014, 21,600 credits per day of uptime.

[2] The tool is available at http://cartography.io.

Table 5.2: The cost of all available Atlas measurements. The variable N refers to the number of packets whereas the variable S refers to packet sizes.

Measurement	**Cost in credits**
DNS/DNS6 (TCP)	20
DNS/DNS6 (UDP)	10
SSLCert/SSLCert6	10
Ping/Ping6	$N * (int(S/1500) + 1)$
Traceroute/Traceroute6	$10 * N * (int(S/1500) + 1)$

pects 1) a country of interest, 2) the amount of credits the user is willing to "pay", and 3) a measurement type. Our tool then determines the amount of available probes (if any), the expected costs, and runs the measurement if the cost is below the user's expected cost.

5.2.3 Assessing measurement integrity

Despite their distribution across a diversity of countries and networks, RIPE Atlas may not fully reflect the Internet as it is experienced by the general public, as probes neither fully emulate the network position nor the configuration of an average user. As an immediate control, efforts are taken to verify the reachability of non-controversial content and identify whether probes use domestic domain name servers. These probes may be excluded from measurements in order to avoid Type 1 and Type 2 errors.

Even with additional precautions, idiosyncratic observations are an inevitable product of the high rate of placement of probes on commercial and academic networks. These institutions may have alternative connectivity that is faster and less highly regulated than consumer networks. Additionally, disrupting or degrading connections based on plain text data, traffic classification or application headers would fall outside of the measurements currently possible with Atlas probes. Lastly, Atlas is unlikely to detect content restrictions imposed by the platforms themselves, such as search manipulation or withholding of content based on a user's location.

5.2.4 Rough consensus validity

The international distribution of web services, such as content delivery networks, has created additional complexity in the determining whether measurement results are genuine. While secure socket layer (SSL) and domain name system security extensions (DNSSEC) utilize third-party trust to validate answers, Certificate Authorities have been previously compromised by state and non-state actors, and DNSSEC is not widely implemented. In order to validate answers within the Atlas network, we use a cross-country comparison of results to queries. This methodology assumes that intermediaries who interfere with connectivity do not coordinate strategies internationally. States and service providers impose different filtering approaches for purposes of localization, infrastructure or even the monetization of blocked traffic. Furthermore, interference is more effective when the public is unaware of the practices and technologies employed against them, placing a strong incentive on secrecy. Therefore, as a simple test of validity, we count the number of countries or ASs that an answer, such as a DNS A record, final network of transit or a certificate hash, is seen. Any response with fewer than the mean number of jurisdictions, or those within private network spaces (RFC 1918 [72]), are treated as potentially aberrant and flagged for further investigation.

5.2.5 Ethical aspects

Atlas was not designed explicitly for analysis of information controls and accordingly, its volunteers likely may not expect that their probes will be used for such purposes. Careless measurements could attract attention and cause repercussions for probe operators. In addition, an increased used of Atlas for politically-sensitive analysis could scare away probe operators and jeopardize the usefulness of the platform. These concerns extend beyond merely complying with Atlas's acceptable usage policies and guide our selection of measurements.

Atlas's measurement types are limited in scope. As of May 2014, it is not possible to create HTTP requests or engage in actual, meaningful communication with arbitrary destinations, which limits the damage caused by reckless measurement. We are not aware of environments where low-level network queries to commonly frequented platforms or services solicit attention from authorities, even when blocked, nor where answers are falsified in order to stifle research. Requests for sites such as Facebook are generated as a part of normal web use due to script inclusion, and Google Public DNS is commonly used due to its reliability. Commonplace sites are a different class of potential monitoring targets than content promoting child abuse or violent extremism.

The balance between research interests and exposure to risk is an area of concern. This has stimulated a broader discussion that will play a factor in future utilization of Atlas. Nevertheless, we stress that great care must be taken when planning measurements because the volume and types of measurements could still be suspicious. We believe that Atlas has a place in the domain of censorship analysis but it has to remain a small place, lest it endangers users or the platform itself. For a more comprehensive ethical discussion, see Wright et al. [61, § 5].

5.3 Case studies

After having presented our measurement platform and parameters, built on top of Atlas, we now evaluate it by discussing two cases of large-scale restrictions to online content and social media. In particular, Turkey's ban on media platforms in Section 5.3.1 and Russia's filtering of opposition LiveJournal content in Section 5.3.2. All dates and times reported follow Coordinated Universal Time.

5.3.1 Turkey's ban of Twitter

In late March, social media users began to report limitations on the availability of Twitter across Turkey's Internet Service Providers. YouTube and Twitter had both become the target of condemnation by Prime Minister Recep Tayyip Erdoğan in preceding months. By March 20, the Turkish government's Information and Communication Technologies Authority (BTK) mandated the filtering of Twitter across the country's service providers.

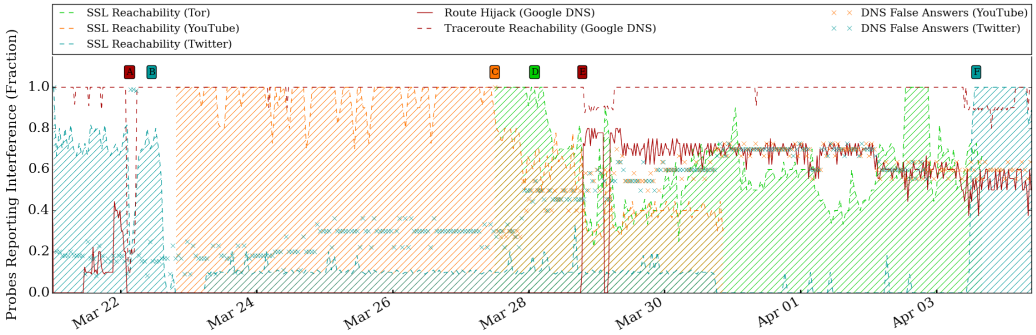


Figure 5.2: The disruption of social media platforms in Turkey from March to April 2014.

Turkey's Internet filtering has previously been characterized as DNS tampering and IP address blocking [73], which both fall under the measurements possible through Atlas. Upon news of the Twitter ban, we scheduled hourly measurements of local DNS answers, SSL connectivity, and traceroute reachability for Twitter, YouTube, Google Public DNS and the Tor Project through ten probes, covering nine ASNs. The selected measurement targets sought to longitudinally document the Turkish government's disruption of controversial political content, identified based statements by authorities and potential use for circumventing controls. Seeking to address an immediate interest for real-time awareness, the measurements did not attempt to assess the whole of the country's content restrictions. As illustrated in Figure 5.2, we found at least six shifts in content restrictions and blocking strategies within a two week period.

While the BTK and compliant ISPs rely on DNS manipulation and IP blocking, it appears that the former is more popular. As of April 24, 2014, the Turkish-language anti-censorship site Engelliweb [74], which tracks blocked content, only lists 167 IP addresses restricted in country, compared to 40,566 domain names. In absence of address blocking or HTTP filtering, users that received valid DNS answers for Twitter's domain names could browse without further interference. As a result, foreign DNS servers quickly became both a circumvention mechanism and a political statement, with the addresses of alternative services offered by Google and OpenDNS reportedly graffitied across the the country in protest of the ban.

On the morning of March 22 (see Figure 5.2, Event A), between 01:00 and 02:00, backbone providers Tellcom İletişim Hizmetleri and Türk Telekom began disrupting Google Public DNS service through the IP address blocking of its two prominent addresses (8.8.8.8 and 8.8.4.4). By 06:00 the same morning, the DNS blocking had been removed across all ISPs. Instead, to buttress the restrictions, providers shortly began to drop all outgoing traffic to IP addresses associated with the twitter.com domain, regardless of port or provider (Event B). By 16:00 of that day, no Atlas probe could directly negotiate an SSL connection with Twitter until the removal of the ban nearly two weeks later.

On March 27 (Event C), after recordings were posted of Turkish national security officials discussing possible military action against Syria, YouTube was blocked through false DNS answers for the youtube.com domain. Within the Atlas network, this restriction appears as a slow decline in the number of probes able to establish a connection to the platform. However, unlike Twitter, a significant minority of probes remained able to communicate with YouTube. Google's intertwined infrastructure presents risk of collateral damage with network prefix restrictions, which were not present with Twitter. Thus, clients that were able to receive a valid address could reliably bypass the ban.

Beginning March 28, Turkish probes began to fail to establish SSL connections to torproject.org (Event D). However, this restriction neither included IP address blocking, nor apparent interference with the accessibility of the actual Tor network. Atlas probes could continue to negotiate valid connections to Tor's directory authories. Throughout the increased manipulation of local DNS services, nearly half of the Atlas probes remained connected due to their use of foreign DNS services.

Later in the evening, March 28, hosts querying foreign-based DNS servers began to receive the same false answers as those provided domestically, leading to a rapid drop in availability of YouTube and Tor (Event E). A publicly-available traceroute scheduled by third-parties on the Atlas network against Google Public DNS returned idiosyncratic and spontaneous shifts in Turkey's network topology timed with these changes. This appears within

Table 5.3: The cost of the measurements discussed in Section 5.3.1. A daily total of 19,200 credits was used.

Target	**Type**	**Probes**	**Freq (s)**	**Credits**
Twitter	SSL	10	3,600	2,400
YouTube	SSL	10	3,600	2,400
Tor	SSL	10	3,600	2,400
Twitter	DNS (U)	10	3,600	2,400
YouTube	DNS (U)	10	3,600	2,400
Twitter	Tracert	10	3,600	7,200
Total (Daily)				19,200
Probes required				0.89

traceroutes as a shortening in the number of hops to Google, with a multifold reduction in traffic latency and the absence of international hosts in path. The core telecommunications provider Türk Telekom had begun to reroute traffic destined for Google to a local DNS server serving false answers. Only TEKNOTEL Telekom maintained consistently valid routes for Google, through Telecom Italia Sparkle. However, two days later Doruk İletişim and Net Elektronik Tasarım reestablished connectivity through Euroweb Romania, circumventing upstream interference. Türk Telekom's redirection was finally removed late on April 7.

By April 3, despite continued hijacking of Google Public DNS and interference with YouTube, Twitter was unblocked for all probes (Event F). The total measurement credits we spent in order to conduct this experiment are shown in Table 5.3.

5.3.2 Private sector cooperation

On March 13, 2014, Russia's Federal Service for Supervision in the Sphere of Telecom, Information Technologies and Mass Communications (Roskomnadzor) ordered the blacklisting of opposition figure Alexei Navalny's LiveJournal blog.

At the same time, independent media portals were filtered, including the news site grani.ru [75]. Similar to Turkey, Internet filtering in Russia is frequently conducted by IP address blocking and DNS poisoning [76, 77]. However, with a random sample of 255 probes across 147 ASs in Russia, only 38 probes on 20 ASs received *aberrant* DNS answers for Grani. Within this subset, probes received a diverse, consistent selection of *ten unique addresses*, including two within private network address space (10.52.34.222 and 192.168.103.162). A greater selection, 40 probes across 23 ASs, of traceroutes to port 80 for the primary address associated with Grani (as of April 30, 23.253.120.92) failed within Russian network space.

In contrast to Grani's domain, a locally-resolved DNS query for the domain navalny.livejournal.com over 255 probes on 146 ASs received a consistent reply of 208.93.0.190, which matched answers internationally with only one anomalous response, a formerly valid address. The blocking of Navalny's blog must be different from Grani. While the returned DNS A record of 208.93.0.190 falls within a network prefix owned by LiveJournal Inc. (208.93.0.0/22), over the 1,462 LiveJournal subdomains in Alexa's Top 1 million list, 1,450 blogs resolved to another address, 208.93.0.150. Based on requests made independently of the Atlas network from Europe, both hosts appear to be front servers for the LiveJournal platform, as they return the same SSL Certificate and content. Requests to 208.93.0.150 with a HTTP Host header set to navalny.livejournal.com retrieves the correct content, and non-blacklisted content is retrievable through 208.93.0.190.

As of April 2014, only five subdomains on livejournal.com could be found whose DNS A records resolved to the address 208.93.0.190, Table 5.4, four of which are listed within Alexa's top sites. All the blogs found on this alternative host have been publicly declared by Russian authorities as in violation the country's media laws for the promotion of political activities or extremism, and two are listed within publicly-available filter site lists.

Based on timing, filtering lists, available domain names records, and Atlas network measurements, it appears that a host was specially established to faciliate Russian restrictions on content within the LiveJournal platform. Using HTTPS Ecosystem Scans as a metric of accessibility [78], the LiveJournal frontend at 208.93.0.190 came online between February 10 and February 17, with the address otherwise unused until then. Two months later, the

Table 5.4: LiveJournal DNS A Records of 208.93.0.190.

Subdomain	**Language**	**Roskomnadzor**
drugoi-nnover	Russian	Yes
m-athanasios	Russian	Yes
imperialcommiss	Russian	Yes
pauluskp	Russian	Yes
navalny	Russian	Yes

Ukrainian LiveJournal blog “Pauluskp” (pauluskp.livejournal.com), which had covered Russian involvement in Crimea, was filtered with the administrative order listing an IP Address of 208.93.0.190. However, as recently as six days before, the blog was recorded as pointing to the main LiveJournal host. Similarly, the movement of Navalny’s blog was noticed within social media [79]. It appears that in the lead up to or at the time of filtering orders, LiveJournal coordinates with authorities to alter the DNS A record for blogs designated by Roskomnadzor, in order to segregate blacklisted content from the rest of the platform.

Segregated LiveJournal content and blacklisted IP addresses are subject to an additional, unknown method of network-layer interception performed within the backbone network of Rostelecom (AS12389). While blog content is not accessible over HTTPS, frontend hosts for LiveJournal offer SSL services for the purpose of securing the transmission of user credentials. On April 28, 78 of 343 Russian probes returned either irregular responses or failed to connect to the alternative LiveJournal host by address. Of this subset, 40 probes on 29 ASs returned SSL certificates with common name or locations fields attributed to Russian ISPs. Based on HTTPS data, the four aberrant certificates captured have been seen previously on seven Russian addresses belonging to the State Institute of Information Technologies, Rostelecom and Electron Telecom Network. Three of these hosts are responsive by their alternative, public address and still match certificates. Two are generic ISP homepages and one notifies of the blocking of the site ‘rutracker.ru.’ Other measurements that are unresponsive could be indicative of port blocking or the redirection of traffic to a server that is not listening for SSL connections.

The invalid certificates indicate that an intermediary in transit has redirected the traffic out of its expected path to a third-party server controlled by Russian entities. This approach is different from the normal man-in-the-middle (MitM) injection of responses seen in countries such as Iran and Syria, and highlights the potential for Russian ISPs to falsify content or gather user credentials. The observed behavior is not limited to protocol or port, although the end host appears to be only responsive to TCP requests as can be seen in Figure 5.3. This holistic interference across Rostelecom's downstream peers suggests redirection at the network layer, rather than application-based classification of traffic associated with deep packet inspection. Moreover, adjacent addresses within the same network, such as the normal frontend for LiveJournal, traverse a valid international path. Instead, blacklisted traffic appears to be coerced into a path controlled by Rostelecom, indicating a narrowly-crafted interference with normal routing through false advertisements or forwarding.

5.4 Conclusion

This chapter discussed the aptitude of the RIPE Atlas platform—which was originally meant for Internet measurements—for censorship analysis. We believe that the platform is a valuable tool to shed light on Internet censorship provided that ethical standards are upheld. Our case studies resulted in evidence suggesting that a government is cooperating with a content provider in order to facilitate filtering. This is an important finding which should guide threat modelling of circumvention tools.

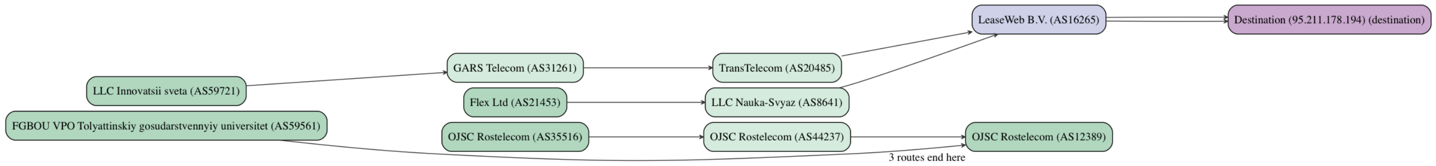

LLC Innovatsii sveta (AS59721)
FGBOU VPO Tolyattinskiy gosudarstvennyiy universitet (AS59561)
GARS Telecom (AS31261)
Flex Ltd (AS21453)
OJSC Rostelecom (AS35516)
TransTelecom (AS20485)
LLC Nauka-Svyaz (AS8641)
OJSC Rostelecom (AS44237)
LeaseWeb B.V. (AS16265)
OJSC Rostelecom (AS12389)
3 routes end here
Destination (95.211.178.194) (destination)

Figure 5.3: Rostelecom's AS12389 hijack of grani.ru traffic in April 2014.

Chapter 6

The ScrambleSuit protocol

6.1 Introduction

We consider DPI harmful. While originally meant to detect attack signatures in packet payload, it is ineffective in practice due to the ease of evasion [26, 80, 81]. At the same time, the previous chapters showed that DPI technology is increasingly used by censoring countries to filter the free flow of information or violate network neutrality [82]. We argue that what makes DPI particularly harmful is the *asymmetry of blocking effectiveness*, i.e., it is hard to stop motivated and skilled network intruders but very easy to censor ordinary user's Internet access. DPI technology ultimately fails to protect critical targets but succeeds in filtering the information flow of entire countries as evidenced by the previous chapters in this thesis..

Numerous well-documented cases illustrate how DPI technology is used by censoring countries. Amongst others, China is using it to filter HTTP [16] and rewrite DNS responses [34]. Iran is known to use DPI technology to conduct surveillance [83]. In Syria, DPI technology is used for the same purpose [84]. Even more worrying, TLS interception proxies, an increasingly common feature of DPI boxes, are used to transparently decrypt and inspect TLS sessions which effectively breaks the confidentiality provided by TLS.

Circumvention tools are meant to evade country-wide DPI installations. We argue that many circumvention tools—Tor included—suffer from two shortcomings which can easily be exploited by censors. First and most im-

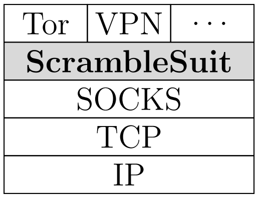


Figure 6.1: ScrambleSuit's protocol stack.

portantly, they are vulnerable to *active probing* as discussed in Chapter 2: recall that the GFW is able to block Tor by first looking for potential Tor connections based on the TLS client cipher list. If such a signature is found on the wire, the GFW reconnects to the suspected Tor bridge and tries to "speak" the Tor protocol with it. If this succeeds, the GFW blacklists the respective bridge. Active probing is not only used to discover Tor but—as we will discuss—also VPNs [85] and obfs2 [86], which is a censorship-resistant protocol. The relevance of active probing attacks is emphasised by the work of Durumeric et al. [53]. By conducting fast Internet-wide scanning, the authors were able to find approximately 80% of all active bridges at the time. From a censor's point of view, active probing is a promising strategy which greatly reduces collateral damage caused by inaccurate signatures. The attack is also non-trivial to defend against because censors can easily emulate real computer users.

Second, circumvention tools tend to exhibit a specific "flow signature" which typically remains static and is the same for all clients and servers speaking the protocol. An example is Tor's characteristic 586-byte signature (see Section 6.3.3). If a censor manages to deploy high-accuracy classifiers trained to recognise these very flow signatures, the respective protocol could easily be spotted and blocked. Most circumvention tools are not polymorphic which makes them unable to survive such filters.

In this work, we present ScrambleSuit; a blocking-resistant transport protocol which tackles the two above mentioned problems. ScrambleSuit defines a thin protocol layer on top of TCP which provides lightweight obfuscation for the transported application layer protocol. As shown in Figure 6.1, ScrambleSuit is independent of its application layer protocol and works with any application supporting SOCKS. As a result, we envision ScrambleSuit to be used by, amongst other protocols, Tor and VPN to tackle the GFW's most recent censorship upgrades. In particular, ScrambleSuit exhibits the following four features:

Pseudo-random payload: To an observer, ScrambleSuit's entire traffic is computationally indistinguishable from randomness. As a result, there are no predictable patterns which would otherwise form suitable DPI fingerprints. This renders regular expressions for the purpose of identifying ScrambleSuit useless.

Polymorphic: Despite the pseudo-random traffic, a censor could still block our protocol based on flow characteristics such as the packet length distribution. ScrambleSuit is, however, able to change its shape to make it harder for classifiers to exploit flow characteristics.

Shared secret: We defend against active probing by making use of a secret which is shared between client and server and exchanged out-of-band. A server only talks to clients if knowledge of the secret is proven by the client.

Usable: We seek to maximise ScrambleSuit's usability. Our protocol integrates in Tor's existing ecosystem. Furthermore, the moderate protocol overhead discussed in Section 6.4 facilitates comfortable web surfing.

Blocking-resistant protocols can be split into two groups. While the first group strives to *mimic typically whitelisted protocols* such as HTTP [87], Skype [88, 89] and email [90], the second group aims to *look like randomness* [91–93]. Randomised protocols have the shortcoming of not being able to survive a whitelisting censor.[1] Nevertheless, we decided in favour of randomising because mimicking comes at the cost of high overhead and we consider whitelisting on a nation scale—at least for most countries—unlikely even though it is often done in corporate networks and some countries appear to be experimenting with the concept [95, 96]. While at some point,

[1] The same fate can meet mimicked protocols if the censor decides to block the cover protocol. Oman is documented to block Skype [94] which would render Skype-based circumvention protocols useless.

our protocol might indeed become victim of a country's whitelisting policy, we believe that our approach provides a fast alternative in many censoring countries unwilling to go that far. So instead of maximising obfuscation while maintaining an acceptable level of usability, we seek to *maximise usability* while keeping an *acceptable level of obfuscation*.

We finally point out that unblockable network protocols do not exist. After all, censors could always "pull the plug" as it was already done in Egypt [97] and Syria [98]. By proposing ScrambleSuit, we do not claim to end the arms race in our favour but rather to raise the bar once again.

The contributions of this chapter are as follows.

- We propose ScrambleSuit, a blocking-resistant transport protocol.
- We present two authentication mechanisms based on shared secrets and polymorphism as a practical defence against active probing and protocol classifiers.
- We implement and evaluate a fully functional prototype of our protocol which is publicly available under a free license.[2]

6.2 Architectural overview

ScrambleSuit is a module for obfsproxy, the Tor Project's obfuscation framework [24]. Figure 6.2 illustrates that obfsproxy acts as a proxy between the Tor client and the Tor bridge. While specifically designed for Tor, obfsproxy can be used by any application as long as it supports the SOCKS protocol.

Internally, ScrambleSuit is composed of several components which are depicted in Figure 6.3. Outgoing network data is first encrypted and then padded by the packet morpher. Before these packets are then sent over the wire, the packet delayer uses small artificial delays to disguise inter-arrival times. Finally, incoming network data is first reassembled to complete ScrambleSuit protocol messages and, after decryption, finally passed on to the local application.

[2] The prototype is available at http://www.cs.kau.se/philwint/scramblesuit/.

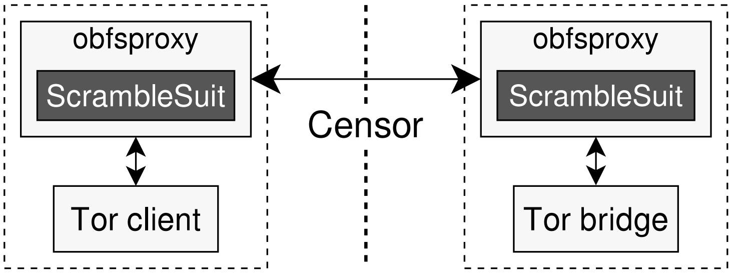


Figure 6.2: ScrambleSuit is a module for obfsproxy which provides a SOCKS interface for local applications. The network traffic between two obfsproxy instances is disguised by ScrambleSuit.

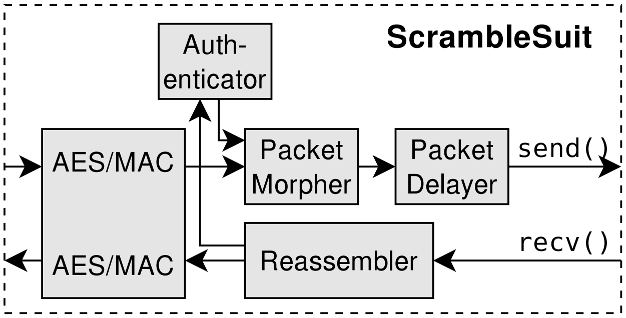


Figure 6.3: Internally, ScrambleSuit handles authenticated encryption of application data, client authentication as well as flow reshaping using a packet morpher and delayer.

6.2.1 Threat model

Our adversary is a *nation-state censor* who desires to block unwanted network protocols and services which would otherwise allow users within the censoring regime the retrieval of unfiltered information or to evade the national filtering system. The censor is making use of payload analysis, flow analysis as well as active probing to identify and then block undesired protocols.

Furthermore, the censor has full *active and passive* control over the national network. The censor can passively monitor all traffic entering and leaving its networks in line rate. We further expect the censor to actively tamper with traffic; namely to inject, drop, and modify traffic as well as hijack TCP sessions.

The censor can also select a subset of suspicious traffic for further inspection on the slow path.[3] This could, for example, involve active probing. We model our censor to also conduct active MitM attacks. While we believe that passive analysis and active probing are significantly easier to deploy, there is evidence that censors are starting to—or at least have the ability to—conduct active MitM attacks as well [99].

Our adversary is also training and deploying *statistical classifiers* to identify and block protocols. While computationally expensive, it would be imaginable that a censor uses this strategy at least on the slow path and perhaps even on the fast path when using inexpensive flow features and efficient classifiers.

Adversary limitations

We expect the censor to be subject to economical constraints. In particular, we assume that the censor is not using a whitelisting approach meaning that only well-defined protocols pass the national filter. Whitelisting implies significant *over-blocking* and we expect this approach to collide with the censor's economical incentives. We also expect the censor to not block protocols when there is only weak evidence for the protocol being blacklisted. This is a direct consequence of avoiding over-blocking to minimise collateral damage.

Finally, we assume that the censor does not have access to or can otherwise influence censored users' computers. We believe that such a scenario is likely to occur in corporate networks but not on a national scale.

6.3 Protocol design

This section will discuss ScrambleSuit's defence against active probing, its encryption and header format, as well as how we achieve polymorphism.

[3] We define the *slow path* as the minute analysis of a small traffic subset as opposed to the *fast path* which covers the majority of all network traffic and, as a result, has to be processed quickly.

6.3.1 Thwarting active probing

We defend against active probing by proposing two mutual authentication mechanisms which rely on a secret which is shared *out-of-band*. A ScrambleSuit connection can only be established if both parties can prove knowledge of this very secret. While one authentication mechanism (see Section 6.3.1) is designed to work well in Tor's ecosystem, the other mechanism (see Section 6.3.1) provides additional security and efficiency if ScrambleSuit is used by other application protocols such as a VPN.

With respect to Tor, there *already exists* an out-of-band communication channel which is used to distribute bridge descriptors to censored users. Naturally, we make use of this channel. If, however, ScrambleSuit is used to tunnel protocols other than Tor, users have to handle out-of-band communication themselves.

Proof-of-Work (again) proves not to work

Before deciding in favour of using a secret exchanged out-of-band, we investigated the suitability of client puzzles. Puzzles—a variant of proof-of-work schemes—could be used by a server to time-lock a secret. This secret can then only be unlocked by clients by spending a moderate amount of computational resources on the problem. One particular puzzle construction, namely time-lock puzzles as proposed by Rivest et al. [100], provides appealing properties such as deterministic unlocking time, asymmetric work load and inherently sequential computation which means that adversaries in the possession of highly parallel architectures have no significant advantage over a client with a single processor.

While *a single* client puzzle cannot be solved in parallel, a censor is able to solve *multiple* puzzles in parallel by assigning one puzzle to every available processing core. This is problematic because our threat model includes censors with powerful and parallel architectures. We estimated the Tor network's bridge churn rate and found that the rate of new bridge IP addresses joining the network (i.e., the amount of puzzles to solve) is not high enough to be able to increase a well-equipped censor's work load beyond the point of becoming *impractical*; at least not without becoming impractical for *clients* *as well.* This balancing problem is analogous to why proof-of-work schemes are also believed to be impractical for the spam problem [101].

In summary, proof-of-work schemes would not require a shared secret but we believe that this small usability improvement would come at the cost of greatly reduced censorship resistance. A censor in the possession of powerful computational resources would certainly be slowed down but could ultimately not be stopped. Active probing would simply become a matter of investing more resources.

Authentication using session tickets

We now present the first of our two authentication mechanisms. We envision it to be used by insecure protocols which, unlike Tor, cannot protect against active MitM attacks. A client can authenticate herself towards a ScrambleSuit server by redeeming a *session ticket.* A session ticket needs to be obtained only once out-of-band. Subsequent connections are then bootstrapped using tickets issued by the server during the respective previous connection. A real world analogy would be Alice redeeming a ticket in order to gain access to a football stadium. Upon entering the stadium (i.e., successful authentication), the guards give Alice a new ticket so that she is able to return for the next match. The same procedure then takes place for the next match. Session tickets are standardised in RFC 5077 [102] and part of TLS since version 1.0. We employ only a subset of the standard since we do not need its full functionality.

The basic idea is illustrated in Figure 6.4. The three-way handshake starts with the client *redeeming* a session ticket. The server then responds by *issuing* a new ticket. Finally, the client *confirms* the receipt of the newly issued ticket.

ScrambleSuit servers issue new session tickets $\mathcal{T}_{t+1}$ which embed a future shared master key k_{t+1} and an issue date d indicating the ticket's creation time. Session tickets are symmetrically encrypted and authenticated with secret keys k_S[4] only known to the server, i.e., $\mathcal{T}_{t+1} = \mathsf{Enc}_{k_S}(k_{t+1} \mid\mid d)$. As a

[4] For simplicity, we refer to these keys as just k_S while they are in fact two symmetric keys: one for encryption and one for authentication.

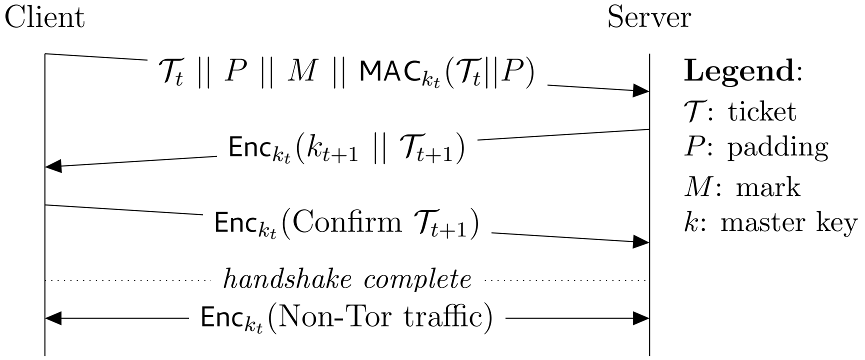

Client
Server
$\mathcal{T}_t \mid\mid P \mid\mid M \mid\mid \mathsf{MAC}_{k_t}(\mathcal{T}_t \mid\mid P)$
$\mathsf{Enc}_{k_t}(k_{t+1} \mid\mid \mathcal{T}_{t+1})$
$\mathsf{Enc}_{k_t}(\text{Confirm } \mathcal{T}_{t+1})$
handshake complete
$\mathsf{Enc}_{k_t}(\text{Non-Tor traffic})$
Legend:
$\mathcal{T}$: ticket
P: padding
M: mark
k: master key

Figure 6.4: The client redeems a valid session ticket $\mathcal{T}_t$ containing the master key k_t. The server responds by issuing a new ticket $\mathcal{T}_{t+1}$ for future use. After the client confirmed the receipt, both parties then exchange application data.

result, a ticket is opaque to the client. Note that together with the ticket, a client also has to learn the master key k_{t+1} in order to be able to derive the same session keys as the server; so clients always obtain the tuple $(k_{t+1} \mid\mid \mathcal{T}_{t+1})$ from servers.

Session tickets have the advantage that the server *does not have to keep track* of issued tickets. Instead, the server's state is outsourced and stored by clients. This reduces a server's load. To verify whether a ticket is still valid, servers simply decrypt the ticket and check the issue date d embedded in the ticket.

Whenever a client successfully connects to a ScrambleSuit server, the server issues a new ticket concatenated to the corresponding master key, $(k_{t+1} \mid\mid \mathcal{T}_{t+1})$, for the client. Recall that the ticket is encrypted and opaque to the client which is the reason why the master key is prepended. This tuple is then placed in a special ScrambleSuit control message (see Section 6.3.2) which is sent immediately after a ticket was successfully redeemed. Coming back to the real world example, Alice now has her new ticket which makes it possible for her to return for the next match.

Finally, the client confirms receipt of the new ticket by sending a dedicated confirmation message to the server. After that, the three-way ticket handshake is completed and application data can be exchanged.

Pseudo-random padding: A censor could now conduct traffic analysis by looking for TCP connections which always begin with the client sending $|\mathcal{T}|$ bytes to the server. To obfuscate the ticket's length, we introduce random padding P and authenticate the ticket $\mathcal{T}_t$ as well as the padding P by computing the message authentication code $\mathsf{MAC}_{k_t}(\mathcal{T}_t \mid\mid P)$ with k_t being the shared master key obtained by the client together with the ticket. Both parties will derive session keys from k_t as discussed in Section 6.3.2.

Locating the message authentication code (MAC): The MAC is computationally indistinguishable from the pseudo-random padding. To facilitate localisation of the MAC, we place a *cryptographic mark* M right in front of it. The mark is defined as $M = \mathsf{MAC}_{k_t}(\mathcal{T}_t)$. After extracting the ticket from the client's first chunk of bytes, the server is now able to calculate the mark and locate the MAC. We use HMAC-SHA256-128 [103] for the MAC and the mark.

Key rotation: We mentioned earlier that a ScrambleSuit server manages secret keys k_S which are used to encrypt and authenticate session tickets. This prevents clients from tampering with tickets and the server can verify that a newly received and authenticated ticket was, in fact, issued by the server. Servers rotate their k_S keys after a period of seven days. After the generation of new k_S keys, the superseded keys are kept for another seven days in order to decrypt and verify (but not to issue!) tickets which were issued by the superseded keys. As a result, tickets are always valid and redeemable for a period of *exactly seven days*; no matter when they were issued. As a result, as long as a user keeps reconnecting to a ScrambleSuit server at least once a week, *key continuity* is ensured and there is no need for additional out-of-band communication.

Foiling replay: At this point, a censor could still intercept a client's ticket and replay it. This would make the server issue a new ticket for the censor. While the censor would not be able to read the resulting ScrambleSuit control message—the shared master key k_t would be unknown—it is sound design to prevent communication without prior authentication.

We prevent replay attacks—or in other words: ticket double spending—by caching the master key k_t embedded in a ticket. If a server encounters an already cached k_t, it does not reply. We begin to cache a key k_t after a new session ticket was issued and the client confirmed that she correctly received the new ticket by using a special ScrambleSuit message type (see Section 6.3.2). We introduced the confirmation step because otherwise a censor could *preplay* a ticket, i.e., intercept it and send it *before* the client. That way, the server would add it to the replay table and would then reject the client's ticket because it would appear to be replayed. This would allow the censor to invalidate the tickets of clients. With the confirmation step, however, censors are no longer able to launch preplay attacks.

Inadequate for Tor: Session tickets provide a strong level of protection. Active probing and replay attacks are foiled but forward secrecy is not provided because the same key k_S is used for seven days to issue new tickets. As a result, we envision session tickets to be satisfactory for most application protocols which do not have these properties. Session tickets alone do not, however, integrate well with Tor's existing ecosystem. The reason lies in how Tor bridges are distributed to users. The process is illustrated in Figure 6.5. Volunteers will set up ScrambleSuit bridges which then publish their descriptors—most notably IP address, port and shared secret—to the bridge authority which then feeds this information into the BridgeDB component ①. In the subsequent step, the gathered descriptors are distributed to censored users ②. The two primary distribution channels are email and HTTPS [10]: Users can ask for bridges over email or they can visit the bridge distribution website[5] and obtain a set of bridges after solving a CAPTCHA. Finally, users can establish ScrambleSuit connections ③.

The problem is that *one bridge descriptor* is typically shared by *many users*. All these users would end up with an identical session ticket. This causes two severe problems. First, our replay protection mechanism does not allow reuse of session tickets. Only the first user would be able to authenticate herself. Second, session ticket reuse would lead to identical byte strings at the beginning of a ScrambleSuit handshake which would be a strong distinguisher. These problems lead us to an *additional authentication mechanism*, discussed in Section 6.3.1, which is optimised for Tor and can function with a secret which is shared by many users as shown in the scenario in Figure 6.5.

[5] URL: https://bridges.torproject.org

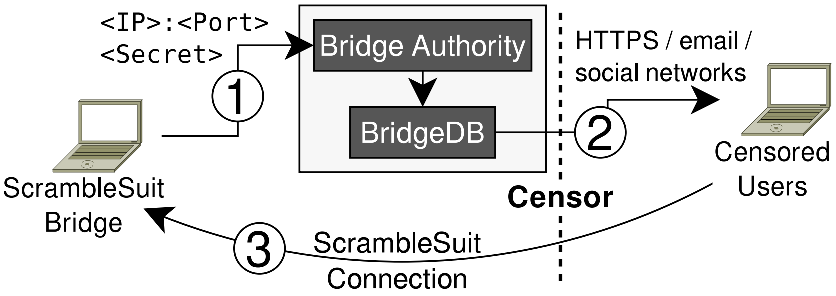


Figure 6.5: ScrambleSuit bridges send their descriptor to the Tor project's bridge authority ①. From there, it is distributed to censored users who learn about IP address, port and the secret *out-of-band* ②. Finally, direct connections can be established ③.

Tor-specific session tickets

If ScrambleSuit is used to tunnel Tor traffic (as opposed to other application protocols such as a VPN), a slightly modified session ticket handshake is used. As illustrated in Figure 6.6, the only two differences are:

1. The ticket confirmation message is *no longer necessary.*

2. The MAC in the first message *contains the variable E* holding a timestamp.

Recall that the purpose of the ticket confirmation message is to foil preplay attacks. When Tor is transported by ScrambleSuit, preplay attacks are no longer problematic because if a ticket is lost, the alternative authentication mechanism discussed in Section 6.3.1 is used instead. This possibility to "fall back" means that a lost ticket no longer implies the inability to authenticate as it does with the classical ticket scheme.

The variable E holds the current Unix timestamp divided by 3600, i.e., the number of hours which have passed since the epoch. Its purpose is to reduce the amount of keys in the replay cache. While this requires clients and servers to have loosely synchronised clocks, the server has to cache redeemed keys for a period of only one hour rather than seven days. We could not use E in the classical ticket scheme illustrated in Figure 6.4 because it would

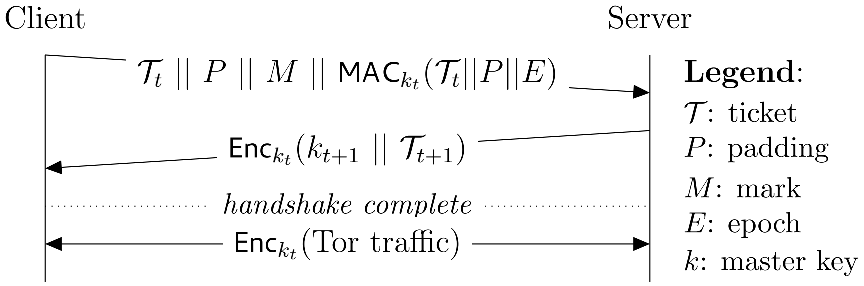

Client
Server
$\mathcal{T}_t \mid\mid P \mid\mid M \mid\mid \mathsf{MAC}_{k_t}(\mathcal{T}_t||P||E)$
$\mathsf{Enc}_{k_t}(k_{t+1} \mid\mid \mathcal{T}_{t+1})$
handshake complete
Enc_{k_t}(Tor traffic)
Legend:
$\mathcal{T}$: ticket
P: padding
M: mark
E: epoch
k: master key

Figure 6.6: The Tor-specific session ticket handshake. In contrast to Figure 6.4, it 1) has no ticket confirmation message and it 2) employs a timestamp E.

enable an attack. A censor could repeatedly terminate connections after a client tried to redeem its session ticket. After a while, the variable E will change since it holds a timestamp. This means that the MAC over the ticket handshake would change whereas the ticket *would not change*. We consider this a strong distinguisher.

Authentication using Uniform Diffie-Hellman

Our second authentication mechanism is an extension of the uniform Diffie-Hellman (UDH) handshake which was proposed in the obfs3 protocol specification [92, §3]. obfs3's handshake makes use of uniformly distributed public keys which are only negligibly different from random bytes. In general, UDH public keys are not distributed over the entire space of 4,096 bits. Public keys are always smaller than the modulus defined in RFC 3526 [104]. The unused bit space is, however, small enough for this distinguisher to be negligible. The probability P of a censor observing a UDH public key smaller than the 4,096-bit modulus n (refer to Kivinen and Kojo [104] for n's value) equals

$$P = \frac{n}{2^{4096} - 1}.$$

A censor could now monitor a server's handshakes over an extended period of time in order to determine if the full 4,096-bit space is never used which would be a sign of UDH being used. Doing that, a censor needs

$$\frac{ln(0.5)}{ln(P)} \approx 2^{66}$$

observations to have a confidence greater than 50% that a server is conducting UDH handshakes. Such an amount of observations is impractical. As a result, UDH can be used to agree on a master key k_t without a censor knowing that Diffie-Hellman is used.

In contrast to obfs3, our version of UDH is based on the 4,096-bit modular exponential group number 16 defined in RFC 3526 [104]. When initiating a UDH handshake, the client first generates a 4,096-bit private key x. The least significant bit of x is then unset in order to make the number even. The public key X is defined as $X = g^x \pmod{p}$ where $g = 2$. The server computes its private key y and its public key Y the same way. To prevent a censor from learning that X is a quadratic residue mod p—a clear distinguisher—the client randomly chooses to send either X or $p - X$ to the server. The server can then derive the shared master key by calculating $k_t = X^y \pmod{p}$. Since the private keys x and y are even, the exponentiations $X^y \pmod{p}$ and $(p - X)^y \pmod{p}$ result in the same shared master key.

Extending Uniform Diffie-Hellman

In its original form, the UDH construction does not protect against active probing. A censor who suspects UDH can simply probe the supposable bridge and opportunistically initiate a UDH handshake. To prevent that attack, we now turn UDH's anonymous handshake into an authenticated handshake in order to be resistant against active attacks.

We do so quite similar to the session tickets discussed in Section 6.3.1. As depicted in Figure 6.7, we concatenate pseudo-random padding P, the mark M, and a MAC. The MAC authenticates the respective public key as well as the padding. The MAC is keyed by a shared secret k_B which is distributed together with the Tor bridge's IP:port tuple over email or HTTPS (see step ① in Figure 6.5). Similar to tickets, the client's mark $M_C = \mathsf{MAC}_{k_B}(X)$ is used to easily locate the MAC. Note that k_B *can be reused* because it is only used to key the MAC. The handshake is conducted using UDH with

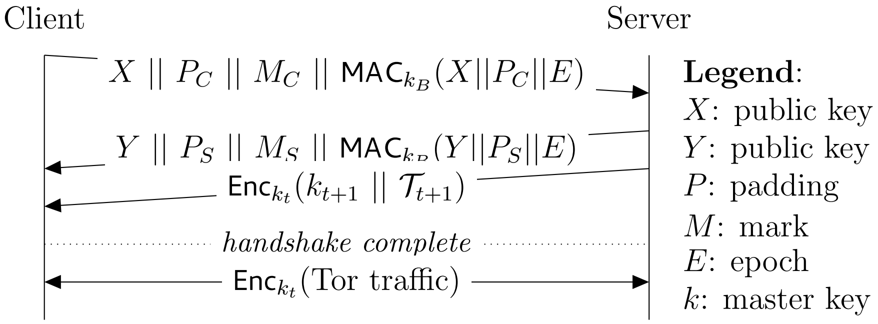

Client
Server
$X \mid\mid P_C \mid\mid M_C \mid\mid \mathsf{MAC}_{k_B}(X||P_C||E)$
$Y \mid\mid P_S \mid\mid M_S \mid\mid \mathsf{MAC}_{k_B}(Y||P_S||E)$
$\mathsf{Enc}_{k_t}(k_{t+1} \mid\mid \mathcal{T}_{t+1})$
handshake complete
Enc_{k_t}(Tor traffic)
Legend:
X: public key
Y: public key
P: padding
M: mark
E: epoch
k: master key

Figure 6.7: After client and server agreed on the master key k_t using Uniform Diffie-Hellman, the server is issuing a new session ticket for the client. Afterwards, both parties exchange Tor traffic.

randomly chosen public keys. As a result, two subsequent UDH handshakes based on the same k_B will appear different to a censor. We defend against replay attacks by adding E, the Unix epoch divided by 3600, to the MAC and cache the MAC for a period of one hour.

A successful UDH key agreement is followed by the server issuing a session ticket for the client. The client will then redeem this ticket upon connecting to the server the next time. Accordingly, we expect the UDH handshake to be done *only once*, namely when a Tor client connects to a bridge for the first time. From then on, session tickets as presented in Section 6.3.1 will be used for authentication.

To a censor, the payload of both authentication schemes is computationally indistinguishable from randomness. Furthermore, both schemes employ a two-way handshake and padding is used for the two handshakes to exhibit the same average length. As a result, a censor who is assuming that a server is running ScrambleSuit is unable to tell whether a client successfully authenticated herself by using UDH or by redeeming a session ticket.

We finally stress that bootstrapping ScrambleSuit using UDH provides *less security* than when bootstrapped using session tickets. Since the secret key k_B for UDH is shared by multiple clients, a malicious client in the possession of k_B who is able to eavesdrop on the connection of another client using the same ScrambleSuit server can conduct active MitM attacks. While Tor does protect against active MitM attacks,[6] this can be problematic for application protocols other than Tor. Therefore, we emphasise that session tickets are the preferred authentication mechanism for insecure protocols whereas our UDH extension's sole purpose is to make ScrambleSuit work well in Tor's infrastructure.

In order for a user to successfully connect to a ScrambleSuit server, she needs a *triple*: an IP address, a TCP port, and a secret which is either the UDH secret k_B or a session ticket tuple ($k_t \mid\mid \mathcal{T}_t$). We expect these triples to be distributed mostly electronically; over email, instant messaging programs, or online social networks. As a result, a user can simply copy and paste the entire triple into her configuration file.

We do, however, also expect verbal distribution of ScrambleSuit triples, e.g., over a telephone line. To facilitate this, we define the encoding format of secrets and tickets to be Base32 which consists of the letters A–Z, the numbers 2–7 as well as the padding character "=". The numbers 0 and 1 are omitted to prevent confusion with the letters I and O. Since there is no distinction between upper case and lower case letters, we hope to make verbal distribution less confusing and error-prone. After all, a ScrambleSuit bridge descriptor would look like:

```
Bridge scramblesuit 193.10.227.195:9002
       password=5TYVADJINHBB67PJSBPSWVR5IO742PVO
```

We believe that the prefix `password=` will find more acceptance amongst users than simply appending the secret.

6.3.2 Header format and confidentiality

Our protocol employs a custom message format whose header is illustrated in Figure 6.8. In a nutshell, ScrambleSuit exchanges variable-sized messages with optional padding. The padding is always discarded by the remote machine.

The first 16 bytes of the header are reserved for an HMAC-SHA256-128 which protects the integrity and authenticity of the protocol message. In

[6] A Tor client contains hard-coded keys of the directory authorities which then sign the network consensus.

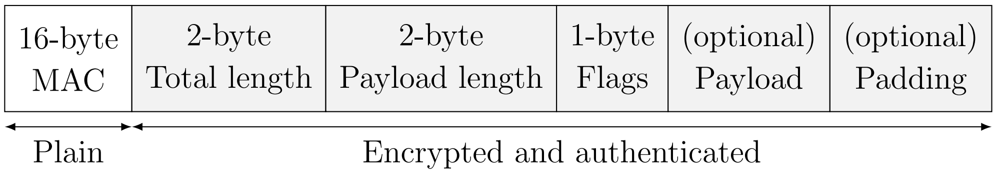


Figure 6.8: ScrambleSuit's message header format. The entire message is computationally indistinguishable from randomness.

Table 6.1: ScrambleSuit's protocol message flags.

Flag name	**Bit #**	**Meaning**
`FLAG_PAYLOAD`	1	Application data.
`FLAG_NEW_TICKET`	2	Newly issued session ticket and master key.
`FLAG_ACK_TICKET`	3	Acknowledgement of receipt of the session ticket.
`FLAG_PRNG_SEED`	4	PRNG seed to reproduce probability distributions.
`FLAG_REKEY`	5	Initiate rekeying before AES's counter wraps.

accordance with the encrypt-then-MAC principle, the keyed-hash message authentication code (HMAC) is computed over the encrypted remainder of the message. The secret key required by the HMAC is derived from the shared master key k_t.

The HMAC is followed by two bytes which specify the total length of the protocol message. ScrambleSuit's maximum transmission unit is 1448-byte-sized messages. Together with an IP and TCP header (which includes the timestamping option), this adds up to 1500-byte packets which fill an Ethernet frame. In order to be able to distinguish padding from payload, the next two bytes determine the payload length. If no padding is used, the payload length equals the total length.

To separate application data from protocol signaling, we define a 1-byte message flag field. The semantics of all five flags is explained in Table 6.1. The first bit signals application data in the message body whereas a message with the second bit set contains a newly issued session ticket. The third bit (which can be set together with the first bit) confirms the receipt of a session ticket. Bit number four allows the server to send a pseudo-random number generator (PRNG) seed to the client which is used for our traffic analysis defence (see Section 6.3.3). Bit five initiates rekeying which happens before the counter used for advanced encryption standard (AES) overflows. To prevent entropy exhaustion attacks, rekeying can only be triggered by the server. We reserve the remaining three bits for future use.

The header is then followed by the message payload which contains the application protocol transported by ScrambleSuit. We employ encryption in order to hide the application protocol, the padding as well as ScrambleSuit's header. With regard to Tor, this means that the already encrypted Tor traffic is wrapped inside yet another layer of encryption. For encryption, we use 256-bit AES in counter mode. The counter mode effectively turns AES into a stream cipher. We use two symmetric keys: one for the traffic $C \rightarrow S$ and one for $S \rightarrow C$. Both symmetric keys as well as the respective nonces for the counter mode are derived from the shared 256-bit master key using HMAC-based key derivation function (HKDF) based on SHA256 [105].

6.3.3 Polymorphic shape

So far, we discussed defences against censors aiming to analyse packet payload or conduct active attacks to reveal ScrambleSuit's presence. However, a censor could make use of *traffic analysis*, i.e., analyse communication aspects other than the payload. In this section, we propose lightweight countermeasures to diminish—but not to defeat!—such attacks. In particular, we will teach every ScrambleSuit server to generate its own and unique "protocol shape".[7]

Our definition of ScrambleSuit's shape is twofold: we consider *packet lengths* and *inter-arrival times*. While encrypting our protocol messages

[7] This happens similar to the *scramble suits* in Philip K. Dick's novel "A Scanner Darkly".

renders payload analysis useless, these two flow metrics still leak information about the transported application [106–108]. As a result, we seek to disguise these characteristics in order to decrease the accuracy of protocol classifiers trained to identify our protocol. Our approach to this problem is *protocol polymorphism*.

We achieve polymorphism by creating *one protocol shape for every server*. When a ScrambleSuit server bootstraps for the first time, it randomly generates a 256-bit seed. This seed is then fed into a PRNG which is used to obtain two discrete probability distributions. These two distributions dictate the desired shape of packet lengths and inter-arrival times. Furthermore, a server communicates its unique PRNG seed to clients (see Table 6.1) after successful authentication. Since the seed is shared by both parties, they can generate identical probability distributions and thus shape their traffic the same way. A censor monitoring two distinct ScrambleSuit servers will observe different distributions for packet lengths and inter-arrival times.

Once our PRNG is seeded, we generate the two distributions by first determining the amount of bins n which is uniformly chosen from the set $\{1..100\}$. In the next step, we assign each bin b_i for $1 \leq i \leq n$ a probability by randomly picking a value in the interval $]0, 1 - \sum_{j=0}^{i-1} b_j[$ with $b_0 = 0$. The following gives an example for four bins.

$$b_0 \leftarrow 0 \tag{6.1}$$

$$b_1 \overset{R}{\leftarrow}]0, 1 - b_0[\tag{6.2}$$

$$b_2 \overset{R}{\leftarrow}]0, 1 - b_0 - b_1[\tag{6.3}$$

$$b_n \overset{R}{\leftarrow}]0, 1 - b_0 - ... - b_{n-1}[\tag{6.4}$$

Packet length adaptation

It is well known that a network flow's packet length distribution leaks information about the network protocol [106, 109] and even the content [108, 110]. For instance, a large fraction of Tor's traffic is composed of 568-byte packets which is the result of Tor's internal use of 512-byte cells plus TLS' header (see Figure 6.11). These 568-byte packets form a strong distinguisher which can be used to spot a Tor flow by simply capturing a few dozen network packets as shown by Weinberg et al. [87]. To defend against such simple applications of traffic analysis, we modify the packet length distribution of our transported application.

Typically, non-interactive TCP applications transmit segments filling the network link's maximum transmission unit (MTU) as long as they have enough to "say". Applications will only send packets smaller than the MTU if there is not enough data in the send buffer. Due to their sheer volume,[8] we deem MTU-sized packets to be of no interest to censors. What we aim to disguise is only packets *smaller than the MTU*. This is done by randomly sampling a packet length from the probability distribution over all our packet lengths. The original packet length is then padded to fit the sampled packet length. The padding can be anything in between 0 and 1520 bytes. The reason for 1520 instead of 1499 bytes is that the smallest unit we can transmit is an empty ScrambleSuit message which counts 21 bytes in size.

Inter-arrival time adaptation

Analogous to packet lengths, the distribution of inter-arrival times between consecutive packets has discriminative power and can be used by censors to identify protocols [112]. While inter-arrival times are frequently distorted by jitter, overloaded middle boxes, and the communicating end points, it would be no sound strategy to assume the network to be unreliable enough to render measurements difficult.

We employ an obfuscation mechanism analogous to the packet length adaptation discussed earlier. First, the shared PRNG seed is used to generate a pseudo-random probability distribution in the range of $[0, 10[$ milliseconds, the reasoning for which is discussed in the next paragraph. Random samples are then drawn from this distribution which are used to artificially delay network packets. Similar to the implementation of SkypeMorph [88], we make use of a dedicated send buffer which is processed independently of the locally incoming data. This decoupling of the incoming and outgoing data stream makes it possible to "reshape" inter-arrival times.

[8] According to a study conducted by CAIDA [111], MTU-sized packets form a significant

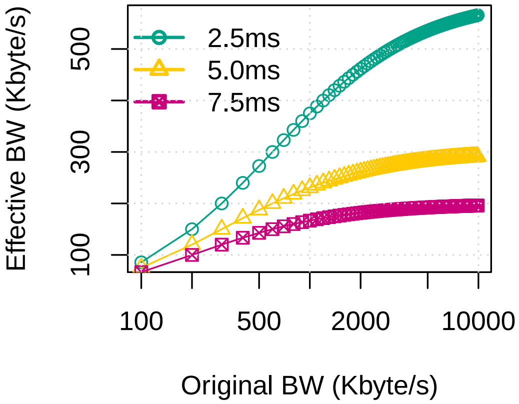


Figure 6.9: The bandwidth cost when employing three different obfuscation delays. ScrambleSuit uses an average obfuscation delay of 5 ms.

The artificial increase of inter-arrival times has a negative impact on throughput. If chosen too high, it can quickly become a nuisance to users. Figure 6.9 illustrates the mapping from original bandwidth (without obfuscation) to effective bandwidth (after obfuscation) under three different average obfuscation delays. The higher the average delay becomes, the smaller is the effective bandwidth. The plotted data is based on the following equation which yields the overhead α,

$$\alpha = \left(\frac{BPS}{MTU} \cdot E[d] \right) \cdot \frac{1}{1000} + 1.$$

The variable MTU refers to the maximum transmission unit which is typically 1500. BPS stands for "bytes per second" and refers to the original bandwidth of the available network link. $E[d]$ represents the expected value of the respective probability distribution d, i.e., the average obfuscation delay. In Figure 6.9, we plot the expected values 2.5, 5, and 7.5 milliseconds, respectively. ScrambleSuit uses the interval of $[0, 10[$ milliseconds for artificial delays. This interval has an expected value of 5 ms which we believe to be a reasonable balance between this obfuscation/performance trade-off. It is ultimately limited to an effective throughput of 300 KB/s and, when Tor

fraction of all observed packet lengths.

is transported, can achieve throughputs around 150 KB/s as we will show in our experimental evaluation.

Shortcomings

It is important to note that for a censor armed with a well-chosen set of features and sufficient computational resources, traffic analysis can be a powerful attack. Robust defences, on the other hand, are believed to be expensive [107]. For instance, Dyer et al.'s simple yet powerful VNG++ classifier [107] only makes use of coarse features such as connection duration, total bytes transferred and the "burstiness" of a flow. Significantly decreasing VNG++'s accuracy would cause a drastically increased protocol overhead.

Nevertheless, traffic analysis does not give censors a *certain answer*. False positives are always a problem and can lead to over-blocking. As mentioned in our threat model, we believe that the censor might use traffic analysis to select a subset of traffic for closer inspection but not to block flows.

6.3.4 Cryptographic assumptions

Our authentication mechanisms rely on 1) AES-CBC, 2) pseudo-random initialisation vectors, 3) HMAC, 4) pseudo-random padding, and 5) uniformly distributed Diffie-Hellman public keys. Exchanged application data is then encrypted using AES-CTR and authenticated by an HMAC. We expect all data exchanged between ScrambleSuit servers and clients to be computationally indistinguishable from random data of the same length.[9] As a result, we have the following assumptions regarding our cryptographic building blocks. We assume AES to be a pseudo-random permutation and our HMAC to be a pseudo-random function. The padding is assumed to come from a cryptographically secure PRNG and the public keys are assumed to be uniformly distributed (see obfs3 [92]).

[9] Our goal is to achieve a security level equivalent to at least 128 bits of symmetric security as defined in [113].

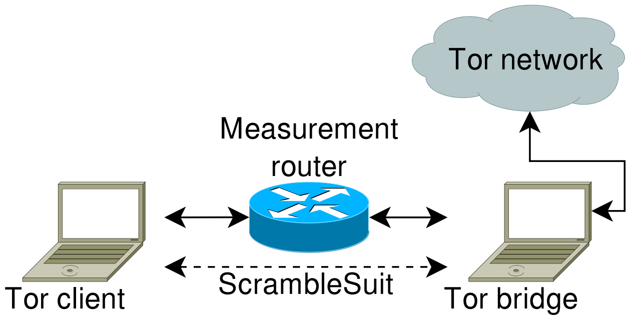


Figure 6.10: Our experimental setup used to measure ScrambleSuit's obfuscation and performance.

6.4 Experimental evaluation

We implemented a fully functional prototype of ScrambleSuit in the form of several Python modules for obfsproxy.[10] Our prototype consists of approximately 2,200 lines of code. We used the library PyCrypto [114] for cryptographic primitives. The measurements discussed below were all conducted using this prototype.

As illustrated in Figure 6.10, our experimental setup consisted of two Debian GNU/Linux machines which were connected over a router performing the measurements. All three machines were connected over 100 Mbit/s Ethernet. We expect this setup to be ideal for a censor because it minimises network interference such as jitter or packet fragmentation. As a result, we believe that a censor would do worse in practice. Both of our machines were running Tor v0.2.4.15-rc and obfsproxy. The Tor bridge was configured to remain private and only used by our client. The bridge then relayed all traffic into the public Tor network. Note that ScrambleSuit was only "spoken" in between the client and the bridge.

[10] The code is available under a free license at http://veri.nymity.ch/scramblesuit/.

6.4.1 Blocking resistance

It is difficult to evaluate the effectiveness of our obfuscation techniques since ScrambleSuit does not have a cover protocol to mimic. Otherwise, our evaluation would simply investigate the similarity between our protocol and its cover protocol. Instead of measuring ScrambleSuit's similarity to a mimicked protocol, we measure the *deviation* from its *transported application*, i.e., Tor. Intuitively, higher deviation means better obfuscation.

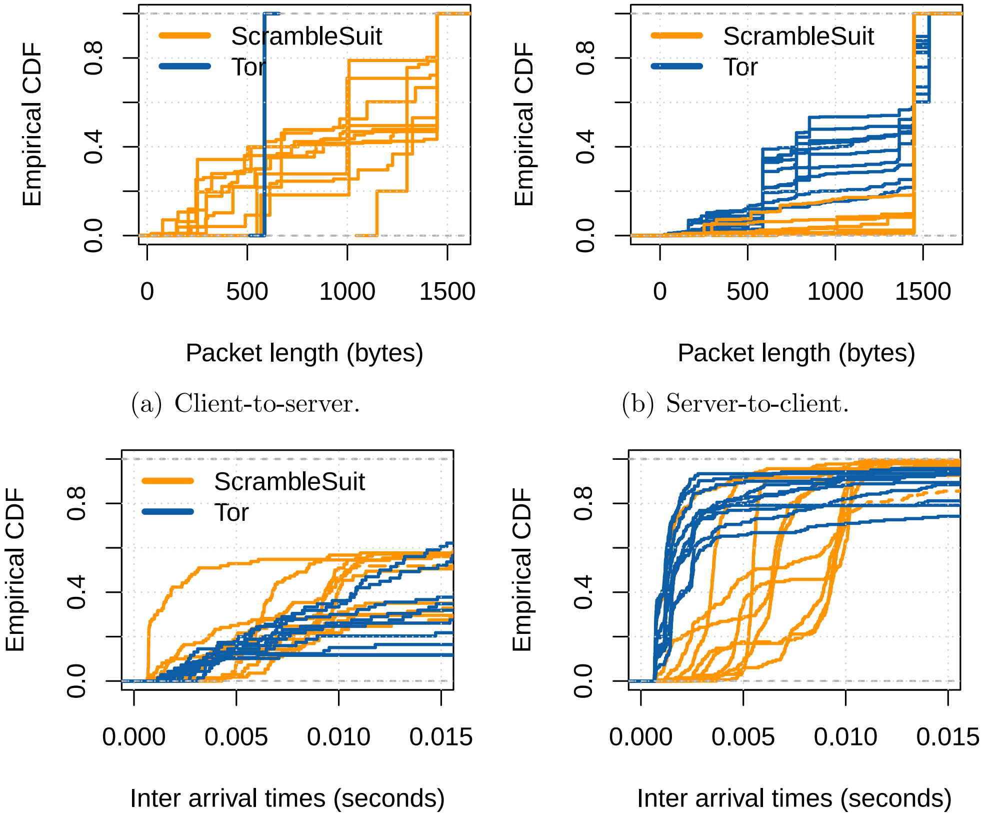
We obtained traces of packet lengths and inter-arrival times for ScrambleSuit and Tor. In the following, we qualitatively compare both traces. To create network traffic for these traces, we repeatedly downloaded the 1 MB Linux kernel v1.0 from kernel.org on the client.[11] We downloaded the file ten times over Tor and ScrambleSuit, respectively. The measurements only covered the download and not ScrambleSuit's authentication or Tor's bootstrapping.

The two packet length distributions are illustrated in Figure 6.11(a) and 6.11(b). The dark blue lines represent Tor flows whereas bright orange lines depict ScrambleSuit. The packet length distributions clearly show the prevalence of Tor's 586-byte packets; even more so in client-to-server traffic where the purpose of these packets is flow control. While server-to-client traffic carries less 586-byte packets, they still form a significant fraction of overall packet lengths. ScrambleSuit effectively eliminates this traffic signature and transmits more MTU-sized packets. Recall that every ScrambleSuit download represents only *one specific shape*. Different servers come with different shapes.

Figure 6.11(c) and 6.11(d) illustrate the inter-arrival times derived from the same data. Again, Tor is shown as dark blue and ScrambleSuit as bright orange lines. The inter-arrival times in Figure 6.11(c) tend to be rather high—only roughly 60% of Tor packets had an inter-arrival delay below 15 ms—because the bulk data was travelling from the server to the client. For this reason, the delays are significantly smaller in Figure 6.11(d).

Once again, ScrambleSuit visibly deviates from Tor's distribution. However, as we will show in Section 6.4.2, artificially increased inter-arrival times

[11] URL: https://www.kernel.org/pub/linux/kernel/v1.0/linux-1.0.tar.bz2



(a) Client-to-server.

(b) Server-to-client.

(c) Client-to-server.

(d) Server-to-client.

Figure 6.11: Tor's and ScrambleSuit's packet length distribution and inter-arrival times for both, client-to-server and server-to-client traffic.

have a negative effect on throughput.

6.4.2 Performance

We are interested in both *computational* as well as *network* overhead. The following sections discuss the amount of overhead ScrambleSuit carries compared to a bare Tor connection.

Cryptographic overhead

Our two authentication mechanisms come with small computational overhead. For both parties, UDH requires two modular exponentiations and two MAC generations.[12] Session tickets—which constitute the majority of authentications—are even cheaper: the client again calculates two MACs whereas the server symmetrically decrypts the ticket and verifies two MACs.

After authentication, protocol messages are symmetrically encrypted and protected by a MAC. We expect the low computational overhead to make ScrambleSuit suitable for resource-constrained devices such as smartphones.

Network overhead

Our protocol's network overhead is increased by the artificial inter-arrival times, packet padding, and the protocol header. In order to gain a good understanding of the exact network overhead, we created a 1,000,000-byte file containing random bytes and placed it on a web server operated by Karlstad University. We then downloaded this file using the tool wget; 25 times over HTTP, Tor, ScrambleSuit, and ScrambleSuit without inter-arrival time obfuscation, respectively. For Tor and ScrambleSuit, we established a new circuit for every download but we used the same entry guard to eliminate unnecessary variance. All data was captured after a Tor circuit has been established, so handshakes are not part of the data. We then calculated the mean $\bar{x}$ and the standard deviation s for several performance metrics. The results are depicted in Table 6.2.

[12] It is necessary to calculate the authenticating MAC as well as the mark used to locate the MAC.

Table 6.2: Mean ($\bar{x}$) and standard deviation (s) of the goodput, transferred KBytes, and the total overhead. The data was generated based on the download of a 1,000,000-byte file.

	HTTP		**Tor**	
	$\bar{x}$	s	$\bar{x}$	s
Goodput	6.3 MB/s	3.4 MB/s	286 KB/s	227 KB/s
C→S KBytes	23.1	1.6	66.4	6.5
S→C KBytes	1047	20.7	1130	12.8
Total overhead	7%	2.2%	19.6%	1.9%

	ScrambleSuit		**ScrambleSuit-nodelay**	
	$\bar{x}$	s	$\bar{x}$	s
Goodput	148 KB/s	61 KB/s	321 KB/s	231 KB/s
C→S KBytes	122.9	24.1	111.9	17.9
S→C KBytes	1397.8	125.7	1342.7	112.2
Total overhead	52.1%	15%	45.5%	13%

The goodput refers to the application layer throughput. We achieved very high values for the HTTP download because the file transfer could be carried out over the local area network (LAN). Tor averaged at roughly 280 KB/s and ScrambleSuit achieved slightly more than half of that. Just like Tor, ScrambleSuit exhibits high standard deviation. The main reason for this is differences in Tor circuit throughput. ScrambleSuit without inter-arrival time obfuscation is comparable to Tor. In fact, it exceeds Tor's throughput but this can again be explained by varying circuit throughput. This shows that the obfuscation of inter-arrival times has the biggest impact on ScrambleSuit's throughput (cf. Figure 6.9).

The next two rows in Table 6.2 refer to the transferred KBytes from client to server (C→S) and server to client (S→C). Note that these metrics cover all the data which was present on the wire; including IP and TCP header. The consideration of IP and TCP overhead is important because ScrambleSuit's packet padding mechanism introduces additional TCP/IP packets. Unsurprisingly, Tor transferred more data than HTTP because of Tor's and TLS' protocol overhead. ScrambleSuit transferred the most data because of the additional protocol header as well as the varying packet lengths. Given ScrambleSuit's 1448-byte MTU, the 21-byte protocol header only accounts for 1.5% overhead.

The last row in Table 6.2 illustrates the total protocol overhead which is simply a function of the previous two rows. HTTP has the lowest overhead followed by Tor and finally ScrambleSuit. Our protocol exhibits 45–50% overhead which is about twice as much as Tor.

6.5 Discussion

Active probing: A censor could still actively probe a ScrambleSuit server. Upon establishing a TCP connection, a censor could proceed by sending arbitrary data. However, without knowing the UDH shared secret or possessing a valid session ticket, authentication cannot succeed and the server will remain silent.

In contrast to SilentKnock and BridgeSPA, ScrambleSuit does not disguise its “aliveness”. While this approach does leak information,[13] it has the benefit of making ScrambleSuit significantly easier to deploy due to lack of platform dependencies such as the kernel interface libnetfilter_queue to parse raw network packets in userspace.

Injection, Modification, Dropping: A censor could tamper with an established ScrambleSuit connection by injecting, modifying, or dropping packets. After authentication, all exchanged data is authenticated which allows the communicating parties to detect such tampering. If the authenticating HMAC of either party is invalid, the connection is terminated immediately.

[13] A censor learns that a server is online but unwilling to talk unless given the “correct” data.

Hijacking a ScrambleSuit connection is reduced to the same problem; a censor would bypass authentication but is unable to talk to the other party because the session keys are unknown. Finally, dropped packets would be handled by TCP's retransmission mechanism whereas terminated connections could manually be restarted by users.

Payload analysis: Payload analysis would only yield data which is computationally indistinguishable from randomness. While most encrypted protocols negotiate session parameters in cleartext, VPNs with pre-shared keys and BitTorrent's message stream encryption also bootstrap using high-entropy network packets. Aside from that, it is difficult to survey how many "fully-random" protocols already exist in the wild. One approach would be to monitor large-scale network links and determine which fraction of TCP streams transports only high-entropy data. Similar work was done by White et al. [115] but the authors focused on per-packet rather than on per-connection measurements.

Ideally, more applications considered legitimate by censors would start transporting only high-entropy payload, thus inflating the set of protocols ScrambleSuit can hide amongst. This could be achieved by a major browser employing such encryption on the transport layer or even by an extension for the TLS protocol which would provide optional support for such a mode.

Flow analysis: Flow analysis would yield a unique distribution of packet lengths and inter-arrival times which is different for all ScrambleSuit servers. While strong traffic analysis defence is expensive, these attacks will always have a range of *uncertainty* causing false positives. Our goal was to further increase this uncertainty. We placed more value on defeating active probing attacks because in contrast to traffic analysis, they enable *deterministic* protocol identification.

6.6 Conclusion

This chapter presented ScrambleSuit; a lightweight transport protocol which provides obfuscation for applications such as Tor. The two major contributions of our protocol are the ability to defend against *active probing* and simple *protocol classifiers*. We achieve the former by proposing two authentication mechanisms—one general-purpose and the other specifically for Tor—and the latter by proposing morphing techniques to disguise packet lengths and inter-arrival times.

We further developed a prototype of ScrambleSuit and used it to conduct an experimental evaluation. We discussed the effectiveness of our obfuscation techniques as well as ScrambleSuit's overhead. Our evaluation suggests that our protocol can provide decent protection against censors who do not overblock significantly. As a result, we believe that our protocol can provide a practical alternative in countries which do not whitelist Internet traffic.

Chapter 7

Related work

Previous work in Internet censorship can be classified into several sub-fields. In particular, we discuss censorship analysis in Section 7.1 and censorship circumvention in Section 7.2.

7.1 Censorship analysis

A comprehensive and detailed understanding of real-world censorship systems is crucial as it is the fundament for censorship circumvention.

7.1.1 Networked services

While a large body of work focuses on the network, transport, and application layer, Internet censorship also manifests on networked applications such as search engines and online social networks. A highly popular service which is frequently subject to censorship is Twitter. In 2012, Thomas et al. analysed how thousands of fraudulent Twitter accounts were abused with the goal of influencing public opinion after elections in Russia [116]. One year later, Verkamp and Gupta investigated four additional cases, two in China and one in Mexico and Syria [117]. With Twitter typically being blocked in China, Weibo is a successful alternative despite censorship on the service level. In 2013, Chen et al. analysed the evolution of censorship on Weibo and how users adapt their communication behaviour when their messages are deleted [118]. Similarly, Zhu et al. analysed message deletion on Weibo [36]. Amongst others, the authors sought to find out what messages are subject to deletion and when they are deleted.

7.1.2 The Great Firewall of China

The Great Firewall of China has received considerable attention from the research community. The first academic study to look at the Great Firewall was published in 2006 by Clayton and Murdoch [16]. The authors established that some keywords are blocked and cause TCP RST segments to be injected. The authors also showed that the blocking can be circumvented by simply ignoring these RST segments. Weaver, Sommer, and Paxson later showed that injected RST segments can be detected [68]. Duan et al. then showed that injected segments can not only be detected, but also protected against [119]. While Clayton and Murdoch focused on HTTP, Lowe, Winters, and Marcus analysed how the GFW handles blocked DNS traffic [33]. The authors found that the GFW blocks DNS on the router level and injects spoofed DNS replies for selected domains. A more detailed study was later conducted by Wright [35]. Wright also analysed regional variations in how the filtering works. So far, the GFW's DNS censorship was believed to be limited to China. However, in 2012, anonymous authors showed that the GFW also censors transit traffic which does not originate in China and does not target IP addresses in China [34]. Most recently, in 2014, anonymous authors conducted a more comprehensive study of the GFW's DNS censorship and inferred the GFW's topological structure [50]. In 2011, Xu, Mao, and Halderman attempted to determine where on the network the GFW is located [20]. Their results suggested that most filtering happens in border autonomous systems but some filtering is also happening in provincial networks. Approximately 15 years after Ptacek's and Newsham's seminal 1998 paper [26], Khattak et al. analysed the GFW's analysis model and found numerous evasion opportunities on the TCP and IP layer [120].

On the one hand, this thesis revisits the GFW's topology by studying it in greater detail than previous work. On the other hand, we analyse novel aspects such as the GFW's spatiotemporal characteristics and how it implements active probing attacks.

7.1.3 Other countries

Traditionally, the largest body of work in censorship analysis has focused on the Great Firewall of China and this thesis is no exception. With more and more countries deploying censorship equipment in their network backbone, this seems to change. In 2012, Anderson documented how Iran began to install private IP networks which are unreachable to hosts outside Iran [121]. One year later, Anderson analysed historical network data and showed how Iran is throttling network protocols, presumably for the sake of censorship [122]. In 2013, Aryan et al. gave a basic overview of how filtering works in Iran including DNS censorship and traffic throttling [123]. In the same year, Nabi provided a first analysis of Internet censorship in Pakistan [124]. Verkamp and Gupta conducted a horizontal study and uncovered censorship in 11 countries which included some poorly explored areas such as Malaysia, Turkey, and South Korea [125]. In 2014, Di Florio et al. analysed Turkey's blocking of Twitter and implemented an application facilitating DNS blocking [126].

7.1.4 Censorship measurement platforms

One-off measurements are of limited value because censorship frequently changes over time; and sometimes does so rapidly. Various censorship measurement platforms have been proposed to conduct longitudinal measurements. In 2007, Crandall et al. proposed ConceptDoppler which uses natural language processing to discover keywords filtered by the GFW [27]. A similar concept was proposed by Sfakianakis, Athanasopoulos, and Ioannidis who suggest to use a platform based on PlanetLab [55] to analyse global web filtering [127]. Most recently, Filastò and Appelbaum proposed OONI [64]. OONI is a flexible censorship measurement framework which is still under development and has already been used in many countries [128].

In this thesis, we proposed a novel concept for a censorship measurement platform which is built on top of RIPE Atlas—an already existing network measurement platform. While our concept has access to thousands of probes, we are limited in the kind of measurements we can conduct. Also, ethical standards have to be considered given that all probes are contributed by volunteers.

Related to censorship measurement platforms, we also proposed the design of a lightweight tool which is run by censored volunteers. This concept is different from previously proposed platforms because our tool is meant to be run by censored users rather than researchers. Again, ethical standards are very important because users are directly involved in the process.

7.1.5 Side channel measurements

Most censorship measurement is done by controlling the censored client, its destination, or the network in between. However, Ensafi et al. showed that side channels in network stacks can be used to, e.g., detect packet dropping between two machines which are not under the researcher's control [14, 43].

Building on the work of Ensafi et al., we employed two different side channel measurement techniques to analyse the GFW spatiotemporally by measuring the reachability between many clients in China and many Tor relays outside of China.

7.2 Censorship circumvention

Numerous censorship-resistant communication systems have been proposed. In the following, we divide these systems into low-latency systems, high-latency systems, proxy distribution, and decoy routing.

7.2.1 Low-latency systems

In 2012, Wang et al. proposed CensorSpoofer [129]. Using IP spoofing, their system decouples the censored user's upstream and downstream channel—in the hope that this makes it more difficult for censors to correlate and then block the user's activity. Moghaddam et al. discussed SkypeMorph which transforms Tor traffic to Skype traffic [88]. A similar concept was later presented by Li, Schliep, and Hopper in 2014 [130]. As long as a censor allows

Skype traffic, censored users should be able to connect to the Tor network. Weinberg et al. presented StegoTorus which can be considered an extended version of the concept of obfsproxy [24, 87]. Depending on steganography modules, StegoTorus obfuscates network traffic and makes it possible to multiplex TCP streams over multiple TCP connections. In contrast to evading deep packet inspection systems, Fifield et al. sought to solve the problem of IP address blocking [131]. Their system employs unaware website visitors as short-lived proxies for censored users.

In 2013, Dyer et al. proposed FTE which transforms a stream of bytes to a set of provided regular expressions [66]. An extension of the concept, LibFTE, was presented one year later [132]. Assuming that a set of unblocked regular expressions is known, censored users are able to transform their network traffic to that very set. In the same year, Fifield, Nakibly, and Boneh discussed the use of online scanning services for censorship circumvention. Using redirection techniques, these services can be used to tunnel network traffic [133]. Zhou et al. proposed SWEET which tunnels blocked network traffic over encrypted email services [90]. While suffering from high latency, most users are able to connect to at least one encrypted email service which is not controlled by their government. Houmansadr et al. designed FreeWave which disguises blocked traffic as Voice-over-IP communication which is expected to be unblocked [89].

In 2014, Nobori and Shinjo proposed VPN Gate [134]. Their system is a volunteer-run set of VPN systems for censored users. VPN Gate is provides stronger blocking-resistance than ordinary VPN services and the authors also discussed several challenges they encountered while scaling their system. Brubaker, Houmansadr, and Shmatikov proposed to tunnel blocked traffic over third-party cloud services [135]. As long as the censor and cloud provider do not collude, censored users can hide their blocked network traffic in cloud traffic. Jones et al. strove to improve the typically limited capacity of HTTP's upstream channel and did so by disguising blocked network traffic as web searches [136].

Designing robust and undetectable circumvention systems is challenging as demonstrated by a number of attacks on these systems. In 2013, Houmansadr, Brubaker, and Shmatikov showed that it is very difficult to completely mimic a target protocol [137]. Small deviations from the target protocol are enough for a censor to distinguish the mimiced from the "real" application. To solve this problem for HTTP, Massar et al. implemented a system which tunnels obfuscated Tor traffic over an actual web browser and web server [138]. Similar to Houmansadr et al., Geddes, Schuchard, and Hopper showed that there are a number of attacks, a censor can conduct in order to break a protocol containing tunneled data while the very same protocol without tunneled data remains mostly intact [139].

ScrambleSuit, which was proposed in this thesis, differs from the circumvention protocols discussed above in that it seeks to be polymorphic rather than exhibit a static flow signature. That way, it should be more difficult for a censor to block all ScrambleSuit instances as they appear to be different from each other. ScrambleSuit merely demonstrates the feasibility of polymorphism and there are many ways to extend the idea in future research.

7.2.2 High-latency systems

In contrast to its low-latency cousin, high-latency systems provide a higher degree of obfuscation at the cost of additional latency and less throughput. Latency can range from many seconds to hours, or even days.

In 2000, Waldman, Rubin, and Cranor presented Publius which is a censorship-resistant anonymous publishing system [140]. The following year, Waldman and Mazières proposed Tangler, a censorship-resistant publishing system which works by "entangling" documents [141]. Stufflefield's and Wallach's Dagster protocol is similar and can work with a single server setting because it intertwines legitimate and illegitimate data [142]. In 2002, Feamster et al. presented Infranet [143]. Cooperating web servers enable to client retrieval of sensitive information by implementing a special tunneling protocol. In 2010, Burnett, Feamster, and Vempala make use of sites which accept user-generated content such as photo-sharing sites to exchange hidden messages [144]. A similar idea was pursued by Invernizzi, Kruegel, and Vigna in 2013 [145]. Their system hides messages in blog posts. Most recently in 2014, Connolly et al. propose TRIST which hides messages in images [146].

7.2.3 Proxy distribution

A classical problem faced by many circumvention systems is the *proxy distribution problem* which can be phrased informally as "how are proxies given to users in need while at the same time avoiding that they can be learned by censors?".

In 2008, Sovran, Libonati, and Li presented Kaleidoscope which attempts to solve this problem by distributing proxies over social networks which reflect real-world trust relationships [147]. Two years later, Mahdian studies this problem from an algorithmic point of view [148]. A different approach was proposed by McCoy, Morales, and Levchenko in 2011 [149]. Their system, Proximax, attempts to maximise the usefulness of proxies while at the same time tries to minimise the risk of getting detected. A more sophisticated model was presented by Wang et al. in 2013 [38]. On the attack side, Ling et al. showed in 2012 how an attacker with moderate resources can detect a large fraction of all Tor bridges [10].

ScrambleSuit also relies on the distribution of ScrambleSuit bridges to censored users. To facilitate the distribution, we rely on the Tor Project's BridgeDB component [150]. BridgeDB distributes Tor bridges over HTTPS, email, and out-of-band channels.

7.2.4 Decoy routing

Traditionally, censorship circumvention systems rely on a censored client connecting to an uncensored proxy. In addition to this end-to-end approach, an alternative *end-to-middle* design emerged over the past years. These systems—commonly referred to as *decoy routing*—rely on cooperating routers in the Internet backbone to circumvent censorship systems.

In 2011, three independently proposed systems introduced the concept of decoy routing [151–153]. These systems differ in their design and implementation but rely on the same concept. One year later, Schuchard et al. discussed an attack and pointed out that the threat model of decoy routing systems should consider an adversary capable of modifying its routing tables [154]. This attack was later shown to be based on incomplete assumptions [155]. In 2014, Wustrow, Swanson, and Halderman improved their original design to make the system more attractive for ISPs to integrate in their networks [156].

Chapter 8

Conclusions

8.1 Summary

This thesis proposed techniques for Internet censorship measurement and circumvention, some of which were implemented and deployed in real-world settings. In particular, we proposed several novel measurement techniques which are based on involvement of censored users, the volunteer-run RIPE Atlas platform, and newly discovered network side channels which are capable of detecting packet loss between two networked devices. We subsequently employed these measurement techniques to analyse several real-world cases of Internet censorship. We determined how the GFW is blocking the Tor anonymity network, and we shed light on two incidents in Turkey and Russia.

With respect to censorship circumvention, we proposed ScrambleSuit, which is a polymorphic traffic obfuscation protocol. ScrambleSuit protects against the GFW's active probing attacks by relying on a shared secret, that is distributed out-of-band. The protocol also provides limited defence against traffic analysis. ScrambleSuit is deployed and part of TorBrowser. In addition to ScrambleSuit, we proposed a way to provoke packet fragmentation, which is able to circumvent DPI boxes that are not able to reassemble TCP streams.

While the results and tools presented in this thesis strengthen a free and open Internet, censorship remains an arms race and more work is needed. It remains to be seen when or if an equilibrium between censoring and circumventing forces will set in.

8.2 Future work

Traditionally, one class of circumvention protocols seeks to hide amongst the set of unblocked protocols such as voice over IP (VoIP) or HTTP [66, 87–89]. An alternative approach is to *adapt unblocked protocols* to provide a universal and pervasive encryption layer which facilitates the hiding of application protocol. While TLS could be named as an example of such a protocol, it is not designed to hide the application protocol. In fact, the clear-text TLS handshake can be inspected by adversaries and provides plenty of clues about the transported application. For example, the cipher list sent by the TLS client could reveal the user's web browser [157]. A more suitable pervasive encryption layer is provided by opportunistic encryption protocols such as tcpcrypt [158] or tcpinc [159] which is in the process of being standardised. These protocols are barely configurable which makes it difficult to determine which application they transport. If such a protocol were to be deployed on the Internet, it would not only protect unblocked traffic but also make it substantially easier for blocked traffic to "hide in the crowd". However, because opportunistic encryption does not provide authentication, it can be broken by an active MitM. Still, active attacks can be detected, which raises the bar for attackers.

Orthogonal to opportunistic encryption, there is a growing trend to outsource communication infrastructure, e.g., by relying on content delivery networks or cloud computing. This trend facilitates surveillance because the organisation providing the communication infrastructure is now in control of the network traffic but, ironically, it can help with censorship circumvention. In particular, it becomes easier for blocked network protocols to "blend in" with unblocked network traffic as both classes of traffic use the same infrastructure. This fact was already exploited for censorship circumvention but there are many opportunities for future work [135, 160].

Finally, the concept of polymorphism for censorship circumvention, as proposed by ScrambleSuit, should be explored further. We only focused on two flow features but did not consider how packet payload can be made polymorphic.

Appendix A

Published work

The following list contains all work which has been published over the course of my Ph.D. studies. The list contains peer-reviewed work, technical reports, and invited articles. For every work, my respective contribution is mentioned.

P.1 Roya Ensafi, Philipp Winter, Abdullah Mueen, Jedidiah R. Crandall. *Large-scale Spatiotemporal Characterization of Inconsistencies in the World's Largest Firewall.* Tech. rep. University of New Mexico, 2014. URL: http://arxiv.org/pdf/1410.0735
Contribution: I contributed the design, implementation, and evaluation (but not the idea) of the SYN backlog scan.

P.2 Collin Anderson, Philipp Winter, and Roya. "Global Censorship Detection over the RIPE Atlas Network". In: *Free and Open Communications on the Internet.* USENIX, 2014. URL: http://cartography.io/foci2014.pdf
Contribution: I contributed the majority of Section 1, 2, and 3.

P.3 Philipp Winter, Richard Köwer, Martin Mulazzani, Markus Huber, Sebastian Schrittwieser, Stefan Lindskog, Edgar Weippl. "Spoiled Onions: Exposing Malicious Tor Exit Relays". In: *Privacy Enhancing Technologies Symposium.* Springer, 2014. URL: http://www.cs.kau.se/philwint/spoiled_onions/pets2014.pdf
Contribution: Richard, Martin, Markus, Sebastian, and Edgar contributed the design, implementation, and evaluation of HoneyConnec-

tor. Stefan read drafts and provided feedback. The remainder was done by me.

P.4 Philipp Winter and Stefan Lindskog. *Spoiled Onions: Exposing Malicious Tor Exit Relays*. Tech. rep. Karlstad University, 2014. URL: http://veri.nymity.ch/spoiled_onions/techreport.pdf
Contribution: Stefan read drafts and provided feedback. The remainder was done by me.

P.5 Philipp Winter, Tobias Pulls, and Juergen Fuss. "ScrambleSuit: A Polymorphic Network Protocol to Circumvent Censorship". In: *Workshop on Privacy in the Electronic Society*. ACM, 2013. URL: http://www.cs.kau.se/philwint/pdf/wpes2013.pdf
Contribution: Tobias and Jürgen helped with the design of the cryptographic building blocks. The remainder was done by me.

P.6 Philipp Winter, Tobias Pulls, and Juergen Fuss. *ScrambleSuit: A Polymorph Network Protocol to Circumvent Censorship*. Tech. rep. http://www.cs.kau.se/philwint/pdf/scramblesuit2013.pdf. Karlstad University, 2013
Contribution: Tobias and Jürgen helped with the design of the cryptographic building blocks. The remainder was done by me.

P.7 Philipp Winter. "Towards a Censorship Analyser for Tor". In: *Free and Open Communications on the Internet*. USENIX, 2013. URL: https://www.usenix.org/system/files/tech-schedule/foci13-papers-archive.zip
Contribution: I am the sole author of this paper.

P.8 Philipp Winter. *Design Requirements for a Tor Censorship Analysis Tool*. Tech. rep. 2013-02-001. The Tor Project, 2013. URL: https://research.torproject.org/techreports/censorship-analysis-tool-2013-02-06.pdf
Contribution: I am the sole author of this paper.

P.9 Philipp Winter and Jedidiah R. Crandall. "The Great Firewall of China: How it Blocks Tor and Why it is Hard to Pinpoint". In: *USENIX ;login* 37.6 (2012), pp. 42–50. URL: http://www.cs.kau.se/philwint/pdf/

usenix-login-2012.pdf
Contribution: I contributed the Section discussing the GFW's attempts to block Tor.

P.10 Philipp Winter and Stefan Lindskog. "How the Great Firewall of China is Blocking Tor". In: *Free and Open Communications on the Internet*. USENIX, 2012. URL: https://www.usenix.org/system/files/conference/foci12/foci12-final2.pdf
Contribution: Stefan read drafts and provided feedback. The remainder was done by me.

P.11 Philipp Winter and Stefan Lindskog. *How China Is Blocking Tor*. Tech. rep. Karlstad University, 2012. URL: http://www.cs.kau.se/philwint/pdf/torblock2012.pdf
Contribution: Stefan read drafts and provided feedback. The remainder was done by me.

P.12 Martin Drašar, Jan Vykopal, and Philipp Winter. "Flow-based Bruteforce Attack Detection". In: *Advances in IT Early Warning*. Ed. by Peter Schoo, Markus Zeilinger, and Eckehard Hermann. Fraunhofer Verlag, 2013. URL: http://www.cs.kau.se/philwint/pdf/early-warning-2013.pdf
Contribution: I contributed Section 2.4.

Glossary

AES advanced encryption standard. 98, 103

API application programming interface. 22, 64, 68

ARMA auto-regressive moving average. 32, 33

AS autonomous system. 16, 17, 72, 77, 78

ASN autonomous system number. 11, 14, 16, 61, 74

CERNET Chinese Educational and Research Network. 45, 48, 49, 51, 52, 54

DHCP dynamic host configuration protocol. 39

DNS domain name system. 16, 25, 52, 59, 69–78, 81, 112, 113

DNSSEC domain name system security extensions. 72

DPI deep packet inspection. 14, 19–21, 27, 60, 61, 66, 81, 83, 119

GFW Great Firewall of China. 5–7, 9, 10, 12–14, 16–21, 23, 25–28, 36, 40, 41, 43, 47, 48, 51, 52, 55, 82, 83, 112–114, 119

HKDF HMAC-based key derivation function. 98

HMAC keyed-hash message authentication code. 97, 98, 103, 109

HTTP hyper text transfer protocol. 9, 14, 25, 58, 61, 67, 69, 72, 74, 77, 81, 83, 107, 108, 112, 116, 120

HTTPS hyper text transfer protocol secure. 62, 64, 77, 78, 91, 95

ICMP Internet control message protocol. 12, 17, 42, 48, 59, 60

IP Internet protocol. 3, 4, 9, 10, 12, 13, 15–19, 23, 25–27, 30–32, 35–37, 39, 41–43, 47–49, 52, 55, 58–62, 64, 65, 74, 75, 77, 78, 88, 91, 92, 95, 96, 98, 108, 112–115

IPID IP identification number. 30–33, 36–39

ISP Internet service provider. 16, 61, 74, 75, 78, 79, 118

IXP Internet exchange point. 52, 54

LAN local area network. 108

MAC message authentication code. 90, 92–95, 105

MitM man-in-the-middle. 79, 86, 88, 96, 120

MTU maximum transmission unit. 100, 105, 108

NIDS network intrusion detection system. 21

PRNG pseudo-random number generator. 98, 99, 101, 103

SNI server name identifier. 59, 60

SSL secure socket layer. 72, 74–79

TBB Tor Browser Bundle. 58, 63

TCP transmission control protocol. 3, 5, 12–14, 17, 21–23, 25, 27, 29, 30, 35, 39, 41–43, 52, 53, 55, 58–60, 66, 67, 71, 79, 82, 86, 90, 96, 98, 100, 108, 109, 112, 115, 119

TLS transport layer security. 15, 21, 22, 59, 60, 81, 82, 88, 100, 108, 109, 120

TTL time-to-live value. 17, 18, 42

UDH uniform Diffie-Hellman. 93–96, 105, 108

UDP user datagram protocol. 17, 60, 71

UTC universal time, coordinated. 13, 18, 49

VoIP voice over IP. 120

VPN virtual private network. 3, 82, 83, 87, 92, 109, 115

VPS virtual private system. 11–15, 21, 35, 36, 39–41, 56

Bibliography

[1] Reporters Without Borders. *Internet Enemies*. 2012. URL: http://march12.rsf.org/en/ (cit. on p. 1).

[2] *The NSA files*. URL: http://www.theguardian.com/us-news/the-nsa-files (cit. on p. 1).

[3] Roger Dingledine, Nick Mathewson, and Paul Syverson. "Tor: The Second-Generation Onion Router". In: *USENIX Security Symposium*. USENIX, 2004. URL: https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf (cit. on pp. 3, 70).

[4] The Tor Project. *Bridge easily detected by GFW*. URL: https://bugs.torproject.org/4185 (cit. on p. 9).

[5] Roger Dingledine and Nick Mathewson. *Design of a blocking-resistant anonymity system*. Tech. rep. The Tor Project, 2006. URL: https://svn.torproject.org/svn/projects/design-paper/blocking.html (cit. on p. 9).

[6] The Tor Project. *Torproject.org Blocked by GFW in China: Sooner or Later?* URL: https://blog.torproject.org/blog/torprojectorg-blocked-gfw-china-sooner-or-later (cit. on p. 9).

[7] The Tor Project. *Tor partially blocked in China*. URL: https://blog.torproject.org/blog/tor-partially-blocked-china (cit. on p. 9).

[8] The Tor Project. *Picturing Tor censorship in China*. URL: https://blog.torproject.org/blog/picturing-tor-censorship-china (cit. on p. 9).

[9] The Tor Project. *China blocking Tor: Round Two*. URL: https://blog.torproject.org/blog/china-blocking-tor-round-two (cit. on p. 9).

[10] Zhen Ling et al. “Extensive Analysis and Large-Scale Empirical Evaluation of Tor Bridge Discovery”. In: *INFOCOM*. IEEE, 2012. URL: http://www.cs.uml.edu/~xinwenfu/paper/Bridge.pdf (cit. on pp. 9, 91, 117).

[11] Amazon Web Services LLC. *Amazon Elastic Compute Cloud (Amazon EC2)*. URL: https://aws.amazon.com/ec2/ (cit. on p. 11).

[12] The Tor Project. *Tor Cloud*. URL: https://cloud.torproject.org (cit. on p. 11).

[13] OpenNet Initiative. *Singapore*. URL: http://opennet.net/research/profiles/singapore (cit. on pp. 11, 70).

[14] Roya Ensafi et al. “Detecting Intentional Packet Drops on the Internet via TCP/IP Side Channels”. In: *Passive and Active Measurement Conference*. Springer, 2014. URL: http://arxiv.org/pdf/1312.5739.pdf (cit. on pp. 12, 28, 30–32, 37, 114).

[15] *netfilter/iptables project homepage*. URL: http://www.netfilter.org (cit. on p. 13).

[16] Richard Clayton, Steven J. Murdoch, and Robert N. M. Watson. “Ignoring the Great Firewall of China”. In: *Privacy Enhancing Technologies*. Springer, 2006. URL: http://www.cl.cam.ac.uk/~rnc1/ignoring.pdf (cit. on pp. 14, 25, 81, 112).

[17] Hal Roberts. *Local Control: About 95% of Chinese Web Traffic is Local*. 2011. URL: https://blogs.law.harvard.edu/hroberts/2011/08/15/local-control-about-95-of-chinese-web-traffic-is-local/ (cit. on p. 14).

[18] Team Cymru, Inc. *IP to ASN Mapping*. URL: https://www.team-cymru.org/Services/ip-to-asn.html (cit. on pp. 16, 61).

[19] Hal Roberts et al. “Mapping Local Internet Control”. In: *Computer Communications Workshop*. IEEE, 2011. URL: https://cyber.law.harvard.edu/netmaps/mlic_20110513.pdf (cit. on p. 16).

[20] Xueyang Xu, Z. Morley Mao, and J. Alex Halderman. "Internet Censorship in China: Where Does the Filtering Occur?" In: *Passive and Active Measurement Conference*. Springer, 2011. URL: http://www.eecs.umich.edu/~zmao/Papers/china-censorship-pam11.pdf (cit. on pp. 16, 53, 112).

[21] The Tor Project. *Tor users via bridges*. URL: https://metrics.torproject.org/users.html?graph=bridge-users&start=2012-01-01&end=2012-06-18&country=cn&dpi=72#bridge-users (cit. on p. 19).

[22] The Tor Project. *Knock Knock Knockin' on Bridges' Doors*. URL: https://blog.torproject.org/blog/knock-knock-knockin-bridges-doors (cit. on p. 19).

[23] Philipp Winter. *Selected Research Papers in Internet Censorship*. URL: http://censorbib.nymity.ch (cit. on p. 20).

[24] The Tor Project. *obfsproxy*. URL: https://www.torproject.org/projects/obfsproxy (cit. on pp. 20, 61, 84, 115).

[25] The Tor Project. *Pluggable Transports*. URL: https://www.torproject.org/docs/pluggable-transports (cit. on p. 21).

[26] Thomas H. Ptacek and Timothy N. Newsham. *Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection*. Tech. rep. Secure Networks, Inc., 1998. URL: http://cs.unc.edu/~fabian/course_papers/PtacekNewsham98.pdf (cit. on pp. 21, 81, 112).

[27] Jedidiah R. Crandall et al. "ConceptDoppler: A Weather Tracker for Internet Censorship". In: *Computer and Communications Security*. ACM, 2007. URL: http://www.csd.uoc.gr/~hy558/papers/conceptdoppler.pdf (cit. on pp. 21, 25, 113).

[28] Dug Song. *fragroute*. URL: http://monkey.org/~dugsong/fragroute/ (cit. on p. 21).

[29] Jong Chun Park and Jedidiah R. Crandall. "Empirical Study of a National-Scale Distributed Intrusion Detection System: Backbone-Level Filtering of HTML Responses in China". In: *Distributed Computing Systems*. IEEE, 2010. URL: http://www.cs.unm.edu/~crandall/icdcs2010.pdf (cit. on pp. 21, 25).

[30] Harald Welte. *The netfilter.org "libnetfilter_queue" project*. URL: http://www.netfilter.org/projects/libnetfilter_queue/ (cit. on p. 22).

[31] Michael Carl Tschantz et al. *On Modeling the Costs of Censorship*. Tech. rep. UC Berkeley, 2014. URL: http://arxiv.org/pdf/1409.3211v1 (cit. on p. 23).

[32] Philipp Winter and Stefan Lindskog. "How the Great Firewall of China is Blocking Tor". In: *Free and Open Communications on the Internet*. USENIX, 2012. URL: https://www.usenix.org/system/files/conference/foci12/foci12-final2.pdf (cit. on pp. 25, 52, 60, 123).

[33] Graham Lowe, Patrick Winters, and Michael L. Marcus. *The Great DNS Wall of China*. Tech. rep. New York University, 2007. URL: http://cs.nyu.edu/~pcw216/work/nds/final.pdf (cit. on pp. 25, 112).

[34] Sparks et al. "The Collateral Damage of Internet Censorship by DNS Injection". In: *Computer Communication Review* 42.3 (2012). URL: http://conferences.sigcomm.org/sigcomm/2012/paper/ccr-paper266.pdf (cit. on pp. 25, 81, 112).

[35] Joss Wright. *Regional Variation in Chinese Internet Filtering*. Tech. rep. University of Oxford, 2012. URL: http://www.cs.kau.se/philwint/censorbib/pdf/Wright2012.pdf (cit. on pp. 25, 112).

[36] Tao Zhu et al. "The Velocity of Censorship: High-Fidelity Detection of Microblog Post Deletions". In: *USENIX Security Symposium*. USENIX, 2013. URL: https://www.usenix.org/system/files/conference/usenixsecurity13/sec13-paper_zhu.pdf (cit. on pp. 25, 112).

[37] Isis Agora Lovecruft. *Automating Bridge Reachability Testing*. 2012. URL: https://bugs.torproject.org/6414 (cit. on p. 26).

[38] Qiyan Wang et al. "rBridge: User Reputation based Tor Bridge Distribution with Privacy Preservation". In: *Network and Distributed System Security*. The Internet Society, 2013. URL: http://www-users.cs.umn.edu/~hopper/rbridge_ndss13.pdf (cit. on pp. 26, 117).

[39] The Tor Project. *The Censorship Wiki*. URL: https://censorshipwiki.torproject.org (cit. on pp. 27, 55, 56, 58).

[40] Roya Ensafi et al. *Detecting Intentional Packet Drops on the Internet via TCP/IP Side Channels: Extended Version*. Tech. rep. University of New Mexico, 2013. URL: http://arxiv.org/abs/1312.5739 (cit. on pp. 28, 30, 32, 37).

[41] *Linux kernel source tree*. URL: http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/net/ipv4/inet_connection_sock.c (cit. on p. 29).

[42] *tcp(7) - Linux man page*. URL: http://linux.die.net/man/7/tcp (cit. on p. 29).

[43] Roya Ensafi et al. "Idle Port Scanning and Non-interference Analysis of Network Protocol Stacks Using Model Checking". In: *USENIX Security Symposium*. USENIX, 2010. URL: https://www.usenix.org/legacy/event/sec10/tech/full_papers/Ensafi.pdf (cit. on pp. 30, 114).

[44] The Tor Project. *Tor directory protocol, version 2*. URL: https://gitweb.torproject.org/torspec.git/blob/HEAD:/dir-spec.txt (cit. on p. 34).

[45] The Tor Project. *Relay descriptor archives*. URL: https://metrics.torproject.org/data.html#relaydesc (cit. on p. 34).

[46] MaxMind. *GeoIP2 City*. 2014. URL: http://www.maxmind.com/en/city (cit. on p. 34).

[47] *Tor Network Status*. URL: http://torstatus.blutmagie.de (cit. on p. 37).

[48] Alexa. *Alexa top sites in China*. URL: http://www.alexa.com/topsites/countries/CN (cit. on p. 37).

[49] Salvatore Sanfilippo. *hping*. 2006. URL: http://www.hping.org (cit. on pp. 41, 42).

[50] Anonymous. "Towards a Comprehensive Picture of the Great Firewall's DNS Censorship". In: *Free and Open Communications on the Internet*. USENIX, 2014. URL: https://www.usenix.org/system/files/conference/foci14/foci14-anonymous.pdf (cit. on pp. 52, 112).

[51] Ye Tian et al. "Topology Mapping and Geolocating for China's Internet". In: *Transactions on Parallel and Distributed Systems* 24.9 (2013), pp. 1908–1917. URL: http://censorbib.nymity.ch/pdf/Tian2013a.pdf (cit. on p. 53).

[52] *Submarine Cable Map 2014*. URL: http://submarine-cable-map-2014.telegeography.com (cit. on p. 53).

[53] Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. “ZMap: Fast Internet-Wide Scanning and its Security Applications”. In: *USENIX Security Symposium*. USENIX, 2013. URL: https://zmap.io/paper.pdf (cit. on pp. 53, 82).

[54] The Tor Project. *Directly connecting users from Iran*. URL: https://metrics.torproject.org/users.html?graph=direct-users&start=2013-01-01&end=2013-03-15&country=ir&events=off#direct-users (cit. on p. 56).

[55] *PlanetLab – An open platform for developing, deploying, and accessing planetary-scale services*. URL: https://www.planet-lab.org (cit. on pp. 56, 67, 70, 113).

[56] Jinliang Fan et al. *Cryptography-based Prefix-preserving Anonymization*. URL: http://www.cc.gatech.edu/computing/Telecomm/projects/cryptopan/ (cit. on p. 58).

[57] The Tor Project. *Tor Browser*. URL: https://www.torproject.org/projects/torbrowser.html.en (cit. on p. 58).

[58] Tim Wilde. *Great Firewall Tor Probing Circa 09 DEC 2011*. 2012. URL: https://gist.github.com/twilde/da3c7a9af01d74cd7de7 (cit. on p. 60).

[59] The Tor Project. *An update on the censorship in Ethiopia*. 2012. URL: https://blog.torproject.org/blog/update-censorship-ethiopia (cit. on p. 60).

[60] The Tor Project. *Iran blocks Tor; Tor releases same-day fix*. 2011. URL: https://blog.torproject.org/blog/iran-blocks-tor-tor-releases-same-day-fix (cit. on p. 60).

[61] Joss Wright, Tulio de Souza, and Ian Brown. “Fine-Grained Censorship Mapping: Information Sources, Legality and Ethics”. In: *Free and Open Communications on the Internet*. USENIX, 2011. URL: http://static.usenix.org/event/foci11/tech/final_files/Wright.pdf (cit. on pp. 61, 73).

[62] Runa A. Sandvik. *Forensic Analysis of the Tor Browser Bundle on OS X, Linux, and Windows*. Tech. rep. The Tor Project, 2013. URL: https://research.torproject.org/techreports/tbb-forensic-analysis-2013-06-28.pdf (cit. on p. 63).

[63] The Tor Project. *GetTor e-mail autoresponder*. URL: https://www.torproject.org/projects/gettor.html.en (cit. on p. 63).

[64] Arturo Filastò and Jacob Appelbaum. "OONI: Open Observatory of Network Interference". In: *Free and Open Communications on the Internet*. USENIX, 2012. URL: https://www.usenix.org/system/files/conference/foci12/foci12-final12.pdf (cit. on pp. 64, 67, 70, 113).

[65] *py2exe*. URL: http://www.py2exe.org (cit. on p. 64).

[66] Kevin P. Dyer et al. "Protocol Misidentification Made Easy with Format-Transforming Encryption". In: *Computer and Communications Security*. ACM, 2013. URL: http://eprint.iacr.org/2012/494.pdf (cit. on pp. 66, 115, 120).

[67] Jillian C. York. *Government Internet Surveillance Starts With Eyes Built in the West*. 2011. URL: https://www.eff.org/deeplinks/2011/09/government-internet-surveillance-starts-eyes-built (cit. on p. 66).

[68] Nicholas Weaver, Robin Sommer, and Vern Paxson. "Detecting Forged TCP Reset Packets". In: *Network and Distributed System Security*. The Internet Society, 2009. URL: http://www.icsi.berkeley.edu/pubs/networking/ndss09-resets.pdf (cit. on pp. 66, 112).

[69] *RIPE Atlas*. URL: https://atlas.ripe.net (cit. on pp. 67, 68, 70).

[70] *Herdict*. URL: http://www.herdict.org (cit. on pp. 67, 70).

[71] Constantine Dovrolis et al. "Measurement Lab: Overview and an Invitation to the Research Community". In: *Computer Communication Review* 40.3 (2010), pp. 53–56. URL: http://www.sigcomm.org/sites/default/files/ccr/papers/2010/July/1823844-1823853.pdf (cit. on p. 70).

[72] Yakov Rekhter et al. *Address Allocation for Private Internets*. 1996. URL: https://www.ietf.org/rfc/rfc1918.txt (cit. on p. 72).

[73] Yaman Akdeniz. *Report of the OSCE Representative on Freedom of the Media on Turkey and The Internet Censorship*. Tech. rep. OSCE, 2010. URL: http://www.osce.org/fom/41091?download=true (cit. on p. 74).

[74] *Engelliweb*. URL: http://engelliweb.com (cit. on p. 74).

[75] International Business Times. *Kremlin Blocks Four Opposition Websites As Ukraine Crisis Brews*. 2014. URL: http://www.ibtimes.com/kremlin-blocks-four-opposition-websites-ukraine-crisis-brews-1561356 (cit. on p. 77).

[76] Federal Service for Supervision of Communications, Information Technology, and Mass Media. 2013. URL: http://rkn.gov.ru/docs/Analysys_and_recommendations_comments_fin.pdf (cit. on p. 77).

[77] John-Paul Verkamp and Minaxi Gupta. "Inferring Mechanics of Web Censorship Around the World". In: *Free and Open Communications on the Internet*. USENIX, 2012. URL: https://www.usenix.org/system/files/conference/foci12/foci12-final1.pdf (cit. on p. 77).

[78] Mark Schloesser et al. *Project Sonar: IPv4 SSL Certificates*. URL: https://scans.io/study/sonar.ssl (cit. on p. 77).

[79] Moscow Institute of Physics and Technology. URL: http://board.rt.mipt.ru/?read=8820778 (cit. on p. 78).

[80] Olli-Pekka Niemi, Antti Levomäki, and Jukka Manner. "Dismantling Intrusion Prevention Systems (Demo)". In: *SIGCOMM*. ACM, 2012. URL: http://conferences.sigcomm.org/sigcomm/2012/paper/sigcomm/p285.pdf (cit. on p. 81).

[81] Mark Handley, Vern Paxson, and Christian Kreibich. "Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics". In: *USENIX Security Symposium*. USENIX Association, 2001. URL: http://static.usenix.org/events/sec01/full_papers/handley/handley.pdf (cit. on p. 81).

[82] Marcel Dischinger et al. "Detecting BitTorrent Blocking". In: *Internet Measurement Conference*. ACM, 2008. URL: http://www.mpi-sws.org/~mdischin/papers/08_imc_blocking.pdf (cit. on p. 81).

[83] Christopher Rhoads and Loretta Chao. *Iran's Web Spying Aided By Western Technology*. 2009. URL: http://ohm.ecce.admu.edu.ph/wiki/pub/Main/ResearchProjects/Irans_Web_Spying_Aided_By_Western_Technology_-_WSJ.com.pdf (cit. on p. 81).

[84] Jillian C. York. *Government Internet Surveillance Starts With Eyes Built in the West*. 2011. URL: https://www.eff.org/deeplinks/2011/09/government-internet-surveillance-starts-eyes-built (cit. on p. 81).

[85] Charles Arthur. *China tightens 'Great Firewall' internet control with new technology*. 2012. URL: http://www.guardian.co.uk/technology/2012/dec/14/china-tightens-great-firewall-internet-control (cit. on p. 82).

[86] The Tor Project. *GFW actively probes obfs2 bridges*. URL: https://bugs.torproject.org/8591 (cit. on p. 82).

[87] Zachary Weinberg et al. "StegoTorus: A Camouflage Proxy for the Tor Anonymity System". In: *Computer and Communications Security*. ACM, 2012. URL: http://web.mit.edu/frankw/www/papers/ccs2012.pdf (cit. on pp. 83, 100, 115, 120).

[88] Hooman Mohajeri Moghaddam et al. "SkypeMorph: Protocol Obfuscation for Tor Bridges". In: *Computer and Communications Security*. ACM, 2012. URL: http://www.cypherpunks.ca/~iang/pubs/skypemorph-ccs.pdf (cit. on pp. 83, 100, 114, 120).

[89] Amir Houmansadr et al. "I want my voice to be heard: IP over Voice-over-IP for unobservable censorship circumvention". In: *Network and Distributed System Security*. The Internet Society, 2013. URL: http://www.cs.utexas.edu/~amir/papers/FreeWave.pdf (cit. on pp. 83, 115, 120).

[90] Wenxuan Zhou et al. "SWEET: Serving the Web by Exploiting Email Tunnels". In: *Hot Topics in Privacy Enhancing Technologies*. Springer, 2013. URL: http://petsymposium.org/2013/papers/zhou-censorship.pdf (cit. on pp. 83, 115).

[91] The Tor Project. *obfs2 (The Twobfuscator)*. URL: https://gitweb.torproject.org/obfsproxy.git/blob/HEAD:/doc/obfs2/protocol-spec.txt (cit. on p. 83).

[92] The Tor Project. *obfs3 (The Threebfuscator)*. URL: https://gitweb.torproject.org/user/asn/obfsproxy.git/blob/HEAD:/doc/obfs3/obfs3-protocol-spec.txt (cit. on pp. 83, 93, 102).

[93] Brandon Wiley. *Dust: A Blocking-Resistant Internet Transport Protocol*. Tech. rep. University of Texas at Austin, 2011. URL: http://blanu.net/Dust.pdf (cit. on p. 83).

[94] *Viewing cable 09MUSCAT1039, SKYPE CRACKDOWN IN OMAN*. 2009. URL: http://wikileaks.org/cable/2009/11/09MUSCAT1039.html (cit. on p. 83).

[95] *Russian "Clean Internet" experiment gets green light*. 2013. URL: http://rt.com/politics/anti-pedophile-safe-internet-russian-169/ (cit. on p. 83).

[96] Small Media. *Iranian Internet Infrastructure and Policy Report: Election Edition 2013 (April - June)*. 2013. URL: http://smallmedia.org.uk/IIIPJune.pdf (cit. on p. 83).

[97] Alberto Dainotti et al. "Analysis of Country-wide Internet Outages Caused by Censorship". In: *Internet Measurement Conference*. ACM, 2011. URL: http://www.caida.org/publications/papers/2011/outages_censorship/outages_censorship.pdf (cit. on p. 84).

[98] Eva Galperin and Jillian C. York. *Syria Goes Dark*. 2012. URL: https://www.eff.org/deeplinks/2012/11/syria-goes-dark (cit. on p. 84).

[99] Martin Johnson. *China, GitHub and the man-in-the-middle*. 2013. URL: https://en.greatfire.org/blog/2013/jan/china-github-and-man-middle (cit. on p. 86).

[100] Ronald L. Rivest, Adi Shamir, and David A. Wagner. *Time-lock Puzzles and Timed-release Crypto*. Tech. rep. Massachusetts Institute of Technology, 1996. URL: http://www.hashcash.org/papers/time-lock.pdf (cit. on p. 87).

[101] Ben Laurie and Richard Clayton. ""Proof-of-Work" Proves Not to Work". In: *Workshop on the Economics of Information Security*. 2004. URL: http://www.cl.cam.ac.uk/~rnc1/proofwork2.pdf (cit. on p. 88).

[102] Joseph Salowey et al. *RFC 5077: Transport Layer Security (TLS) Session Resumption without Server-Side State*. 2008. URL: https://tools.ietf.org/html/rfc5077 (cit. on p. 88).

[103] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. *RFC 2104: HMAC: Keyed-Hashing for Message Authentication*. 1997. URL: https://www.ietf.org/rfc/rfc2104.txt (cit. on p. 90).

[104] Tero Kivinen and Mika Kojo. *RFC 3526: More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)*. 2003. URL: http://tools.ietf.org/html/rfc3526 (cit. on pp. 93, 94).

[105] Hugo Krawczyk and Pasi Eronen. *RFC 5869: HMAC-based Extract-and-Expand Key Derivation Function (HKDF)*. 2010. URL: https://tools.ietf.org/html/rfc5869 (cit. on p. 98).

[106] Manuel Crotti et al. "Traffic Classification through Simple Statistical Fingerprinting". In: *SIGCOMM Computer Communication Review* 37.1 (2007). URL: http://www.sigcomm.org/sites/default/files/ccr/papers/2007/January/1198255-1198257.pdf (cit. on p. 99).

[107] Kevin P. Dyer et al. "Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail". In: *Security & Privacy*. IEEE, 2012. URL: http://kpdyer.com/publications/oakland2012-peekaboo.pdf (cit. on pp. 99, 102).

[108] Xiang Cai et al. "Touching from a Distance: Website Fingerprinting Attacks and Defenses". In: *Computer and Communications Security*. ACM, 2012. URL: http://www.cs.sunysb.edu/~xcai/fp.pdf (cit. on p. 99).

[109] Erik Hjelmvik and Wolfgang John. *Breaking and Improving Protocol Obfuscation*. Tech. rep. Chalmers University of Technology, 2010. URL: http://www.iis.se/docs/hjelmvik_breaking.pdf (cit. on p. 99).

[110] Andriy Panchenko et al. “Website Fingerprinting in Onion Routing Based Anonymization Networks”. In: *Workshop on Privacy in the Electronic Society*. ACM, 2011. URL: http://lorre.uni.lu/~andriy/papers/acmccs-wpes11-fingerprinting.pdf (cit. on p. 99).

[111] CAIDA. *Packet size distribution comparison between Internet links in 1998 and 2008*. 2010. URL: http://www.caida.org/research/traffic-analysis/pkt_size_distribution/graphs.xml (cit. on p. 100).

[112] Mohamad Jaber, Roberto G. Cascella, and Chadi Barakat. “Can we trust the inter-packet time for traffic classification?” In: *International Conference on Communications*. IEEE, 2011. URL: http://www-sop.inria.fr/members/Chadi.Barakat/ICC2011.pdf (cit. on p. 100).

[113] *ECRYPT II Yearly Report on Algorithms and Keysizes*. 2012. URL: http://www.ecrypt.eu.org/documents/D.SPA.20.pdf (cit. on p. 102).

[114] Dwayne C. Litzenberger. *PyCrypto*. URL: https://www.dlitz.net/software/pycrypto/ (cit. on p. 103).

[115] Andrew M. White et al. “Clear and Present Data: Opaque Traffic and its Security Implications for the Future”. In: *Network and Distributed System Security*. The Internet Society, 2013. URL: http://cs.unc.edu/~amw/resources/opaque.pdf (cit. on p. 109).

[116] Kurt Thomas, Chris Grier, and Vern Paxson. “Adapting Social Spam Infrastructure for Political Censorship”. In: *Large-Scale Exploits and Emergent Threats*. USENIX, 2012. URL: https://www.usenix.org/system/files/conference/leet12/leet12-final13_0.pdf (cit. on p. 111).

[117] John-Paul Verkamp and Minaxi Gupta. “Five Incidents, One Theme: Twitter Spam as a Weapon to Drown Voices of Protest”. In: *Free and Open Communications on the Internet*. USENIX, 2013. URL: https://www.usenix.org/system/files/tech-schedule/foci13-papers-archive.zip (cit. on p. 111).

[118] Le Chen, Chi Zhang, and Christo Wilson. “Tweeting Under Pressure: Analyzing Trending Topics and Evolving Word Choice on Sina Weibo”. In: *Conference on Online Social Networks*. ACM, 2013. URL:

http://www.ccs.neu.edu/home/cbw/pdf/weibo-cosn13.pdf (cit. on p. 112).

[119] Haixin Duan et al. “Hold-On: Protecting Against On-Path DNS Poisoning”. In: *Securing and Trusting Internet Names*. National Physical Laboratory, 2012. URL: http://conferences.npl.co.uk/satin/papers/satin2012-Duan.pdf (cit. on p. 112).

[120] Sheharbano Khattak et al. “Towards Illuminating a Censorship Monitor’s Model to Facilitate Evasion”. In: *Free and Open Communications on the Internet*. USENIX, 2013. URL: https://www.usenix.org/system/files/tech-schedule/foci13-papers-archive.zip (cit. on p. 112).

[121] Collin Anderson. *The Hidden Internet of Iran: Private Address Allocations on a National Network*. Tech. rep. 2012. URL: http://arxiv.org/pdf/1209.6398v1 (cit. on p. 113).

[122] Collin Anderson. *Dimming the Internet: Detecting Throttling as a Mechanism of Censorship in Iran*. Tech. rep. University of Pennsylvania, 2013. URL: http://arxiv.org/pdf/1306.4361v1.pdf (cit. on p. 113).

[123] Simurgh Aryan, Homa Aryan, and J. Alex Halderman. “Internet Censorship in Iran: A First Look”. In: *Free and Open Communications on the Internet*. USENIX, 2013. URL: https://www.usenix.org/system/files/tech-schedule/foci13-papers-archive.zip (cit. on p. 113).

[124] Zubair Nabi. “The Anatomy of Web Censorship in Pakistan”. In: *Free and Open Communications on the Internet*. USENIX, 2013. URL: https://www.usenix.org/system/files/tech-schedule/foci13-papers-archive.zip (cit. on p. 113).

[125] John-Paul Verkamp and Minaxi Gupta. “Inferring Mechanics of Web Censorship Around the World”. In: *Free and Open Communications on the Internet*. USENIX, 2012. URL: https://www.usenix.org/system/files/conference/foci12/foci12-final1.pdf (cit. on p. 113).

[126] Andrea Di Florio et al. “Bypassing Censorship: a proven tool against the recent Internet censorship in Turkey”. In: *Reliability and Security Data Analysis*. IEEE, 2014. URL: http://www.gains-project.eu/wp-content/uploads/2014/09/PID3398055.pdf (cit. on p. 113).

[127] Andreas Sfakianakis, Elias Athanasopoulos, and Sotiris Ioannidis. "CensMon: A Web Censorship Monitor". In: *Free and Open Communications on the Internet*. USENIX, 2011. URL: http://static.usenix.org/event/foci11/tech/final_files/Sfakianakis.pdf (cit. on p. 113).

[128] The Tor Project. *Raw reports*. URL: https://ooni.torproject.org/reports/ (cit. on p. 113).

[129] Qiyan Wang et al. "CensorSpoofer: Asymmetric Communication using IP Spoofing for Censorship-Resistant Web Browsing". In: *Computer and Communications Security*. ACM, 2012. URL: http://hatswitch.org/~nikita/papers/censorspoofer.pdf (cit. on p. 114).

[130] Shuai Li, Mike Schliep, and Nick Hopper. "Facet: Streaming over Videoconferencing for Censorship Circumvention". In: *Workshop on Privacy in the Electronic Society*. ACM, 2014. URL: http://www-users.cs.umn.edu/~hopper/facet-wpes14.pdf (cit. on p. 114).

[131] David Fifield et al. "Evading Censorship with Browser-Based Proxies". In: *Privacy Enhancing Technologies Symposium*. Springer, 2012. URL: http://crypto.stanford.edu/flashproxy/flashproxy.pdf (cit. on p. 115).

[132] Daniel Luchaup et al. "LibFTE: A Toolkit for Constructing Practical, Format-Abiding Encryption Schemes". In: *USENIX Security Symposium*. USENIX, 2014. URL: https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-luchaup.pdf (cit. on p. 115).

[133] David Fifield, Gabi Nakibly, and Dan Boneh. "OSS: Using Online Scanning Services for Censorship Circumvention". In: *Privacy Enhancing Technologies Symposium*. Springer, 2013. URL: http://crypto.stanford.edu/~dabo/pubs/papers/redirects.pdf (cit. on p. 115).

[134] Daiyuu Nobori and Yasushi Shinjo. "VPN Gate: A Volunteer-Organized Public VPN Relay System with Blocking Resistance for Bypassing Government Censorship Firewalls". In: *Networked Systems Design and Implementation*. USENIX, 2014. URL: https://www.usenix.org/system/files/conference/nsdi14/nsdi14-paper-nobori.pdf (cit. on p. 115).

[135] Chad Brubaker, Amir Houmansadr, and Vitaly Shmatikov. "CloudTransport: Using Cloud Storage for Censorship-Resistant Networking". In: *Privacy Enhancing Technologies Symposium*. Springer, 2014. URL: https://www.cs.utexas.edu/~shmat/shmat_pets14.pdf (cit. on pp. 115, 120).

[136] Ben Jones et al. "Facade: High-Throughput, Deniable Censorship Circumvention Using Web Search". In: *Free and Open Communications on the Internet*. USENIX, 2014. URL: https://www.usenix.org/system/files/conference/foci14/foci14_jones-8-8-14.pdf (cit. on p. 115).

[137] Amir Houmansadr, Chad Brubaker, and Vitaly Shmatikov. "The Parrot is Dead: Observing Unobservable Network Communications". In: *Symposium on Security & Privacy*. IEEE, 2013. URL: http://www.cs.utexas.edu/~amir/papers/parrot.pdf (cit. on p. 116).

[138] Jeroen Massar et al. "JumpBox – A Seamless Browser Proxy for Tor Pluggable Transports". In: *Security and Privacy in Communication Networks*. Springer, 2014. URL: http://jeroen.massar.ch/publications/files/SECURECOMM2014-JumpBox.pdf (cit. on p. 116).

[139] John Geddes, Max Schuchard, and Nicholas Hopper. "Cover Your ACKs: Pitfalls of Covert Channel Censorship Circumvention". In: *Computer and Communications Security*. ACM, 2013. URL: http://www-users.cs.umn.edu/~hopper/ccs13-cya.pdf (cit. on p. 116).

[140] Marc Waldman, Aviel D. Rubin, and Lorrie Faith Cranor. "Publius: A robust, tamper-evident, censorship-resistant web publishing system". In: *USENIX Security Symposium*. USENIX, 2000. URL: http://www.eecs.harvard.edu/~mema/courses/cs264/papers/waldman00publius.pdf (cit. on p. 116).

[141] Marc Waldman and David Mazières. "Tangler: A Censorship-Resistant Publishing System Based On Document Entanglements". In: *Computer and Communications Security*. ACM, 2001. URL: http://www.cs.nyu.edu/~waldman/tangler.ps (cit. on p. 116).

[142] Adam Stubblefield and Dan S. Wallach. *Dagster: Censorship-Resistant Publishing Without Replication*. Tech. rep. Rice University, 2001. URL: http://www.cs.rice.edu/~dwallach/pub/dagster-tr.pdf (cit. on p. 116).

[143] Nick Feamster et al. "Infranet: Circumventing Web Censorship and Surveillance". In: *USENIX Security Symposium*. USENIX, 2002. URL: http://wind.lcs.mit.edu/papers/usenixsec2002.pdf (cit. on p. 116).

[144] Sam Burnett, Nick Feamster, and Santosh Vempala. "Chipping Away at Censorship Firewalls with User-Generated Content". In: *USENIX Security Symposium*. USENIX, 2010. URL: http://www.usenix.org/event/sec10/tech/full_papers/Burnett.pdf (cit. on p. 116).

[145] Luca Invernizzi, Christopher Kruegel, and Giovanni Vigna. "Message In A Bottle: Sailing Past Censorship". In: *Annual Computer Security Applications Conference*. ACM, 2013. URL: http://seclab.cs.ucsb.edu/media/uploads/papers/invernizzi_miab_acsac_2013.pdf (cit. on p. 116).

[146] Christopher Connolly et al. "TRIST: Circumventing Censorship with Transcoding-Resistant Image Steganography". In: *Free and Open Communications on the Internet*. USENIX, 2014. URL: https://www.usenix.org/system/files/conference/foci14/foci14-connolly.pdf (cit. on p. 116).

[147] Yair Sovran, Alana Libonati, and Jinyang Li. "Pass it on: Social Networks Stymie Censors". In: *International Workshop on Peer-to-Peer Systems*. USENIX, 2008. URL: http://www.news.cs.nyu.edu/~jinyang/pub/kalei_iptps08.pdf (cit. on p. 117).

[148] Mohammad Mahdian. "Fighting Censorship with Algorithms". In: *International Conference on Fun with Algorithms*. Springer, 2010. URL: http://mahdian.org/censorship.pdf (cit. on p. 117).

[149] Damon McCoy, Jose Andre Morales, and Kirill Levchenko. "Proximax: A Measurement Based System for Proxies Dissemination". In: *Financial Cryptography and Data Security*. Springer, 2011. URL: http://cseweb.ucsd.edu/~klevchen/mml-fc11.pdf (cit. on p. 117).

[150] The Tor Project. *The bridge distribution database*. URL: https://gitweb.torproject.org/bridgedb.git (cit. on p. 117).

[151] Josh Karlin et al. “Decoy Routing: Toward Unblockable Internet Communication”. In: *Free and Open Communications on the Internet*. USENIX, 2011. URL: http://static.usenix.org/event/foci11/tech/final_files/Karlin.pdf (cit. on p. 117).

[152] Eric Wustrow et al. “Telex: Anticensorship in the Network Infrastructure”. In: *USENIX Security Symposium*. USENIX, 2011. URL: http://www.usenix.org/event/sec11/tech/full_papers/Wustrow.pdf (cit. on p. 117).

[153] Amir Houmansadr et al. “Cirripede: Circumvention Infrastructure using Router Redirection with Plausible Deniability”. In: *Computer and Communications Security*. ACM, 2011. URL: http://hatswitch.org/~nikita/papers/cirripede-ccs11.pdf (cit. on p. 117).

[154] Max Schuchard et al. “Routing Around Decoys”. In: *Computer and Communications Security*. ACM, 2012. URL: http://www-users.cs.umn.edu/~hopper/decoy-ccs12.pdf (cit. on p. 117).

[155] Amir Houmansadr, Edmund L. Wong, and Vitaly Shmatikov. “No Direction Home: The True Cost of Routing Around Decoys”. In: *Network and Distributed System Security*. The Internet Society, 2014. URL: http://dedis.cs.yale.edu/dissent/papers/nodirection.pdf (cit. on p. 117).

[156] Eric Wustrow, Colleen M. Swanson, and J. Alex Halderman. “TapDance: End-to-Middle Anticensorship without Flow Blocking”. In: *USENIX Security Symposium*. USENIX, 2014. URL: https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-wustrow.pdf (cit. on p. 118).

[157] Marek Majkowski. *SSL fingerprinting for p0f*. 2012. URL: https://idea.popcount.org/2012-06-17-ssl-fingerprinting-for-p0f/ (cit. on p. 120).

[158] Andrea Bittau et al. “The case for ubiquitous transport-level encryption”. In: *USENIX Security Symposium*. USENIX, 2010. URL: http://tcpcrypt.org/tcpcrypt.pdf (cit. on p. 120).

[159] *TCP Increased Security (tcpinc)*. URL: https://datatracker.ietf.org/wg/tcpinc/charter/ (cit. on p. 120).

[160] David Fifield. *meek*. URL: https://trac.torproject.org/projects/tor/wiki/doc/meek (cit. on p. 120).

[161] Roya Ensafi et al. *Large-scale Spatiotemporal Characterization of Inconsistencies in the World's Largest Firewall*. Tech. rep. University of New Mexico, 2014. URL: http://arxiv.org/pdf/1410.0735 (cit. on p. 121).

[162] Collin Anderson, Philipp Winter, and Roya. "Global Censorship Detection over the RIPE Atlas Network". In: *Free and Open Communications on the Internet*. USENIX, 2014. URL: http://cartography.io/foci2014.pdf (cit. on p. 121).

[163] Philipp Winter et al. "Spoiled Onions: Exposing Malicious Tor Exit Relays". In: *Privacy Enhancing Technologies Symposium*. Springer, 2014. URL: http://www.cs.kau.se/philwint/spoiled_onions/pets2014.pdf (cit. on p. 121).

[164] Philipp Winter and Stefan Lindskog. *Spoiled Onions: Exposing Malicious Tor Exit Relays*. Tech. rep. Karlstad University, 2014. URL: http://veri.nymity.ch/spoiled_onions/techreport.pdf (cit. on p. 122).

[165] Philipp Winter, Tobias Pulls, and Juergen Fuss. "ScrambleSuit: A Polymorphic Network Protocol to Circumvent Censorship". In: *Workshop on Privacy in the Electronic Society*. ACM, 2013. URL: http://www.cs.kau.se/philwint/pdf/wpes2013.pdf (cit. on p. 122).

[166] Philipp Winter, Tobias Pulls, and Juergen Fuss. *ScrambleSuit: A Polymorph Network Protocol to Circumvent Censorship*. Tech. rep. http://www.cs.kau.se/philwint/pdf/scramblesuit2013.pdf. Karlstad University, 2013 (cit. on p. 122).

[167] Philipp Winter. "Towards a Censorship Analyser for Tor". In: *Free and Open Communications on the Internet*. USENIX, 2013. URL: https://www.usenix.org/system/files/tech-schedule/foci13-papers-archive.zip (cit. on p. 122).

[168] Philipp Winter. *Design Requirements for a Tor Censorship Analysis Tool*. Tech. rep. 2013-02-001. The Tor Project, 2013. URL: https://research.torproject.org/techreports/censorship-analysis-tool-2013-02-06.pdf (cit. on p. 122).

[169] Philipp Winter and Jedidiah R. Crandall. “The Great Firewall of China: How it Blocks Tor and Why it is Hard to Pinpoint”. In: *USENIX ;login* 37.6 (2012), pp. 42–50. URL: http://www.cs.kau.se/philwint/pdf/usenix-login-2012.pdf (cit. on p. 122).

[170] Philipp Winter and Stefan Lindskog. *How China Is Blocking Tor*. Tech. rep. Karlstad University, 2012. URL: http://www.cs.kau.se/philwint/pdf/torblock2012.pdf (cit. on p. 123).

[171] Martin Drašar, Jan Vykopal, and Philipp Winter. “Flow-based Bruteforce Attack Detection”. In: *Advances in IT Early Warning*. Ed. by Peter Schoo, Markus Zeilinger, and Eckehard Hermann. Fraunhofer Verlag, 2013. URL: http://www.cs.kau.se/philwint/pdf/early-warning-2013.pdf (cit. on p. 123).