The Utility of the Tor Network
Qualification Exam Report

HAMMAS BIN TANVEER, The University of Iowa

1 ABSTRACT

Tor is the most used anonymity network boasting more than 2 million users everyday[28]. Tor aims at providing high user anonymity, while maintaining low latency, to users mostly in censored or highly surveilled regions. In order to provide anonymity, Tor faces challenges on three fronts. First, Tor relies on volunteer owned and operated network resources which are limited in number. Second, state actors operating in censored regions try to prevent access to the Tor network. Third, Tor users face differential treatment by end servers. These problems contribute to the degradation of the Tor network. In this paper, I provide an analysis of the problems that the Tor network faces at the user, network and the end server level.

2 AN INTRODUCTION TO TOR

In recent times, global internet censorship and surveillance have been on a constant rise. Newer countries are partaking and deploying more sophisticated censorship and surveillance techniques than ever, for example the great China Firewall and the PRISM surveillance program [34] [11] [1]. This has led to a "chilling effect" on internet users [18] – *i.e* average internet users becoming more reluctant to engage in standard activities on the internet. Furthermore, it has also led to prosecution of minorities and groups involved in sensitive reporting [5]. The rising privacy concerns coupled with a will and need for a free and open internet has driven people towards maintaining anonymity online. This has led to a migration towards anonymity networks; one such anonymity network is Tor [27]. Tor is an anonymity software with over 2 million daily users. Most of these users reside in regions most affected by censorship like Russia and Iran[28]. Tor is currently a network of over 8000 volunteer owned and operated servers dedicated to providing low latency and high anonymity to its users.

A typical lookup for a resource on the internet is done generally by an unambiguous connection between two entities; the user looking up the resource and the end server which holds this resource. The user requests for a resource and the end server returns this resource through a direct channel established between the two. This connection is prone to interception or passive analysis by a third-party actor. This actor can, at most, see the content that is being requested or at least, link the two entities involved in the exchange. Tor provides anonymity by creating unlinkability between the two communicating parties.

Author's address: Hammas Bin Tanveer, The University of Iowa.

2.1 How Tor Works

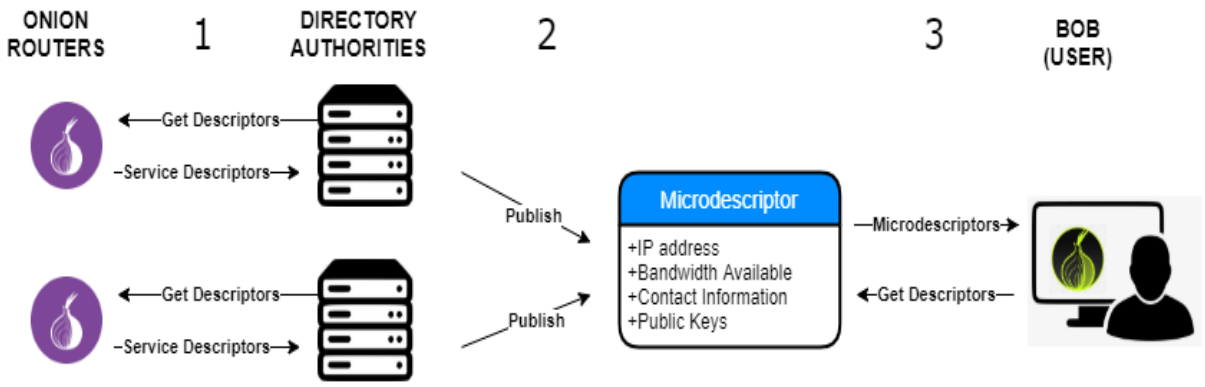

Fig. 1. A basic overview of a Tor client setup. Each *Onion relay* posts its service descriptors which are then gathered bu the *Directory Authorities* as shown in step 1. The *Directory Authorities* then publish the minimal rendition of these service descriptors called *Microdescriptors*. These are then fetched by user's Tor client to build a *Tor circuit*.

When a Tor user wants to fetch a certain resource from the internet, the Tor client does not establish a direct connection with the end server. It instead establishes a *Tor circuit*. A *Tor circuit* is a series of 3 or more servers that route traffic between the user and the end server. First, the Tor client establishes an *Onion proxy* on the user's machine which is then used for all subsequent connections with the Tor network. This *Onion proxy* then contacts the *directory authorities* to get a list of all the currently active volunteer servers running the Tor proxy; these servers are called *Onion routers*. The *directory authorities* are a set of centralized servers which hold information about individual *onion routers*; this information is called the *service descriptor*. The *service descriptor* is published by each *onion router* itself after a specified amount of time and it holds information like the IP address and contact information of the *onion router*, the allowed outgoing ports, its public keys etc. *Directory authorities* hold the most minimal rendition of these *service descriptors*, called *Microdescriptors* which are downloaded by every Tor client every time it starts. This start up procedure of a Tor client is illustrated in Figure 1.

Now that the *Onion proxy* is aware of all the *Onion routers* that are up and working, it chooses, at minimum, 3 *onion routers* to construct the *Tor circuit*. The *Tor Circuit* is a strategically chosen combination of 3 *onion routers* namely, the entry relay (or *guard node*), the middle relay and the exit relay. Strategically choosing the the 3 *onion routers* for a Tor circuit consists of considerations such as not choosing the same *onion router* for the same circuit, not choosing invalid routers and not choosing two routers in the same /16 subnet. The circuit selection also takes into account the outgoing ports allowed by a certain router and the available bandwidth of all the *onion routers* involved in the Tor circuit.

After choosing the 3 *onion routers*, the *Onion proxy* initiates by establishing a connection with the *Guard node* or the entry relay. The *Tor circuit* is then extended incrementally; from the entry relay to the middle relay and then from the middle relay to the exit relay. During this incremental building of

the Tor circuit, a symmetric key is established between each of the *Onion routers* and the user's *Onion proxy*; this is indicated in **Figure 2** by different colored keys with the *relays*.

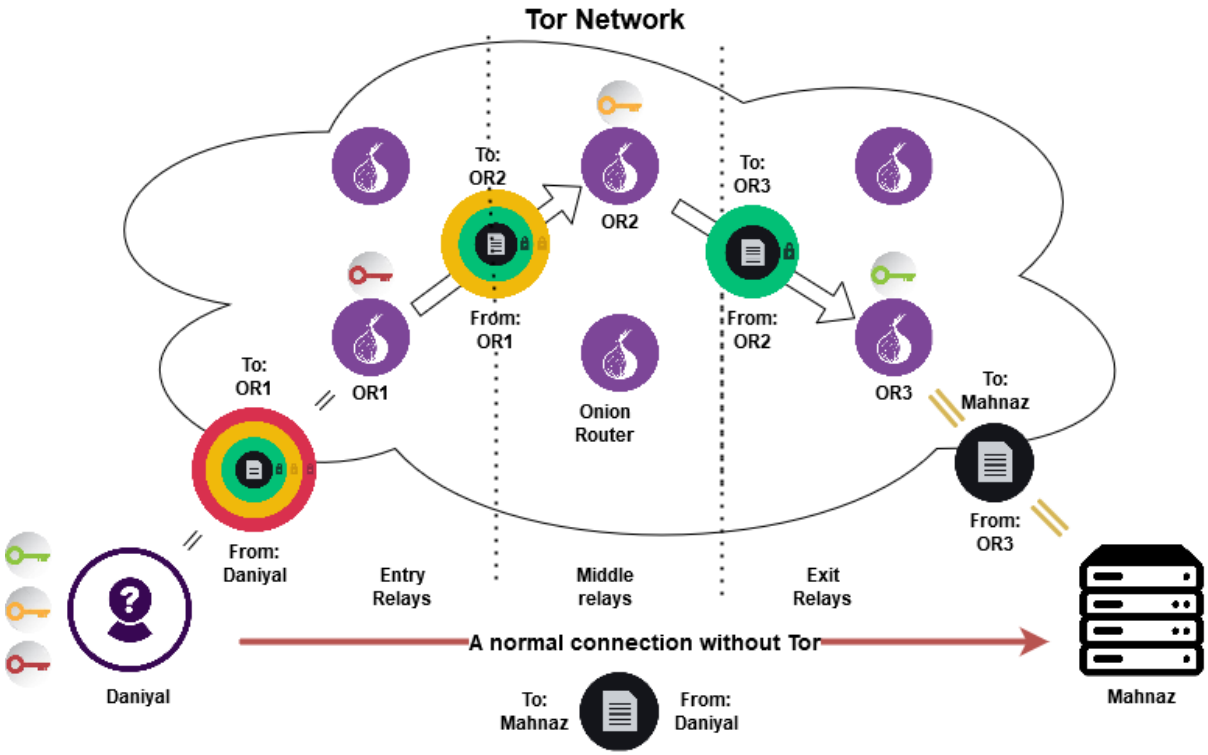

Fig. 2. A basic overview of a comparison between a Tor connection and a regular connection without Tor. In a normal connection, request is simply sent to the end server (Mahnaz) that the user (Daniyal) is trying to connect to. However, this request goes through a series of 3 relays in a Tor connection. Each relay has a symmetric key established with the *Onion Proxy* running on the user's Tor client. These are illustrated by the red, yellow and green keys. The data is encrypted using all 3 of these keys by the *Onion Proxy* running on user's Tor client. The traffic is first encrypted using the green key, followed by the yellow and then the red key. When the data gets to OR1, it can decrypt the outer most layer using the red key it possesses. This leaves the data with only the green and yellow layers which are then subsequently removed by OR2 and OR3 using the yellow and green keys respectively. OR3 now has the content that was originally sent which it then forwards to the appropriate end server (Mahnaz).

After the circuit construction is complete, all the traffic between the user and the end server flows through this Tor circuit. However, a user connecting to a website may require requesting several resources from many different end servers and this may require using more than one circuit. A user's *Onion proxy* will make many Tor circuits preemptively to avoid any delays if one of the circuits

becomes dysfunctional, or if one of the circuits haven been used for more than 60 seconds.

Now the user traffic is ready to flow through these *Tor circuits* to the concerned end servers. Considering a Tor circuit consisting of three relays, all traffic originating from the user is encrypted thrice using the keys established during circuit creation. The traffic is encrypted successively and in order by keys established with the last, middle and the first relay respectively. This creates 3 "layers" over the data itself as shown in the data packet encapsulated by green, yellow and red layers in **Figure 2**. These "layers" are then "peeled off" (decrypted), one by one, by each relay in the circuit. Each relay can only "peel" or decrypt the layer that was created using the key it possesses. This ensures that every relay only has information about where the data is coming from and where it is going to making sure none of the relays have a complete picture of the *Tor circuit*. The entry node knows only the user and the middle relay so it does not know where the request is headed to. The middle relay knows only about the entry relay and the exit relay so it does not know where the request is coming from or headed to. The exit relay only knows the middle relay and the end server so it does not know who asked for the resource it is supposed to fetch.

It should be noted that as indicated by the wide arrows in **Figure 2**, the connections between the entry (OR1) and the middle relay (OR2) and the connections between the middle (OR3) and the exit relay (OR3) remain inside the Tor network itself. However, connections indicated by the thin arrows represent that the connection has one of the actors, sender or receiver, outside the Tor network. Traffic between *Onion proxies* and *Onion routers* and between two *Onion routers* is TLS encrypted. This means that only the connection between the exit relay and the end server might not be TLS encrypted depending on whether the end server supports TLS or not; this is indicated by the yellow line in **Figure 2**.

This very design of a Tor connection allows a user of the Tor network to attain unlinkability with the end server and achieve total encryption. Although this design of the Tor network allows the user to remain anonymous while accessing the internet, it has its drawbacks when compared to a regular internet connection.

2.2 The problems with Tor

As Tor aims at providing high anonymity to its users as compared to a user of the regular internet, a Tor connection differs substantially when compared to a regular internet connection. A user who wishes to use the Tor network can only do so by first configuring the Tor proxy. Once the Tor proxy is set up on the user's machine, it can only use the available, volunteer run, intermediate relays to route its traffic to the concerned end servers. The end servers only communicate with the exit relay of the Tor circuit. These exit relays ask for the resource that the user requested and it is returned to the exit relay by the end server. The exit relay then sends the data back to the user who originally asked for it. In a normal connection, however, a user does not need to install extra software and there is a direct connection between the user and the end server. A direct connection involves no extra relays to route traffic and the content is returned directly to the user.

Although these differences are vital when it comes to providing anonymity to Tor users, they can be the root of a lot of problems.

- Installing a new software and making sure its running correctly can be a complicated process for a typical internet user.
- The reliance on a limited number of volunteer run relays to route traffic through the Tor network can lead to several problems. First, the limited number of these relays means that the bandwidth available to route user traffic is bounded. Second, routing traffic through extra intermediate relays as compared to a regular internet connection means that the Tor network will be slower in reaching the end server.
- The very design of the Tor network makes it a prey to fate sharing. As established earlier, every user's traffic exits the network from the exit relay in the Tor circuit. This means that the end server is only in contact with the exit relay of the circuit and regards all the traffic as coming from the exit relay itself. As the number of Tor users is much higher than the number of the relays available, many people are bound to use the same relay as their exit relay. This means that if a subset of people using the same exit relay sends malicious traffic to the end server, the end server might end up blocking any incoming traffic from the exit relay. This results in differential treatment for Tor users as compared to a typical internet user.

In the next section, I will describe in detail how and to what extent these problems harm the efficiency and efficacy of the network. I will also describe techniques that have been put in place to curb these shortcomings and to what extent they have been successful.

3 THE UTILITY OF THE TOR NETWORK

A typical Tor connection consists of three steps; the user accessing the Tor network, the traffic being routed through the Tor network and finally the traffic reaching the end server through the exit relay. As these steps are not characteristic of a regular internet connection, they might cause problems that a user of a regular internet connection might not face. This leads us to define the utility of the Tor network in terms of the experience of a user using a regular internet connection. The utility of the Tor network is defined as follows:

- The number of people who can successfully access the Tor network as compared to the number of people trying to access the Tor network
- The extra latency and bandwidth challenges that Tor users experience as compared to an individual trying to access web resources over the regular internet
- The number of web resources on the regular internet that a Tor user can access

3.1 The number of people who can actually access the Tor network as compared to the number of people trying to access the Tor network

Obtaining anonymity through the Tor network requires installation of the Tor software. The Tor software sets up an *Onion proxy* on the user's machine which is then subsequently used for further communications with the Tor network as shown earlier in **Figure 2**. There are complications with both the parts of this setup.

- Any individual trying to install the Tor software can come across complications while navigating through the installation process. Even after successfully completing the installation process, the individual might have problems with using the software or making sure it is working correctly. We call these the usability challenges.
- An individual in a certain region with internet censorship can have his or her access to the Tor network blocked by certain authorities. This forces the individual to use some sort of circumvention techniques to access the Tor network. We call these censorship challenges.

3.1.1 ***Usability Challenges.***

Using the Tor network requires the installation of the Tor proxy on a user's machine. This can be achieved in two ways. One is to install the Tor proxy and route user's browser traffic through it. The other is to use the Tor browser which comes preconfigured with the Tor proxy [29].

The earliest version of Tor was in the form of a proxy which had to be configured with a regular browser. This involved configuring Tor with third party software for complete anonymity and ease of use. Clark et al. did a cognitive walkthrough of "installing, configuring and running Tor" [4]. Cognitive walkthrough is a technique which analyzes the usability of a software based on how a typical user interacts with a piece of new software. The premise behind this approach is that the users learn how to use a piece of software by interacting with the software interface and then relying on the instructions provided by the software to complete a certain task. They defined "core tasks" than anyone trying to use the Tor proxy must be able to complete. The core tasks consisted of installing, configuring and disabling Tor and its components. Furthermore, the user must also be able to confirm that their traffic is being anonymized.

They evaluate these tasks against a set of usability guidelines. The guidelines include criteria which focus on user's ability to understand how to operate and engage with the software. The guidelines also evaluate the terminology used in the directions given to a user when setting up Tor or coming across errors related to the setup.

They consider four different setups of using Tor. The first one used the manual setup which consisted of using Tor with Vidalia (Graphical interface for Tor) and Privoxy (Third party proxy software). Two of the remaining three approaches used third party Firefox browser extensions to set up Tor on the user's machine. The last one used a stand alone browser based on Firefox browser which came preconfigured with Tor called Torpark.

They found that all of these different setups had problems which violated their guidelines. Installing Tor often used complicated jargon and vague directions that a typical user might not be able to understand and follow. Although configuring Tor and verifying if it was working was easier using the extensions, the user still had to do all the steps included in setting up Tor the manual way. Torpark eliminated the complications of setting up Tor manually but verifying whether Tor was working still remained a problem.
After the initial release of the Tor software, the Tor project introduced the Tor browser bundle. It was the Tor project's attempt to reach a broader user base by giving the users an option to use a piece of software they are already familiar with; a browser. Norcie et al. conducted a usability analysis of the

Tor browser bundle [17]. Their main premise behind the study was that the "users of privacy enhancing technologies need to be aware of the security tasks they need to perform. They must then be able to perform these task(s) without making any dangerous errors, and then be comfortable enough with the interface to continue using it." They tried to analyze stop points in setting up the Tor browser and using it to download a new desktop background for their machine. They define stop points as "places in an interface where a user may find themselves facing a requirement for user action such that the user is unable to proceed."

They find that even though downloading the Tor browser bundle is a simple process, it still has a myriad of stop points that can drive users away. They conducted the experiment using 25 college students who were considered to be more tech savvy than a typical internet user. They found that long launch times of the browser coupled with a slower browsing experience as compared to a regular browser were the most standout challenges. They also found that users had problems with differentiating between a browser window which used Tor and one which did not. Some security features like redirecting users from Google captcha forms to another search engine like Duck-DuckGo caused further confusions.
The current Tor browser comes with a graphical user interface called the Tor launcher which Tor users can use to set up additional traffic routing before connecting to the Tor network itself. Tor launcher is useful specially for Tor users in regions with internet censorship. As Tor constantly publishes the consensus data publicly, blocking the Tor network can be as trivial as fetching this document and blocking IP addresses of Tor relays listed in it. Tor launcher helps users set up additional proxies or Tor bridges (Tor relays not listed on the public consensus data) to circumvent censorship.

Lee et al. performed a cognitive walkthrough coupled with a quantitative and qualitative analysis of the Tor launcher [14]. They find that Tor launcher uses a lot of technical terms in its instructions. It also leaves a user with "too many options" to choose from providing poor feedback if the user configures something wrongly. 50% of the users were not able to configure Tor bridges and the ones who successfully did had a median time of 40 minutes. They conclude that these problems might drive away potential users and make Tor harder to use in censored regions.

These studies conclude that a typical user faces a range of usability challenges from complicated jargon to a difficult setup with unclear instructions. However, the usage of Tor has since evolved. Tor now comes preconfigured in the Tor Browser [29] which is as simple to download as a regular browser. After years of design changes to the browser that first launched in 2009, the current Tor browser's experience might be very different from what these studies have concluded.

To this end, I will now discuss the findings of **"Peeling the Onion's User Experience Layer: Examining Naturalistic Use of the Tor Browser"** by Gallagher et al. [10] in detail. This is the latest study on the usability of the Tor browser which focuses on the naturalistic use of the browser instead of one in a lab setting for experiments.

To get as complete a view as they can about the usability of the Tor browser, Gallagher et al. collected both, qualitative and quantitative responses. These responses were collected during and after a 7 day experiment which comprised of users using the Tor browser as their first choice for everyday

tasks. The participants were students at a university and were instructed to use the browser for tasks ranging from web browsing to completing online assignments. The participants were, however, free to switch to another browser if they wished. The data collection consisted of filling out short questionnaires, open-ended self reports and participating in one-on-one interviews. The questionnaires were programmed to pop up when the users switched browsers between the Tor browser and any other browser. It contained questions related to specific problems that caused the shift and the kind of content they were surfing. The self-reports and the interviews took place after the completion of the study and were focused on problems the users faced in addition to the ones listed in the questionnaires.

After aggregating the results from the questionnaires, self-reports and the interviews, the authors divided the problems into eight high-level categories as seen in **Figure 3**. Almost half of the questionnaires reported one of the several functionality challenges the users faced. They ranged from failure to load important parts of the website to an inability to stream content. User's also faced challenges related to e-commerce and online news websites.

Issue	Category Labels	Description	Reports	Participants	Total Reports	Total Participants
Broken Functionality	Broken Web Site	Some part of the Web site did not work.	13	6	54	9
	Unresponsive Web Site	The Web site did not load.	13	7		
	Streaming Content	Video streaming did not work.	9	3		
	Reduced Productivity	A productivity-oriented feature could not be used.	9	3		
	Login	Logging into the Web site failed.	2	1		
	Browser Update Required	Accessing the content required a different browser version.	2	1		
	Browser Dependent Content	Accessing the content required a specific browser.	2	1		
	Shopping	A financial transaction could not be completed.	1	1		
	Specific File Types	A specific file type could not be viewed.	3	1		
Latency	Latency	Access was slow.	41	8	41	8
Inconvenience	Inconvenience	A feature present in other Web browsers was missing.	2	2	2	2
Differential Treatment	Tor Traffic Block	The Web site blocked connections from the Tor network.	2	2	5	2
	CAPTCHAs	The Web site wanted to verify that the access was by a human.	3	1		
Geolocation	Geolocation	The Web site was customized to the locale of the Tor circuit's exit node.	2	2	2	2
Crash	Browser Crash	The Tor Browser crashed.	3	3	3	3
Other	Other	The participant provided no information or reported a non-UX problem.	13	5	14	5
	No Perceived Need for the Tor Browser	The participant saw no reason to use the Tor Browser for the task at hand.	1	1		

Fig. 3. Usability challenges faced by Tor users in numbers

The other major problem participants faced was that of latency. As we saw earlier, Tor traffic is routed through at least 3 extra relays between the user and the end server. This extra traffic routing inherently causes the page loads to be much slower than that of a regular browser resulting in loss of productivity for users. User's also faced inconvenience as the Tor browser does not allow saving passwords or adding bookmarks. This can be problematic for everyday users who visit the same set of websites everyday.

Over the course of 7 days, users realized that they were facing differential treatment while visiting certain websites through the Tor browser. They reported complete denial of access to some websites while others required filling CAPTCHAs to gain access. They were also denied access to certain websites due to geolocation challenges. As the Traffic leaves the Tor network through the exit relay, the end server treats the user as if he/she is connecting from where the exit relay is situated. This resulted in some participants not being able to access important websites or the website being loaded

in a different language.

Furthermore, the operational messaging of the Tor browser confused users. The browser would use phrases like "proxy server is refusing connections" and "loading relay information" during web browsing or browser setup. According to users, this information was not useful as it provided little to no direction in terms of resolving the issue. Tor browser's default search engine is Duck-DuckGo. Some participating found Duck-DuckGo's search results less relevant than Google's. They also had difficulties searching the web because Duck-DuckGo does not provide an autocomplete functionality.

As previous studies have concluded, configuring and setting up Tor has been a persistent issue. The use of Tor browser in a lab setting or a naturalistic environment has resulted in various broken functionality and latency challenges. These challenges can lead to "stop points" in setting up Tor not allowing individuals to use the Tor network at all. Even if an individual sets up Tor, latency challenges and differential treatment might cause them to shift to a regular browser. These problems can hinder people from accessing the Tor network. Another problem which hinders people from accessing the Tor network is censorship.

3.1.2 ***Censorship challenges***.

The Tor network is reliant on a number of publicly listed directories for circuit creation. After fetching this information, Tor users connect to the IP address and ports of some of these publicly listed relays. This allows the Tor network to be censored in a lot of different ways. There have been instances of the Tor project's website being blocked [26] in some countries. Others have blocked access to the public Tor network [21]. This can be done in two trivial ways. First, the IP address and port combinations of all the relays present in the publicly listed consensus data can be blocked. Second, the IP addresses of the directory authorities can be blocked. This would block Tor user's access to the Directory authorities who provide the user's with the consensus data. These trivial approaches make blocking Tor as simple as fetching the publicly listed IPs and dropping any traffic aimed for them. This has resulted in widespread censorship of Tor in 9 countries [20].

Despite the widespread censorship of the Tor network, there are ways users in censored regions can circumvent censorship and still access Tor. The two main ways to circumvent censorship are using Tor bridges or pluggable transports.

Tor bridges are *Onion routers* that are not listed on the publicly available consensus data. They act as the entry relay, instead of the *Guard Node*, for a user who needs a bridge to connect to the Tor network. Birdges can be either default or private. Default bridges are usually managed by Tor. They are either given out by default in the Tor browser or they can be asked for by sending an email to the Tor project. Tor only lets out a certain limited amount of bridges to a single user. Private bridges, however, can only be accessed by word of mouth. They are usually bridges that are run by individuals and are passed on to their peers.

Default bridges can be discovered in many different ways [23]. The adversary can take up many different identities at the same time and ask for bridges constantly. Considering a state level adversary

like China, the amount of identities it can create is substantially large than the number of bridges that Tor manages. This will cause all the bridges to be known by the adversary and hence be blocked.

Private bridges, however, are much more decentralized. They are often distributed by word of mouth so they are much harder to find by an organized manner. Dunna et al. analyzed how unpublished (private) bridges were discovered in China [7]. To carry out the experiment, they set up a virtual machine in China and three privately owned bridges in the US, Canada and UK. Then they tried connecting to their private bridges through this virtual machine. They realized that as soon as they connected to the private bridge, the connection would be blocked the bridges would be put on a blacklist of 12 hours. If the machines which were hosted to run private bridges run other services and communicate with the virtual machine in China, it would not be blocked. This finding is consistent with previous studies which conclude that China uses deep packet inspection to detect Tor traffic [8]. China employs the "Great Firewall" which monitors every connection that is made out of the country. The packets in these connections are then undergone Deep packet Inspection (DPI) to reveal certain information about the connections. Saputra et al. found that a Tor connection has certain characteristics which differentiate it from a regular internet connection [30]. As all known Tor relays and bridges are already blocked by China, any other Tor connections detected by DPI is considered to be one with a private bridge. The IP of this private bridge is then blacklisted.

Pluggable transports help Tor traffic to not stand out. Pluggable transports are protocol-level obfuscation plugins for Tor. They disguise Tor traffic as traffic of other popular protocols like Skype or obfuscate the Tor traffic itself. This complicates the jobs of the censors who conduct DPI as Tor traffic passing through pluggable transports looks like traffic of another common protocol. Furthermore, blocking traffic of a protocol as common as SKype or HTTP is not feasible due to their popularity.

Some of the earlier well known pluggable transports inlude Skypemorph [15] and StegoTorus [33]. These pluggable transports perform traffic morphing on Tor traffic which makes it look more like traffic from another protocol. Skypemorph uses the Skype protocol to hide user traffic and Stegotorus uses HTTP. They hide traffic within a connection between a client and a server communicating through this protocol. They make sure that the flow level statistics like packet size's and inter packet times are also mimicked for the given protocol. These pluggable transports were dismissed by Houmansadr et al. who found that a flow level analysis of these protocols allowed easy detection of a pluggable transport [12]. This was owed to the "glaring discrepancies between their crude imitations and the behavior of genuine protocol implementations."

There are other pluggable transports that are more commonly used and are fully implemented unlike the ones above. Tor also maintains pluggable transports itself like obfs3 [22]. I will now discuss how the implemented pluggable transports fare against state level adversaries. To this end, I will discuss **An Analysis of Tor Pluggable Transports Under Adversarial Conditions** by Shahbar et al. for an in depth analysis.

This paper analyzes five different pluggable transport under adversarial conditions namely Obfs3, FTE, Scramblesuit, Meek and Flashproxy. Shahbar et al. consider an adversary which can use flow level

analysis on all the traffic passing through the censor. Flow level characteristics are different from what a DPI can analyze. A state level DPI like the Great Firewall of China can analyze protocol level details that are an attribute of the Tor protocol e.g the set of Encryption techniques the Tor client or server supports. However, flow level characteristics depend more on how quickly the packets flow in a given connection, how big is an average packet etc.

All the 5 above mentioned pluggable transports use different techniques to hide Tor traffic.

Obfs3 relies on obscuring the traffic between a Tor client and server. We should not that this traffic is already "safe" in that it is already encrypted. The content of this connection cannot be seen. However, the establishment of a connection between the Tor client and Tor server has a few characteristics that stand out. This technique relies on obfuscating that connection establishment by using an asymmetric key. This makes the job of the censor harder in looking for the characteristics that stand out. ScrambleSuit deploys the same technique but with two additions. First, it establishes a shared secret between the Tor client and the Bridge to ensure that the two parties know who they are talking to. Second, it can also obfuscate packet's arrival times and sizes to throw off censors that try to use flow level characteristics to recognize Tor traffic.

FTE stands for Format-transformation encryption. FTE works under the assumption that censored regions use certain regular expressions based off of certain traffic characteristics. This helps them identify which traffic belongs to which protocol and whether it should be blocked or not. FTE encodes Tor traffic in such a way that its regular expressions would look like that of another chose protocol e.g HTTP.

Meek relies on popular Content Delivery Networks or websites like Google.com to route user traffic to the bridge. This technique is called domain fronting. It uses the "Server name identification" field of TLS to make it look like that the packet is headed to a commonly used website like google.com. However, the host header inside the HTTP packet is actually meant for the meek servers. As the HTTP headers are encrypted, the censor cannot decipher who the traffic is actually headed to.

Flashproxy relies on a myriad of proxies run on volunteer websites which the Tor client can use to connect to the Tor network. The Tor client constantly shifts between these proxies to relay its traffic back and forth between a Tor client and a Bridge.

Shahbar et al. collected traffic samples of all 5 of the pluggable transports in a "real network environment". As these pluggable transports use different protocols to hide Tor traffic, they use five protocols as background traffic; HTTP, HTTPS, SSH, BitTorrent and Encypted BitTorrent. After capturing these traffic samples, they divide these into flows. A flow is defined as a five tuple consisting of The source IP address, the destination IP address, the source port, the destination port, and the protocol. After using the 5 tuple to extract flows, they then remove these attributes from the flows themselves. They then use a tool called Tranalyzer to extract 65 flow level features from the flows. These features include but are not limited to "Number of packets been transmitted, Number of packets been received, Minimum packet length, Maximum packet length, Average packet size, and Mean inter-arrival time." All these features can be analyzed by a passive adversary.

They then split the data into 2/3 training set and 1/3 testing set and performed a 10 class classification. They also performed 10 cross validation on this data and measured the F-measures for each of the 5 pluggable transports and five background Traffic protocols. **Figure 4** shows the F-measure scores of all the 10 different protocols in consideration. Results show that flow level techniques were able to detect pluggable transport traffic with a very high accuracy using flow level features. Using either split or 10 fold validation provides very similar results.

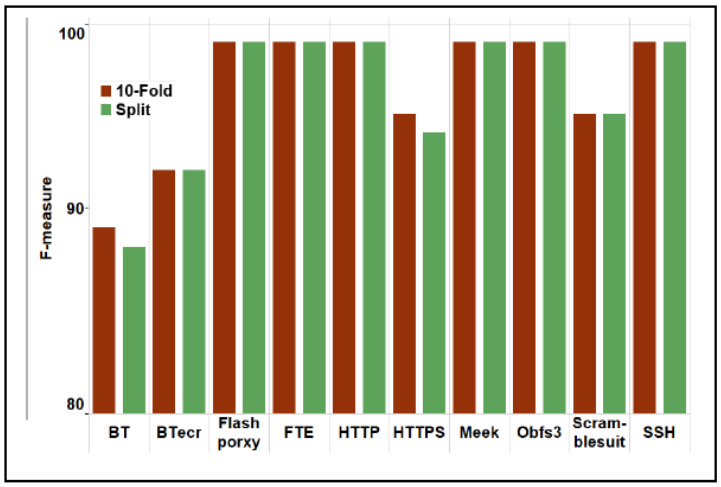

Fig. 4. F-measure scores of all the 10 protocols under analysis using Split and 10-fold validation

Using all 65 features for a state level traffic volume can be infeasible due to number of computations. To this end, Shahbar et al. used a feature ranker to analyze which features contributed to the accuracy of the results the most. They find that the duration of the flow, the number of bytes sent and the maximum packet size were the top 3 features which contributed most to the accuracy. They then used only these 3 features to predict which flow belonged to which protocol. The results are listed in **Figure 5**. It shows that using only a fraction of the total features available can give an F-measure score of over 80% for each of these protocols allowing them to be detected using flow level analysis.

	Class	TP Rate %	FP Rate %	Pre-cision %	Recall %	F-Measure %
Background Traffic	*HTTP*	97.4	0.8	96.9	97.4	97.1
	HTTPS	79.1	0.1	85.2	79.1	82
	SSH	98.7	0	99.9	98.7	99.3
	BT	82	2.7	81.7	82	81.9
	BTecr	86.3	3	88.8	86.3	87.5
Pluggable Transports Traffic	*FTE*	97.7	0.3	97.4	97.7	97.6
	Scramble suit	84.6	0.1	92	84.6	88.1
	Meek	95	0.4	91.7	95	93.3
	Flash proxy	97.8	1.2	95	97.8	96.4
	Obfs3	98.3	0	98.8	98.3	98.6
Overall Correctly Classified Instances	*93%*					

Fig. 5. Different accuracy measures for all protocols using only 3 features

These results conclude that all the pluggable transports under consideration were identifiable with a high accuracy. They argue that this is due to the fact that pluggable tansports like Obfs3 simply obfuscate the protocol but does not change the timing, ordering or sizing of the packets. Other pluggale transports like Flashproxy change an important flow level characteristic, flow duration, but does nothing

to change packet sizes. They suggest that using a pluggable transport that obfuscates traffic and changes flow level characteristics will be better at detection evasion.

3.2 The extra latency and bandwidth challenges that Tor users experince as compared to an individual trying to access web resources over the regular internet using a regular browser

The Tor network achieves anonymity by routing user's traffic through a set of Tor relays. This entails user's traffic being passed through a set of these volunteer run relays which are distributed all over the world. As the relays are volunteer operated, they are limited in number so a lot of Tor users share the same relays for their circuits. This requires extra processing and queuing delays before the packets can be sent out to the next destination in the circuit. As routing traffic between multiple relays is necessary to provide anonymity to Tor users, this overhead cost of latency cannot be avoided.

Fabian et al. studied how the latency affect a regular Tor user's browser experience [9]. They mapped the results from previous usability studies of the internet to the latency that is faced by Tor users. They quote previous studies which establish that user's tolerance for a web page starts to decrease after the first 2 seconds and then falls sharply between 7 to 15 seconds. They then measure the core latency of the top 50 websites. They define core latency as the time it takes for a single HTTP request to go to the end server and return back. Their results show that Tor's median core latency is 7 seconds as compared to a core latency of 1.37 seconds for a regular internet user. They then conducted the same experiment but this time calculated the average latency. Average latency was defined as the time it takes to download a full web page. They report that the average latency for Tor users is 17 seconds whereas it is 3.29 seconds for a regular internet user. They conclude that mapping these load times to previous studies show a median of 88% users abandoning a web page while it loads.

The cost of providing anonymity can be measured and attributed to certain parts and workings of the network. The delay can be caused by a number of reasons ranging from how the Tor client selects its circuits to how the Tor relays handle incoming and outgoing traffic.

Dhunge et al. analyzed which parts of the Tor overlay network contributed most to these delays [6]. They set up a Tor circuit consisting of a Tor client, entry relay, exit relay and an end server as shown in **Figure 6**. They controlled the Tor client and the exit relay to measure the delays experienced by Tor users due to geographic latency and router delay. Geographic latency was defined as the extra time a packet takes to travel around the relays in the circuit. The router delay was defined as the queuing and processing delays in the entry router that was not under their control. They argue that using a 2 hop circuit in which only one relay is not under their control (entry relay) helps them better understand the delays.

They made a single cell of 512 bytes and sent it through the network from the OP to the Web server. They noted time at 4 different times as shown in **Figure 7**; T1 when the packet left the OP (Tor client); T2 when packet reached the Exit router; T3 when the packet left the exit router and T4 when the packet reached the OP again. The Total delay is then calculated using the formula:

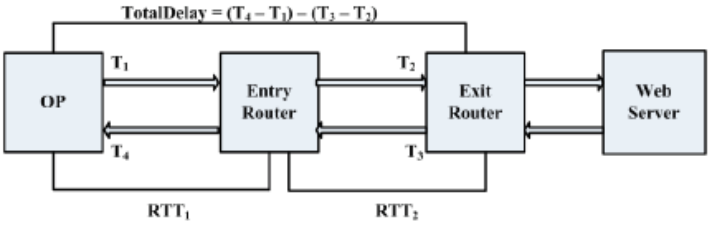

Fig. 6. Experimental setup of Dhunge et al.

TotalDelay = (T4 - T1) - (T3 - T2)

Total delay is the total time the packet spent within the network to get back to OP. They performed this experiment on 1426 routers and found that 23% of the routers experience a delay of more than 1 second. Note that this circuit consists of 2 relays instead of the minimun 3 in an actual circuit so the actual delay is expected to be greater.

The total delay can be due to two reasons. First, the latency between the relays due to the distance. Second, the processing and queuing time the router takes to forward packets called the router delay. To single out the contributions of router delay and latency, they perform a round trip time. They send TCP SYN packets from OP to entry relay (RTT1) and from the exit relay to the entry relay (RTT2). This would be the Total delay due to latency. So the Router delay would be:

RouterDelay = TotalDelay - (RTT1 + RTT2)

They find that circuits with Total delay > 1 second had most of it contributed by router delay. However, the values of router delay fluctuated a lot. They also saw that 7.5% of the circuits faced a RTT latency of 450ms or more. They also find a lot of relays with a router delay 2 seconds. They concluded that this was because of a lot of packets being sent to one relay causing them to wait before reading and forwarding more packets.

Achieving privacy over the Tor network is a trade off between performance and latency. Although Tor can introduce circuit selection algorithms which would reduce geographical latency, they often introduce predictable behaviors which might fall prey to an adversary. For example, Tor can choose circuits by the shortest path algorithm to the end server. This makes identifying the Tor circuits a user might take a much easier task. To this end, we will discuss **NavigaTor: Finding Faster Paths to Anonymity** [2]. In this paper, Annessi et al. introduce a new path selection algorithm for Tor and analyze the extent to which it can reduce Tor's latency challenges.

Annessi et al. first discuss how Tor currently establishes circuits. Relays are weighted by their available bandwidth and then chose randomly to maintain anonymity and performance. No two relays in the same /16 subnet are chose making sure that traffic is routed over geographically distinct regions. The client then preemptively builds ten circuits so the overhead of circuit creation is not incurred during a

client's browsing session. Each relay retains maintains a bucket rate which only allows a set amount of traffic to pass through the relay in a certain amount of time.

This paper talks about three algorithms that can help reduce the latency and increase throughput of the Tor network:

- **Circuit Build time (CBT)**: This client-side method aims at avoiding congested relays. It relies on the fact that a circuit which builds faster than others must be less congested. This algorithm constantly calculates a timeout value by using circuit build times and a priori information about the statistical distribution of these times. If any circuit takes more time than this timeout value, it is discarded from the pool of available circuits. Note that circuit build times include the times it takes to build a complete circuit (including computing encryption keys between client and relays on the circuit).
- **Congestion-Awareness**: This technique is based on isolating the delays caused by congestion from other delays such as those caused by propagation. This is done by computing the round trip times of of circuits five times after building them.
- **Circuit RTT**: This technique is introduced by Annessi et al. Instead of measuring whole circuit build times, it instead computes the round trip time to the exit relay. This involves sending an invalid request to the exit relay after the circuit is established. As the request cannot be completed, the exit relay sends back an error message. This time delay between sending the initial packet and receiving the error message is used as a relay performance indicator. Circuits with less Circuit-RTT are prefered over circuits with more Circuit-RTT.

Two types of circuit characteristics are then measured using the above defined methods:

- **Time to first byte (TTFB)**: This is done by measuring the time elapsed between sending an HTTP HEAD method to google.com and receiving the first byte.
- **Throughput Measurement**: This involves hosting a static web page hosting a file of 5MB on a CDN server. The throughput is designed as the time it takes for circuits to download this file.

To measure these characteristics, they used 19 machines in PlanetLab and built a million circuits from each machine using both CBT and circuuit RTT. They measure median TTFB for the fastest 80% and fastest 50% of the circuits. The results are summarized in **Figure 7**. They find that using their method, median and 90th percentile TTFB is reduced by 11.4% and 19.2% respectively. Using CBT isntead, the median and 90th percentile TTFB is reduced by 8.8% and 11.4% respectively. **Figure 7** shows that this difference is even more pronounced if the fastest 50% of circuits are selected instead.

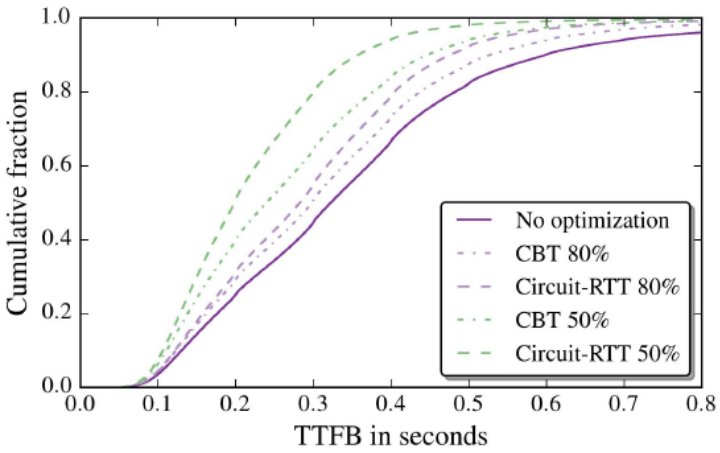

Fig. 7. CDF of TTFB for different methods

They then conducted the same experiment to measure increase in throughput. This time, they established a 100,000 circuits on each of their planet lab machines and measured throughput for the fastest 80% and fastest 50% of the circuits. They find that using their method, median and 90th percentile throughput is increased by 21.1% and 24.9% respectively. Using CBT instead, the median and 90th percentile throughput is increased by 19.1% and 20.0% respectively. The results for throughput increase are much more pronounced if the fastest 50% of the circuits are chosen.
creation.

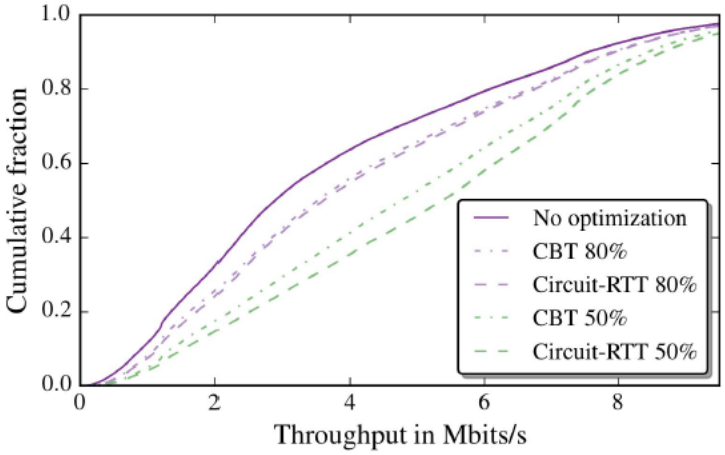

Fig. 8. CDF of throughput for different methods

The experiment was then repeated for the Congestion-Aware method. 100,000 circuits were built and the fastest 80% of the circuits were chosen. These were then chose to measure the median TTFB. Congestion-Aware method improved the median TTFB by almost half the percentage that Circuit-RTT was able to increase it by.

To analyze how secure these algorithms are, they calculated the skew in relay selection using the Gini-Coefficient. "A Gini coefficient G = 0 represents perfectly uniform node selection, and G = 1 implies perfect inequality, i.e., the same node is always chosen". They calculated the Gini-coeffecient of the relays selected while using the 3 different methods of circuit. They find that Gini-coefficient increased by 4.8% for CBT and 7.8% for Circuit-RTT. This result is more pronounced when the fastest 50% of the circuits are chosen instead of 80%. As these methods make relay selection less uniform, it makes it easier to identify the relays being used by a client. This represents that there is a trade-off between anonymity and performance.

These studies show that due to inherent design of the Tor network, it is bound to be slower than the regular internet. Although some parts of then network can be tuned to increase performance, there is often an anonymity cost associated with it.

3.3 The number of web resources that an individual using the Tor network can access as compared to an individual using a regular browser

The Tor network is prey to fate sharing. As we saw in **Section 2**, each user's traffic leaves the Tor network through the exit relay. This makes the end server see all the traffic coming directly from the exit server instead of the user. If the user using this exit relay performs a malicious activity, the end server would associate this activity with the exit relay. This might result in the end server restricting incoming connections from this exit relay. As the number of Tor users is much larger than the number

of available relays, many users end up sharing exit relays. If any one of the users using the same exit relay performs a malicious activity, it might render the exit relay useless for all the users. This is how Tor users "share" fate with other Tor users.

Tor is the source of a lot of malicious activity on the internet. According to cloudflare, 94% of all requests that come from the Tor network are malicious [19]. Furthermore, Sqreen application protection found that a Tor IP is 30 times more likely to launch attack on their customers as compared to a regular IP [3]. Although this malicious traffic might originate from a handful of Tor users, the end servers treat Tor users deferentially as a whole.

Khattak et al. studied the extent of this differential treatment on the transport and application layer [13]. To study the differential treatment on the transport layer, they conducted scans on the whole IPv4 address space. They started off by scanning the whole IPv4 space through a set of control nodes based in the US and the UK. They sent TCP SYN packets to the whole address space for 7 days and gathered IPs which replied with a SYN-ACK. These IPs constituted the RAW footprint of the IPv4 space. They then calculated 2 further footprints:

- **LAX:** Set of IPs which replied to all control nodes atleast once.
- **STRICT:** Set of IPs which replied to all control nodes on all 7 days.

Khattak et al. then use 4 Tor nodes to send TCP SYN packets to these 3 different IP footprints. They consider an IP blocking connections from Tor if the IP address never responds during the seven days. On all 4 nodes, they face on average 15%, 12% and 2% blocking on the RAW, LAX and STRICT web footprints.

To measure application level blocking of Tor, they conducted two experiments.

First, they visited the Alexa Top 1000 URLs from all of Tor exit nodes and their control servers. They sent HTTP requests to these URLs from over 900 exit nodes everyday for 4 days. The pages that responded with a non-200 HTTP response were deemed as ones blocking Tor exit node IPs. They saw this response by 3.67% of the websites. They found that 15.6 websites on average blocked more than 60% of Tor nodes.

Second, they did a longitudinal analysis on data for 2300 distributed URLs which were accessed by Tor and Control nodes at the same time over the span of a year. The data was in form of (Tor,Non-Tor) tuple listing the response codes when these URLs were accessed at the same time. They saw that 6.8% of these URLs only blocked the Tor connection.

Although this study established that Tor users are indeed discriminated against, it lacked certain insights. First, this study's crawler only visited the homepage of the URLs it visited which is not representative of a typical user behavior. Second, it did not highlight whether Tor was blocked due its "reputation" or actual abuse traffic.

To this end, we will now discuss **"Characterizing the Nature and Dynamics of Tor Exit Blocking"** by Singh et al [32]. This paper studies the amount of discrimination an actual Tor user might face on

the internet and whether this is motivated by Tor's reputation or actual abuse traffic.

To analyze how rampant the differential treatment is for Tor users, they designed their own crawler. The crawler not only visits the Alexa top 500 websites, but also instruments any login or search functionality that is present in the website. This functionality allowed them to study differential treatment more realistically in terms of a typical Tor user. Their crawler was based on a full-fledged browser (Firefox) with bot-detection avoidance techniques. This would make their crawler more robust against websites which deploy bot-detection to block detected.

They chose a set of 100 exit relays which support HTTP(S) connection and four of their own exit relays for the crawler. They also had a control connection which used a university network to visit these websites. From both these vantage points, they visited each of these pages and recorded the HAR file, the HTML source and a screenshot of the page. To detect differential treatment, they compare the perceptual hash of the pair of images. If the perceptual hash distance of the screenshot from the control connection and Tor connection was below a certain threshold, they flagged it is discrimination. This can result due either a block page or a CAPTCHA being shown instead of the actual page.

They find that the homepages of 20% of Alexa Top 500 websites discriminate against Tor users. Furthermore, 17.5% of the websites with search functionality and 17% of websites with search functionality also perform differential treatment on their respective homepages. Instrumenting the crawler to use the search and login functionality increases the differential treatment. It rises by 4% for search capable websites and 7.5% for login capable websites.

Figure 9 and **Figure 10** show the prevalence of discrimination and discrimination by site category respectively. They find that relays are at most discriminated by 32.6% of the Alexa Top 500 websites. Furthermore, 50% of the exit relays are are discriminated against by 27% of websites. They also notice that search websites discriminate the least against Tor exits followed by News. Shopping and Social Networking websites follow similar trends and they are the most aggressive at blocking Tor exits with 50% of them blocking over 60% of Tor exits.

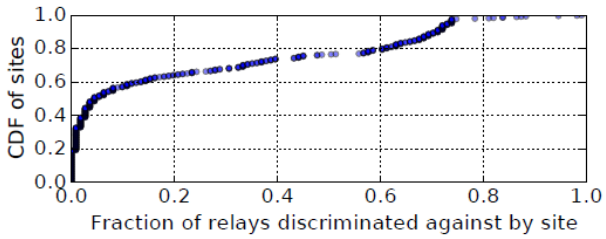

Fig. 9. CDF of discrimination against relays by site

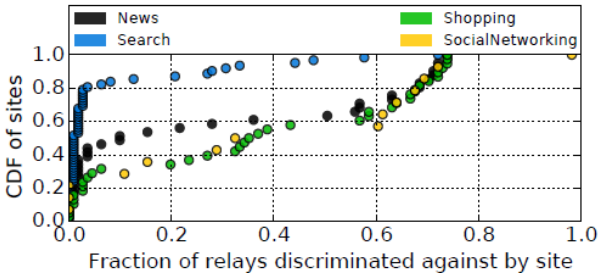

Fig. 10. CDF of discrimination against relays by site category

They also went on to measure the number of users visiting Alexa top 1 million who faced failed TLS handshakes or HTTP requests. To conduct this experiment, the first divide the Alexa top 1 million into different buckets. The first bucket contains the top 100 websites ranked from 1 to 100 and the n^{th} set for $n > 1$ contains the top $100 * 2^{n-2} + 1$ to $100 * 2^{n-1}$. For each of these buckets, they calculated the number of: (1) HTTP requests to the front-page of a website, (2) Error status code in the responses, (3) HTTP(S) handshakes initiated and (4) number of timed-out responses.

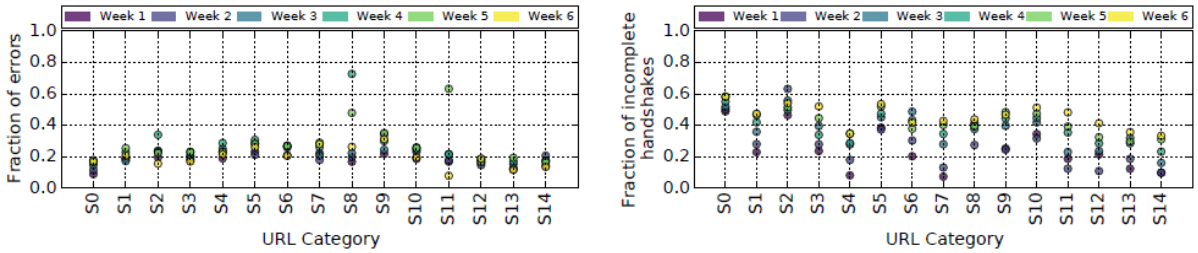

Fig. 11. Percentage of error codes/failed handshakes in all buckets

They observe that over a period of 5 weeks, the percentage of HTTP requests with an error code in response varied between 15% and 33%. The percentage of incomplete HTTP(S) handshakes over the same period varied between 35% and 50%. **Figure 11** shows the amount of HTTP requests that returned an error code in each of the buckets of websites (on the left). It also shows the amount of failed HTTP(S) handshakes in each of these buckets (on the right). The amount of failures in each bucket remain similar indicating that website rank does not affect the discrimination rates of websites.

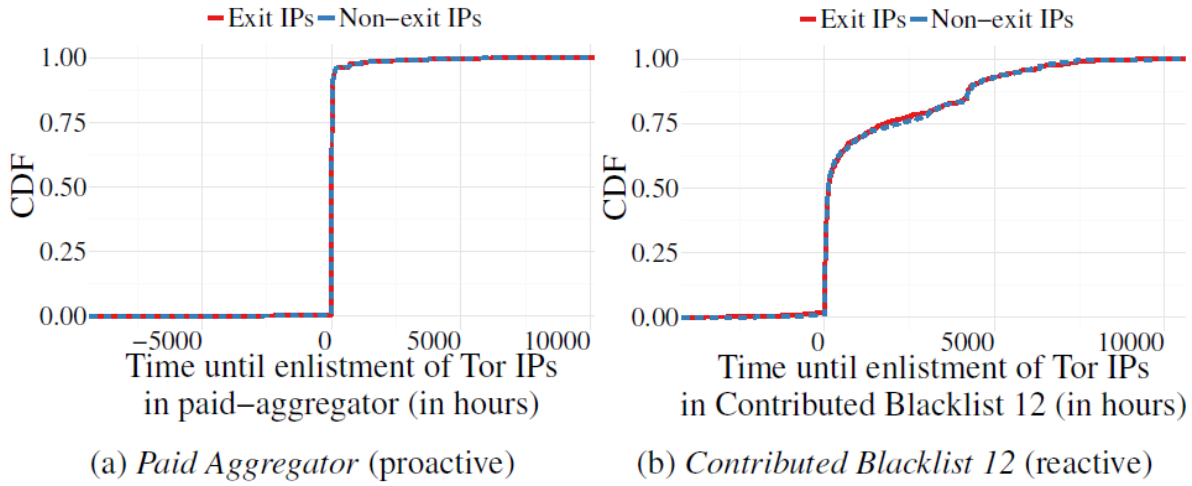

Fig. 12. CDF of how quickly Tor exits were blocked by proactive and reactive lists

They also analyzed whether Tor was blocked proactively or reactively – *i.e* based on historical events/reputation or based on current events. To this end, they analyzed historical data from from 110 different IP blocklists. They deem a blocklist as being proactive if 30% of all Tor exits were included in the blocklist within 24 hours of them going online. **Figure 12** shows the different behavior of the two lists. One of the proactive lists blocked 76% of all Tor exits within 24 hours of them appearing online. Other reactive lists blocked only 0.06% of the Tor exits. In total they identified 2 lists as being totally proactive, 4 as being totally reactive and 84 lists using a combination of the two approaches.

4 THE CURRENT TOR LANDSCAPE

The strength of an anonymity network such as Tor depends on the number of users using the network. This is referred to as the anonymity set of the anonymity network. The larger the anonymity set, the harder it is for the adversary to single out individual users of the network. To garner more users, the Tor project is constantly working on making Tor more accessible and easier to use.

Currently, the single most used way to access the Tor network is through the Tor browser. To make it more usable, the Tor project is open to public for UX/design changes that users think might help improve the experience of using the Tor browser. They have set up several discussion boards and have included several changes suggested by the public into the Tor browser [16]. Furthermore, they are also working towards introducing a new pluggable transport called the SnowFlake [24]. Snowflake works by forwarding internet traffic of Tor users in censored regions through browsers of normal users. It has been introduced in the Tor browser Alpha as an add-on. This makes its use as easy as that of any other add-on/extension of any other browser.

The Tor Project has also worked towards improving its reputation. Recently, Cloudflare, a major content delivery network introduced the Cloudflare onion services [31]. This allows any website hosted on CloudFlare servers to be hosted specifically for Tor users as well. This results in Tor users not being treated deferentially by end servers making their web browsing experience similar to that of someone using a regular internet connection. There has also been ongoing discussion on how to incentivize more volunteers to set up Tor relays [25]. This would provide performance enhancements to the current Tor network making the network faster.

5 CONCLUSION

Tor is the largest anonymity network in the world. While it strives to provide anonymity to users and highly serveilled and censored regions, it faces a lot of challenges. Some of these challenges arrive from the nature of the network itself while others come from third party actors involved – *i.e* censors and end servers. These challenges cause difficulties for users in accessing and using the network thereby degrading the network's utility. However, Tor is making consistent efforts in engaging with the community and making it more accessible in censored regions.

REFERENCES

[1] ACLU. 2018. *The NSA Continues to Violate Americans' Internet Privacy Rights.* https://www.aclu.org/blog/national-security/privacy-and-surveillance/nsa-continues-violate-americans-internet-privacy (Accessed on 09/01/2020).

[2] Robert Annessi and Martin Schmiedecker. 2016. Navigator: Finding faster paths to anonymity. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 214–226.

[3] Christophe Cassa. [n.d.]. *Tor – the good, the bad, and the ugly.* https://blog.sqreen.io/tor-the-good-the-bad-and-the-ugly// (Accessed on 09/01/2020).

[4] Jeremy Clark, Paul C Van Oorschot, and Carlisle Adams. 2007. Usability of anonymous web browsing: an examination of tor interfaces and deployability. In *Proceedings of the 3rd symposium on Usable privacy and security*. 41–51.

[5] CNN. 2018. *Journalist jailed by Iran talks about how Anthony Bourdain changed his life.* https://money.cnn.com/2018/06/10/media/anthony-bourdain-jason-rezaian/index.html (Accessed on 09/01/2020).

[6] Prithula Dhungel, Moritz Steiner, Ivinko Rimac, Volker Hilt, and Keith W Ross. 2010. Waiting for anonymity: Understanding delays in the Tor overlay. In *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*. IEEE, 1–4.

[7] Arun Dunna, Ciarán O'Brien, and Phillipa Gill. 2018. Analyzing China's Blocking of Unpublished Tor Bridges. In *8th {USENIX} Workshop on Free and Open Communications on the Internet ({FOCI} 18)*.

[8] Roya Ensafi, Philipp Winter, Abdullah Mueen, and Jedidiah R Crandall. 2015. Analyzing the Great Firewall of China over space and time. *Proceedings on privacy enhancing technologies* 2015, 1 (2015), 61–76.

[9] Benjamin Fabian, Florian Goertz, Steffen Kunz, Sebastian Müller, and Mathias Nitzsche. 2010. Privately waiting–a usability analysis of the tor anonymity network. In *SIGeBIZ track of the Americas Conference on Information Systems*. Springer, 63–75.

[10] Kevin Gallagher, Sameer Patil, Brendan Dolan-Gavitt, Damon McCoy, and Nasir Memon. 2018. Peeling the Onion's User Experience Layer: Examining Naturalistic Use of the Tor Browser. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 1290–1305.

[11] Guardian. 2018. *The great firewall of China: Xi Jinping's internet shutdown.* https://www.theguardian.com/news/2018/jun/29/the-great-firewall-of-china-xi-jinpings-internet-shutdown (Accessed on 12/04/2019).

[12] Amir Houmansadr, Chad Brubaker, and Vitaly Shmatikov. 2013. The parrot is dead: Observing unobservable network communications. In *2013 IEEE Symposium on Security and Privacy*. IEEE, 65–79.

[13] Sheharbano Khattak, David Fifield, Sadia Afroz, Mobin Javed, Srikanth Sundaresan, Vern Paxson, Steven J Murdoch, and Damon McCoy. 2016. Do you see what I see? differential treatment of anonymous users. Internet Society.

[14] Linda Lee, David Fifield, Nathan Malkin, Ganesh Iyer, Serge Egelman, and David Wagner. 2017. A usability evaluation of tor launcher. *Proceedings on Privacy Enhancing Technologies* 2017, 3 (2017), 90–109.

[15] Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. 2012. Skypemorph: Protocol obfuscation for tor bridges. In *Proceedings of the 2012 ACM conference on Computer and communications security*. 97–108.

[16] ninavizz. [n.d.]. #21183 (Basic Usability Issues) – Tor Bug Tracker & Wiki. https://trac.torproject.org/projects/tor/ticket/21183. (Accessed on 09/01/2020).

[17] Greg Norcie, Kelly Caine, and L Jean Camp. 2012. Eliminating stop-points in the installation and use of anonymity systems: a usability evaluation of the tor browser bundle. In *5th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETS)*. Citeseer.

[18] Jon Penney. 2017. Internet surveillance, regulation, and chilling effects online: A comparative case study. *Regulation, and Chilling Effects Online: A Comparative Case Study (May 27, 2017)* 6, 2 (2017).

[19] Matthew Prince. [n.d.]. The Trouble with Tor. https://blog.cloudflare.com/the-trouble-with-tor/. (Accessed on 09/01/2020).

[20] Tor Project. [n.d.]. *Censorship Wiki.* https://trac.torproject.org/projects/tor/wiki/doc/OONI/censorshipwiki (Accessed on 09/01/2020).

[21] Tor Project. [n.d.]. Picturing Tor censorship in China | Tor Blog. https://blog.torproject.org/picturing-tor-censorship-china. (Accessed on 09/01/2020).

[22] Tor Project. [n.d.]. *pluggable-transports/obfsproxy.* https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/tree/doc/obfs3/obfs3-protocol-spec.txt (Accessed on 09/01/2020).

[23] Tor Project. [n.d.]. *Research problems: Ten ways to discover Tor bridges.* https://blog.torproject.org/research-problems-ten-ways-discover-tor-bridges (Accessed on 09/01/2020).
[24] Tor Project. [n.d.]. Snowflake. https://snowflake.torproject.org/. (Accessed on 09/01/2020).
[25] Tor Project. [n.d.]. Tor incentives research roundup: GoldStar, PAR, BRAIDS, LIRA, TEARS, and TorCoin. https://blog.torproject.org/tor-incentives-research-roundup-goldstar-par-braids-lira-tears-and-torcoin. (Accessed on 09/01/2020).
[26] Tor Project. [n.d.]. Torproject.org Blocked by GFW in China: Sooner or Later? | Tor Blog. https://blog.torproject.org/torprojectorg-blocked-gfw-china-sooner-or-later. (Accessed on 09/01/2020).
[27] Tor Project. 2019. *The Tor Project | Anonymity Online.* https://www.torproject.org/ (Accessed on 09/01/2020).
[28] Tor Project. 2020. *Tor Metrics.* https://metrics.torproject.org/userstats-relay-country.html (Accessed on 09/01/2020).
[29] Tor Project. 2020. *Tor Project | Download.* https://www.torproject.org/download/ (Accessed on 09/01/2020).
[30] Ferry Astika Saputra, Isbat Uzzin Nadhori, and Balighani Fathul Barry. 2016. Detecting and blocking onion router traffic using deep packet inspection. In *2016 International Electronics Symposium (IES).* IEEE, 283–288.
[31] Mahrud Sayrafi. [n.d.]. Introducing the Cloudflare Onion Service. https://blog.cloudflare.com/cloudflare-onion-service/. (Accessed on 09/01/2020).
[32] Rachee Singh, Rishab Nithyanand, Sadia Afroz, Paul Pearce, Michael Carl Tschantz, Phillipa Gill, and Vern Paxson. 2017. Characterizing the nature and dynamics of Tor exit blocking. In *26th {USENIX} Security Symposium ({USENIX} Security 17).* 325–341.
[33] Zachary Weinberg, Jeffrey Wang, Vinod Yegneswaran, Linda Briesemeister, Steven Cheung, Frank Wang, and Dan Boneh. 2012. StegoTorus: a camouflage proxy for the Tor anonymity system. In *Proceedings of the 2012 ACM conference on Computer and communications security.* 109–120.
[34] Jonathan L Zittrain, Robert Faris, Helmi Noman, Justin Clark, Casey Tilton, and Ryan Morrison-Westphal. 2017. The shifting landscape of global internet censorship. *Berkman Klein Center Research Publication* 2017-4 (2017), 17–38.