Recommended Paper

Stronger Bridge Mechanisms of Tor which Take into Consideration Exhaustive Adversarial Models

Fei Feng[1,a) Kanta Matsuura[1,b)

Received: November 27, 2014, Accepted: June 5, 2015

Abstract: Tor is the most popular anonymous communication tool in the world. Its anonymity, however, has not been thoroughly evaluated. For example, it is possible for an adversary to restrict access to the Tor network by blocking all the publicly listed relays. In response, Tor utilizes *bridges*, which are unlisted relays, as alternative entry points. However, the vulnerabilities of the current bridge mechanism have not been thoroughly investigated yet. We first investigate the vulnerabilities of the current bridge mechanism under different adversarial models. Then we compare the current bridge mechanism with our two proposals and discuss their effects on the security and performance of Tor.

Keywords: anonymous communication, Tor, bridge

1. Introduction

In today's expanding online world, there is an increasing concern about the protection of anonymity and privacy in electronic services. The Tor [6] network is the most popular anonymous communication system. It is a low-latency anonymous, private and censorship resistant network whose relays are run by volunteers around the world. Tor is used by private citizens, corporations, and governments to protect their online communications, as well as users trying to circumvent censorship. Its security is therefore essential for the safety and commercial concerns of its users.

While a common use of Tor is to protect privacy, a growing set of Tor users use it as a tool for censorship resistance. Since the destination of a Tor's client is hard to control or determine, Tor can be an effective tool for accessing sites that some regimes may wish to block or censor. However, because the list of all Tor relays are publicly available from directory servers, blocking access to Tor is as simple as downloading the list and blocking connections to the tuples of IP/port it contains.

To counteract this situation, designers of Tor introduced a new method of accessing the Tor network: *bridges* [5], which are designed to help censored users. Bridges are Tor relays that are not listed in the main Tor directory authorities and are alternatives for publicly listed entry relays as entries into the Tor network. Since there is no complete public list of bridges, even if the ISP is filtering connections to all the known Tor relays, they probably will not be able to block all the bridges.

Effective attacks to block or attack the bridge mechanism [12], [27], [28] have been found and conducted in the wild. However, the vulnerabilities of the current bridge mechanism have not been thoroughly investigated yet. This motivates us to construct stronger bridge mechanisms against those attacks. We first thoroughly investigate the vulnerabilities of the current design under exhaustive adversarial models. Then we compare our two proposals with the current design and show their effectiveness through simulations regarding to security and performance of Tor.

Roadmap. We start by introducing some background in Section 2. We then go on to describe our goals and adversarial models in Section 3. Next, we investigate the vulnerabilities of the current bridge mechanism under those adversarial models in Section 4. After that, we propose two alternative methods, and compare them with the current bridge mechanism in Section 5. Conclusions and future work are discussed in Section 6.

2. Background

2.1 Overview of Tor

The Tor network, the implementation of the second generation of Onion Routing, aims to prevent users from being linked with their communication partners; i.e., someone monitoring a client should be unable to discover which server he/she is accessing, and the server (or someone monitoring the server) should be unable to discover the identity of the client using Tor to access it.

The Tor network is a TCP overlay network whose infrastructure is run entirely by volunteers. Tor users download and install the Tor client software, which acts as a SOCKS proxy interfacing their client software with the Tor network. The client first connects to one of the directory authorities, which monitor relays' availability and bandwidth capacity, and then periodically generates a list of status for these known Tor relays. From these authorities the client downloads a list, which is called the consensus file. The client then selects three of these relays, and builds

[1] Institute of Industrial Science, The University of Tokyo, Meguro, Tokyo 153–8505, Japan
[a) fengfei@iis.u-tokyo.ac.jp
[b) kanta@iis.u-tokyo.ac.jp

The initial version of this paper was presented at Computer Security Symposium 2014 (CSS 2014) in October 2014. This paper was recommended to be submitted to Journal of Information Processing (JIP) by Program Chair of CSS 2014.

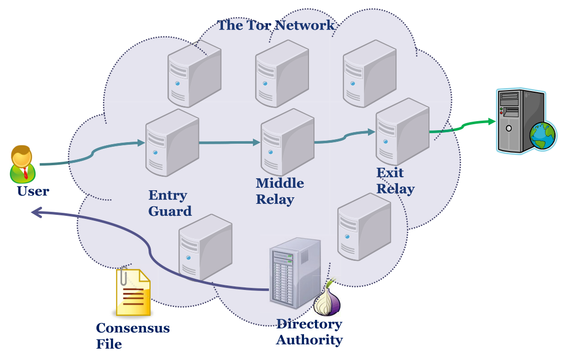


Fig. 1 The structure of the Tor network.

an encrypted channel, which is secured by a session key established through an authenticated Diffie-Hellman key exchange, to the first relay (called the entry guard). Over this encrypted channel, the Tor client builds an encrypted channel to the middle relay, and then via this channel, connects to the third relay (called the exit relay). In this way, the client has a connection to the exit relay, but the exit relay is not aware of whom the entry guard or client is; similarly the entry guard does not know which exit relay the client has selected. **Figure 1** shows the structure of the Tor network.

The client selects an exit relay, then the guard relay, finally the middle relay. There are additional constraints on the path, such as avoiding using more than one relay from a given /16 subnet or the same relay family. Details are available in the Tor Path Specification [24].

2.2 Relays

The relays in a Tor path, and their capabilities and limitations, are of fundamental interest.

The entry position is occupied by a type of relay called the entry guard. Entry guards were first proposed by Wright [29]. Only relays considered fast and stable can receive the Guard flag. It is generally considered better to pick a few (by default, three) possibilities for the entry position and repeatedly use them. Every client maintains a list of those possibilities, which is called the guard list. When the Tor client constructs this list, it selects an expiry time for each of the guards in the list from the range of 30–60 days uniformly at random. After that time, the guards will be dropped and repopulated through a process called guard rotation. When paths are constructed by the Tor client, the entry relay to be used is selected uniformly at random from the client's guard list.

Exit relays are special in that they connect directly to a server (e.g., web server) and thus may have exit policies restricting the types of traffic exiting from them. If traffic requires a certain port, the client will have to pick an exit relay which supports the service they require.

According to the Tor Path Specification [24], there are four kinds of relays to select from when forming paths: guard relays, exit relays, guard + exit relays (suitable for either the guard or the exit position), and non-flagged (i.e., middle) relays. It, however, should be noted that all four kinds of relays can occupy the middle position. Allowing such an occurrence seems a wasteful of scarce resources, but it does permit more randomness in path construction. However, Tor does impose a penalty in the form of weights. Depending on the position and scarcity of the types of relays, they will be weighted by an additional factor during path selection.

2.3 Bridges

Tor users can send email and instant messages, surf websites, and post content online without anyone knowing who or where they are. Consequently, it is widely acknowledged as an important tool for freedom of expression and censorship resistance. That is clearly a worry for authoritarian regimes that want to control and limit their citizens' access to the Tor network. As a list of Tor relays is publicly available from directory authorities, it is trivial for an ISP to block all connections to Tor by blocking access to IP addresses of all Tor relays [27], [28].

To help censored users counteract this situation, designers of Tor introduced a new method of accessing the Tor network: *bridges* [5]. Bridges are Tor relays that are not listed in the main Tor directory authorities and are alternatives for publicly listed entry guards as entries into the Tor network. Even if the ISP is filtering connections to all the known Tor relays, they probably will not be able to block all the bridges, because there is no complete public list of bridges.

A bridge can be operated on a server or a personal computer by a Tor user who is willing to help censored users to reach the Tor network. A standard Tor client can be easily configured to operate as a bridge. Bridges can be strictly unlisted (in which case information of this bridge is spread within a group of people by word of mouth), or their descriptors can be distributed online by the Tor Project. The bridge authority keeps track of valid bridges, and the bridge database [22] distributes bridge information through the web and email, which makes it easy for any client to find a few bridges. On the other hand, the distributing mechanism attempts to make it difficult for an attacker to enumerate bridges in a short time, which is realized by restricting the distribution of bridge descriptors to one set per 24-bit IP address prefix in a week.

Censored users can get several bridges by visiting the BridgeDB site [2] or send an email to bridges@bridges.torproject.org with the line "get bridges" in the body of the mail.

3. Goals and Adversarial Models

3.1 Goals

We have two major goals. As suggested above, our first goal is to investigate the current Tor network's and its bridge mechanism's vulnerabilities to several known attacks against bridges.

Our other major goal is to propose stronger alternative bridge mechanisms. We investigate whether adopting our proposing designs helps to mitigate these known attacks. We compare the current Round-robin Method with the two methods we propose through simulation experiments.

3.2 Adversarial Models

Effective attacks to block or attack the bridge mechanism [12], [27], [28] are being found in existing works and also conducted in the wild. Those adversaries may have different purposes and motivations. Some of them try to enumerate bridges and block the usage of Tor, while others may want to profile or locate bridge users. As a result, they also conduct attacks via different means – passively or actively. In order to propose a stronger bridge mechanism, we first exhaustively summarize possible adversarial models. Only with these adversarial models can the vulnerabilities of the current bridge mechanism be thoroughly investigated.

Censorship. We first consider an active adversary with full control of the local network, who is capable of monitoring, injecting, replaying, shaping and dropping packets but only within his network bounds. An example is the Great Firewall as described by Wilde [27]. When a Tor user within the adversary's network bounds establishes a connection to a bridge, deep packet inspection (DPI) boxes identify the Tor protocol. Shortly after the Tor connection is detected, active scanning is initiated. The scanner pretends to be a normal Tor user and tries to establish a Tor connection to the suspected bridges. If it succeeds, the bridge will be blocked. The details of how the Great Firewall blocks are described by Winter and Lindskog [28]. Iran has also used this strategy to block Tor twice [9].

Enumeration of bridges by malicious middle relays. Next, we consider an adversary who runs malicious Tor relays to discover bridges. This attack has been floating around in the wild, and was documented by Ling et al. [12]. Normal clients use entry guards for the first node of their paths to protect them from long-term profiling attacks, but bridge users use their bridge as a replacement for the first node. As a result, if an adversary runs a relay that does not have the Guard flag and rejects to be an exit relay, the only position it will end up is as a middle one. Then, nodes that build paths to connect to this relay are normal relays and bridges. The adversary can easily identify whether the node, which connected to his malicious middle relay, is a bridge or not, by referring to the public consensus files which contains all IP addresses of relays. This attack can also be conducted by operating a relay and actively scanning the port of each client that connects to it, to confirm which ones are running services of the Tor protocol [13], [25].

Malicious bridges. Bridge relays are donated by volunteers who are willing to help censored users. On the other hand, it is hard to trust all of them, because some of them may be operated by an attacker. In this adversarial model, we consider a passive adversary who runs malicious Tor bridges. His goal is to do traffic observation when his malicious bridges are used as the first node into the Tor network. Then the adversary may perform a statistical profiling attack or a fingerprinting attack [8], [16], [17] on the user. However, the concrete procedures of these attacks are out of the scope of this research. What we will investigate is the relationship between the number of malicious bridges and the number of clients compromised.

Bridge set fingerprinting. It has been discussed in the Tor community that the sets of entry guards might be the fingerprint for a user [4], [11]. When a user connects to Tor from multiple locations where the network is monitored by the same adversary (e.g., a malicious network provider), his persistent use of the same set of entry guards uniquely identifies the user and shows the adversary that connections are all coming from the same user. This could also allow malicious exit nodes – in connection with other attacks – to link clients across destinations. This fingerprinting problem is also related to bridges, and it is even worse because there are guard rotations for normal Tor clients, while there is no rotation of bridges.

To investigate how vulnerable the current bridge design is under these adversarial models, and the chance that an adversary blocks or compromises Tor bridge, we conduct simulation experiments with publicly available data provided by the CollecTor [3].

4. Vulnerabilities of the Current Bridge Mechanism

4.1 Experiment Design

We have developed a Tor bridge path simulator for bridge users by using Stem [20], which is a Python controller library for Tor. The simulator is based on information from the Tor Path Specification [24], the Tor Directory Protocol [23], the Tor Bridge Specification [21] and the Tor source code, which accepts the existing bridge descriptors and bridge network statuses as input. Unlike publicly available historical relay descriptors, bridge descriptors are sanitized by the Tor project by removing or replacing all potentially identifying information, since making bridge data available would defeat the purpose of making bridges hard to enumerate for censors. For example, the bridge identity is replaced with its SHA1 value. Details are described in CollecTor [3]. However, the sanitizing does not affect our simulation experiment because we could identify a bridge by its unique SHA1 value in our simulator.

Our simulator implements only Tor's relay selection logic and does not simulate the actual construction of paths, the data transmission, or network effects such as congestion. To simulate Tor's path selection precisely, the simulator generates paths with specified constraints, such as only using relays with the Exit flag for the exit position, and avoiding using more than one relay from a given /16 network. Regarding the exit policies of exit relays, we assume bridge users do web browsing with ports 80 and 443. When simulating a path, our simulator adopts Adjusted Bandwidth-weighted Random Selection, which is the algorithm currently utilized by Tor. Weighting factors are multiplied by the bandwidths of relays, and relays are selected with a probability proportional to those weighted bandwidths.

It is worth mentioning that, Tor clients choose relays according to relays' *measured bandwidth* in consensus file, but the Tor Project evaluates the performance that users experience with the bridges' or relays' *advertised bandwidth* recorded in descriptors. This is why in our simulator, measured bandwidth is utilized when simulating paths, while advertised bandwidth is utilized when evaluating the performance that users experience. What's more, as mentioned in the Tor source code, when weighting bridges, they enforce 20 KB/s as the lower and 100 KB/s as the upper bound of believable bandwidth, because till now there is no way for the Tor Project to verify a bridge's bandwidth.

In our simulations, we only simulate bridge clients instead of normal Tor clients, because this research assumes that all publicly available Tor relays have been blocked by the adversary. Tor users inside the adversary's network boundary can not access the Tor network by using normal entry relays, so they have to use Tor bridges as alternative entries into the Tor network.

Our simulator not only simulates paths but also bridge users' clients. In those clients, there are N configured bridges chosen from all the bridge network statuses of February 2014, which contain information on 14,463 bridges in total. The number N varies under different scenarios. In the following experiments, we use the bridge network status and relay consensus of the Tor network on 28th February 2014 as input, when there were 3,014 bridges online, to simulate the bridge users and adversary on that day. Our simulator then checks among these N bridges configured in a client, how many of them are online on 28th February. This is meant to simulate the behavior of bridge clients more closely to the real Tor network.

All the historical data used in our experiment are downloaded from CollecTor [3]. We run the simulator on a 8-core 3.20 GHz Intel Core i7 machine with 23.5 GB of memory on Ubuntu 12.04 with the 3.2.0 Linux kernel.

4.2 Censorship

We first conduct the simulation of censorship events to investigate how vulnerable the current bridge mechanism is to censorship events. Suppose there is an active adversary with full control of the local network within his network bounds. When a Tor user within the adversary's network bounds establishes a connection to a bridge, deep packet inspection (DPI) boxes identify the Tor protocol. Shortly after the Tor connection is detected, active scanning is initiated. The scanner pretends to be a normal Tor user and tries to establish a Tor connection to the suspected bridges. If the connection is built, the censor can be sure it is a bridge and thus block it.

We assume there are 200 Tor bridge users within the adversary's network bounds. In all of the clients, there are 4 to 12 bridges configured, which are chosen from all the bridge network statuses as of February 2014. The number of bridges configured in each client is different because every bridge user knows a different number of bridges. In our simulations, the number is chosen uniformly at random from 4 to 12. The adversary could block N% of the online bridges used by the 200 users. We run all the experiments for three rounds, and the average values of the results are shown in **Table 1**.

As explained in Section 4.1, not all of the bridges recorded in February's bridge network statuses are online on 28th February. Thus, some clients may only have offline bridges and have no access to the Tor network even if there is no censorship taking place.

As shown by Table 1, 17.9% of the clients have no online bridges. Then the adversary starts the active blocking of clients' online bridges. We assume he can block N% of all clients' online bridges. The results are shown in Table 1, in which 'Increment' shows the increment of the percentage of clients with no online bridges compared to the value before censorship occurring.

After 75% are blocked, only fewer than 35% clients have access to the Tor network. Blocking 75% may seem like a difficult task, but it is possible if the adversary is of the scale of the Great Firewall.

4.3 Enumeration of Bridges

Since bridges are not publicly listed, adversaries who want to block usage of Tor would like to enumerate bridges. Next, we conduct simulations of an adversary who tries to enumerate bridges. This passive adversary runs malicious Tor relays to discover bridges. If the adversary runs a relay that doesn't have the Guard flag and rejects it to be an exit relay, the only position it will end up is as a middle one. Thus nodes that build paths to connect to malicious middle relays are normal relays and bridges, because normal clients use entry guards for the first node of their paths, while bridge users use their bridge as a replacement for the first node. The adversary can easily identify whether these nodes are bridges or not, by referring to the public consensus files which contains all IP address of relays.

We simulate 20,000 Tor bridge clients and 5 Tor paths for every clients. In all the clients, there are 4 to 12 bridges configured (the number of bridges is chosen uniformly at random). The adversary runs X malicious middle relays. By changing the number of malicious relays controlled by an adversary, we investigate the number of bridges enumerated by the adversary. We run the experiments for three rounds, and values shown in **Table 2** are average results.

On 28th February, there were 3,014 bridges online. As shown by Table 2, when the adversary controls 200 malicious relays, over 53% unique bridges running on that day will be enumerated. Because a Tor client changes its path every 10 minutes, our scenario could be considered if the adversary conducts this attack for 50 minutes. If the adversary conducts this attack for a longer period of time, by waiting for bridge clients to change their paths, he will be capable of enumerating more bridges. This attack can be conducted with comparatively low cost because the adversary can rent IPs on Amazon EC2, for example.

4.4 Malicious Bridges

It is hard to trust every bridge, because they are donated by volunteers, and thus some of them may be operated by an attacker. In this adversarial model, we consider a passive adversary who runs malicious Tor bridges. His goal is to observe users' traffic

Table 1 Simulation results of censorship events.

Percentage of Bridges Blocked	Percentage of Clients with No Online Bridge	Increment
0%	17.9%	0.0%
25%	28.3%	56.7%
50%	41.2%	147.9%
75%	65.8%	256.2%

Table 2 Simulation results of the enumeration of bridges by malicious middle relays.

Number of Malicious Middle Relays	Number of Enumerated Bridges	Percentage of Enumerated Bridges
50	559	18.55%
100	814	27.01%
150	1,286	42.67%
200	1,617	53.65%

Table 4 Simulation results of clients with different number of bridges.

Number of Bridges	Number of Clients that have no Online Bridges	Number of Bridge Sets	Percentage of Clients that can Access the Tor Network	Number of Expected Users per Set
3	49,416	14,295	50.6%	7.00
4	39,430	22,371	60.6%	4.47
5	30,936	31,133	69.1%	3.21
6	24,669	39,453	75.3%	2.53
7	19,486	47,780	80.5%	2.09
8	15,367	55,263	84.6%	1.81
9	12,229	61,733	87.8%	1.62
10	9,655	68,103	90.3%	1.47
11	7,683	73,065	92.3%	1.37
12	5,965	77,888	94.0%	1.28

Table 3 Simulation results of clients compromised by malicious bridges.

Number of Malicious Bridges	Number of Compromised Clients	Percentage of Compromised Clients
0	0	0.00%
50	548	2.74%
100	1,112	5.56%
150	1,611	8.06%
200	2,094	10.47%

when his malicious bridges are used as first nodes of Tor paths. Then the adversary may perform statistical profiling attacks or fingerprinting attacks [8], [16], [17] on these users.

We simulate 20,000 Tor bridge clients and 5 Tor paths for every clients. In all of the clients, there are 4 to 12 bridges configured (the number of bridges is chosen uniformly at random). By changing the number of malicious bridges controlled by an adversary, we investigate the number of clients compromised by the adversary. If one or more paths of the five paths is started with a malicious bridge, the client is defined as compromised. The results are shown in **Table 3**. This attack could also be performed by renting IPs on Amazon EC2 with comparatively low cost.

4.5 Bridge Set Fingerprinting

It has been discussed in the Tor community that the sets of entry guards might be the fingerprint for a user [4], [11]. When a user connects to Tor from multiple locations where the network is monitored by the same adversary, his set of entry guards uniquely identifies the user and shows the adversary that connections are all coming from the same user. This fingerprinting problem is also related to bridges, and it is even worse because there are guard rotations for normal Tor clients, while there is no rotation of bridges.

We simulate 20,000 Tor bridge clients and 5 Tor paths for every client. In all of the clients, there are 4 to 12 bridges configured (the number of bridges is chosen uniformly at random). We run the experiments for four rounds, and the average number of distinct bridge sets is 12,633. Thus, the number of expected users per set is 1.58, which means every user's guard set is distinct with high probability.

It is obvious that the more bridges one knows, the less likely his client has no online bridge. On the other hand, the more bridges one knows, the more likely the bridge set is unique to him/her. In the following experiments, we assume every client is configured with the same number of bridges, to investigate the relationship between the number of bridges, the number of unique bridge sets, and the number of clients that have no online bridge. We simulate 100,000 Tor bridge clients for every experiment. The results are shown in **Table 4**. It is clear from these results that there is a trade-off between availability and privacy in terms of the possibility of being fingerprinted. We consider 7 as an optimal number, because when 7 bridges are configured, over 80% clients have access to the Tor network and the number of expected users is 2.09, which ensures most clients will not have a unique bridge set. However, the assumption that every bridge client in the Tor network knows the same number of bridges, is not the real case, because every client knows of a different number of bridges. What we suggest is that, if a bridge user knows over 7 bridges, just configure 7 in the client, and periodically change the set of bridges.

5. Possible Countermeasures

In the previous section, we investigate the vulnerabilities of the current bridge mechanism under four different adversarial models. In this section, we propose two alternative methods of round-robin over all bridges, and investigate whether these methods make the bridge mechanism more robust against the aforementioned attacks compared to the current Round-robin Method.

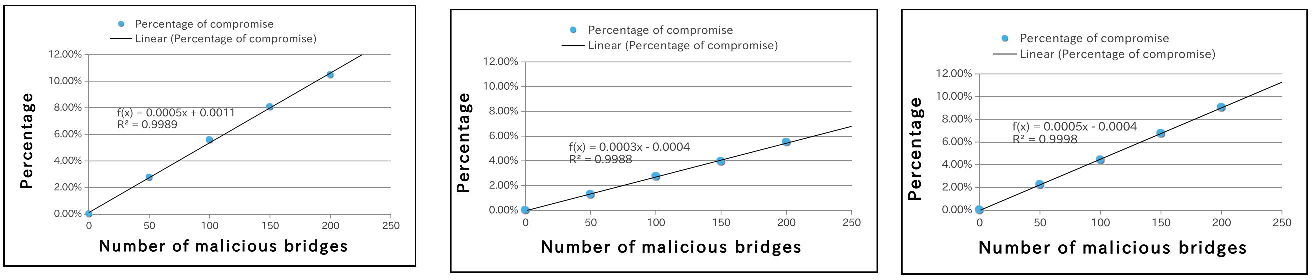
5.1 Alternative Methods

Currently, if ten bridges are configured, the client round-robins over all of them. Instead of the Round-robin Method, we propose two alternative methods.

- **Top One Method:** The first method we propose is to repeatedly utilize the first bridge configured in the client, and only moving to other bridges if the current one becomes offline. The client will try to find usable bridge sequentially from the top of the list, and return to use the ones that precede it in the list, when one of them becomes online again,
- **Top Three Method:** The second method is to randomly choose one of the first three bridges configured in the client, when a path is being constructed. When there are fewer than three bridges configured, the client randomly chooses the first or the first two bridges. This method imitates the current entry guard mechanism.

5.2 Comparison

We conduct simulation experiments to compare the current



(a) Round-robin.

(b) Top one bridge.

(c) Top three bridges.

Fig. 2 Linear approximation of the relationship between the number of malicious bridges and percentage of compromised clients under three different methods.

Table 5 Comparison of three methods under the malicious bridge model.

	Number of Compromised Clients	Percentage of Compromised Clients
Round-robin	1,019	5.10%
Top One	523	2.62%
Top Three	934	4.67%

Round-robin Method with the two alternatives under these adversarial models except the censorship model. It is because our proposals can not mitigate such active attack as the censorship model that identifies Tor protocol, but researches have been done on how to make Tor's traffic undetectable [7], [14], [18], [26].

First, we assume there is an adversary who runs 100 malicious bridges and simulate 20,000 clients and 5 paths for every client respectively to investigate how much clients he can compromise under three different methods. Let's recall that if at least one path is started with a malicious bridge, the client is defined as compromised. **Table 5** shows when the Round-robin Method is used, clients are more likely to be compromised. And the Top One Method is the safest method under this adversarial model. It is easy to understand that repeatedly utilizing the first one can decrease the possibility of a client being exposed to malicious bridges.

Then, we change the number of malicious bridges under the three methods respectively, and find a linear approximation of the relationship between the number of malicious bridges and percentage of compromised clients in **Fig. 2**. Figure 2 shows that results are almost linear. As stated in Section 4.1, clients choose bridges randomly from all bridges in February 2014, and the simulator adopts Tor's current path selection algorithm – Adjusted Bandwidth-weighted Random Selection. Although, it contains randomness, every relay's probability of being selected at a specific position can be estimated. This is why the results are almost linear.

Figure 2 also shows that under the adversarial model of malicious bridges, the Top One Method is the safest method, while the current Round-robin Method is the most vulnerable one.

Next, we compare the three methods under the bridge set fingerprinting model. We simulate 20,000 clients and 5 paths for every client respectively. The results in **Table 6** show that when using the Top One Method, on average 6.66 users share the same bridge set, and hence the adversary can not link a bridge set to a particular user. In contrast, when using the Round-robin Method

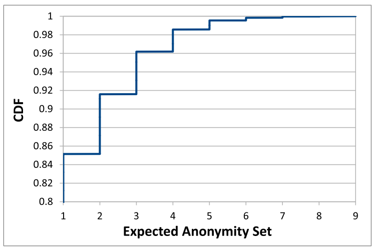
Table 6 Comparison of three methods under bridge set fingerprinting model.

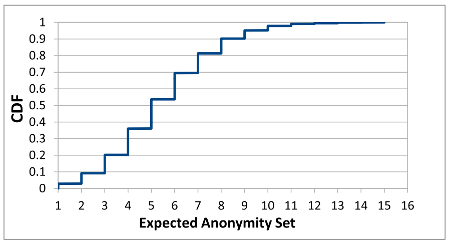
	Number of Distinct Bridge Sets	Number of Expected Users per Set	Number of Clients with a Unique Bridge Set
Round-robin	12,655	1.58	10,775
Top One	3,004	6.66	86
Top Three	12,642	1.58	10,765

or the Top Three Method, over half of the clients have their unique bridge set, which could be the fingerprint of these clients. These results show that repeatedly utilizing the first bridge can significantly mitigate this problem, while using the top three bridges does little help compared to the original Round-robin Method. This is also easy to understand, because when repeatedly utilizing the first one, the bridge set that a client has will be its first bridge. Therefore many clients share the same bridge set. In contrast, in the Round-robin Method, since the bridge set of a client is its bridge list, the probability of different clients having the same bridge set is much lower. When a client has its unique bridge set, it may be fingerprinted by an adversary.

We plot the cumulative distribution functions of the size of the expected anonymity set. This is the number of expected users per bridge set, for the three methods respectively, in order to analyze the probability of being fingerprinted. As **Fig. 3** shows, when there are 20,000 bridge users, repeatedly utilizing the first one can ensure the median user an anonymity set of 6 users. On the other hand, in the Round-robin Method or the Top Three method, over 85% of the bridge sets only has one user and over 53.8% of the clients have their own unique bridge set. As a result, this could allow malicious exit nodes – in connection with other attacks – to link clients across destinations, and malicious network providers to link mobile clients across locations.

Then we compare the three methods under the enumeration of bridges by malicious middle relays model. We assume there is an adversary who runs 100 malicious middle relays, and simulate 20,000 clients and 5 paths for every client for three rounds. **Table 7** shows the number of bridges found when using the default Round-robin Method and our two proposals. It shows that in the three rounds of simulation, sometimes the adversary can enumerate more bridges using our proposals, which means our proposals can not mitigate this attack. We consider the reason why our proposals can not mitigate this attack is that currently


(a) Round-robin.


(b) Top one bridge.

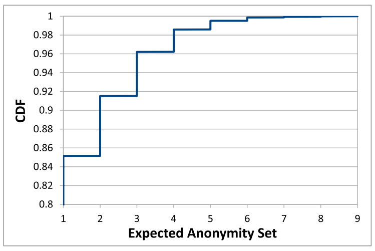

(c) Top three bridges.

Fig. 3 Cumulative distribution of anonymity set size under three different methods.

Table 7 Comparison of three methods under enumeration of bridges by malicious middle relays model.

	Number of Bridges Found (Round 1)	Number of Bridges Found (Round 2)	Number of Bridges Found (Round 3)
Round-robin	826	848	577
Top One	692	965	1,782
Top Three	525	408	1,423

Table 8 Comparison of the performance of three methods.

	Average Bandwidth (B/s) (Round 1)	Average Bandwidth (B/s) (Round 2)	Average Bandwidth (B/s) (Round 3)
Round-robin	50,747	50,781	50,696
Top One	50,251	50,903	50,785
Top Three	49,042	49,109	49,289

we assume that the adversary just insert N malicious non-guard non-exit relays without considering the possibility of being chosen as a middle relay, which causes the dispersion of the results in different rounds.

5.3 Discussions

In the Top One Method, a specific bridge is frequently used by a client. There might be concerns whether there is any disadvantage or negative effect on anonymity or performance. We discuss this in this section.

We first investigate whether our proposals will negatively affect the performance. When a fast bridge is selected, as long as the middle and exit relays are not slow, the client can experience fast service. As mentioned in Section 4.1, the simulator weights bridges by enforcing the lower and upper bound of their bandwidth in the network status file, but it evaluates the performance by their advertised bandwidth from bridge descriptor files. We use the average bandwidth of all online bridges configured in a client as the metric for evaluating the performance of the Round-robin Method, the bandwidth of the first bridge for the Top One Method, and the average bandwidth of the first three bridges for the Top Three Method. We simulate 20,000 clients and 5 paths for every client respectively for three rounds, and the results in **Table 8** are the average value. Table 8 shows that the Top One Method does not negatively affect the performance, while the Top Three Method causes a slight degradation in performance. The reason for this slight degradation is still unknown, and we consider it as a future task.

Other concerns when adopting the Top One Method include that the diversity of paths might decrease, and the load of network traffic might concentrate on some bridges. We simulate 20,000 clients and 5 paths for every client for these three methods respectively.

Table 9 Comparison of the diversity at the bridge position under three methods.

	Entropy at the Bridge Position	Gini Coefficient at the Bridge Position
Round-robin	11.47	0.19
Top One	11.41	0.24
Top Three	11.46	0.20

Table 10 Comparison of how many times each bridge is used under three methods.

	Average	Standard Deviation
Round-robin	27.06	9.30
Top One	27.03	11.88
Top Three	27.20	9.90

We use *Shannon entropy* as the diversity metric. Bauer et al. [1] adopted Shannon entropy as their definition of anonymity and diversity of Tor's path selection. The higher the value is, the more diverse these paths are. Snader and Borisov [19] adopted a different metric, the *Gini coefficient*. It is a measure of inequality, utilized frequently in the field of economics. The Gini coefficient G equals the normalised area between the CDF of the probability distribution being measured, and the uniform CDF. For a population's income inequality, taking y_i ($i = 1$ to n), which is indexed in non-decreasing order, to mean the income of a person, the Gini coefficient is:

$$G = \frac{1}{n}\left(n + 1 - 2\left(\frac{\sum_{i=1}^{n}(n + 1 - i)y_i}{\sum_{i=1}^{n} y_i}\right)\right)$$

A low Gini coefficient indicates a more equal distribution, with 0 corresponding to complete equality, while a higher Gini coefficient indicates more unequal distribution, with 1 corresponding to complete inequality. In Tor's path selection, $G = 0$ represents uniform distribution over all candidate relays and $G = 1$ indicates that the same single relay will always be chosen. **Table 9** shows the Shannon entropy and Gini coefficient at the bridge position of three methods. These values show that using the Top One Method decreases the diversity of paths.

To investigate how many times each bridge is used, we calculate the average and the standard deviation values. The results in **Table 10** show that there is not much difference in the average values, while the standard deviations show using the Top One Method causes some bridges being used more frequently, which means it will cause traffic load concentration.

Through these analyses, we prove that repeatedly utilizing the first bridge will cause a decrease in path diversity and poorer load balancing. However, this problem is easy to solve. Utilizing the Top One Method does not mean that the client will have to use the same bridge forever. Rotation should be introduced if the Top One Method is adopted in Tor. The rotation of bridges can imitate that of the entry guard.

In the current version of the Tor protocol, instead of choosing a new guard every time, every Tor client maintains a guard list, of several (by default, three) pre-selected guards, When the Tor client constructs this list, it selects an expiry time for each of the guards in the list from the range of 30–60 days uniformly at random. After that period of time, the expired guards will be dropped and repopulated, which is called guard rotation. This has been described in Section 2.2.

Guard rotation serves several purposes. First, it is controversial whether it is better to be compromised with some probability all the time or either to be completely safe until the next guard rotation. Øverlier and Syverson [15] proved that the latter is better and hence the guard mechanism was introduced to Tor. As a result, if a client chooses a malicious guard unluckily, guard rotation enables this user to regain some privacy. Second, if clients never rotate their guards, then guards would accumulate more clients the longer they participated in the Tor network, which leads to poor load-balancing. However, recent works [4], [10] have suggested that the current parameters for guard rotation, including the period of rotation time, may exposure users to greater loss in privacy. Dingledine et al. [4] suggests extending the period of guard rotation to 9 months.

Although bridges and guards are similar in that they all serve as the entries into the Tor network, they are not the same. We thus believe that the rotation mechanism of bridges, including the best period of bridge rotation, may not be the same as that of guard rotation without further research. So we regard it as another future task.

6. Conclusions and Future Work

In this research, we first investigate the vulnerabilities of the current bridge mechanism under four adversarial models through simulations using our bridge path simulator. Then we compare it with our two proposals. We discover our proposal of repeatedly utilizing the first bridge (the Top One Method) can effectively mitigate the attacks under the malicious bridge model and the bridge set fingerprinting model. This proposal is easy to implement and proved to not negatively affect performance. So we consider it practical for the Top One Method to be adopted in Tor.

As described in Section 5.3, utilizing the Top One Method does not mean that the client will use the same bridge forever. Rotation should be introduced if the Top One Method is adopted in Tor. The rotation of bridges can imitate that of entry guards, but the details, including the period and how to obtain new bridges, remain as open research questions.

As mentioned in Section 5.2, the two proposals seem not helpful in mitigating the enumeration of bridges by malicious middle relays. We regard countermeasures toward this attack as our future work.

Acknowledgments This work was partly supported by JSPS KAKENHI Grant Number 25280045.

References

[1] Bauer, K., McCoy, D., Grunwald, D., Kohno, T. and Sicker, D.: Low-resource Routing Attacks Against Tor, *Proc. 2007 ACM Workshop on Privacy in Electronic Society*, *WPES '07*, pp.11–20, New York, NY, USA, ACM (2007).

[2] Bridge DB: The Tor Project (online), available from ⟨https://bridges.torproject.org/⟩ (accessed 2014-12).

[3] CollecTor: CollecTor (online), available from ⟨https://collector.torproject.org/index.html⟩ (accessed 2014-12).

[4] Dingledine, R., Hopper, N., Kadianakis, G. and Mathewson, N.: One Fast Guard for Life (or 9 months), *Proc. 7th Workshop on Hot Topics in Privacy Enhancing Technologies*, *HotPETs 2014*, Amsterdam, Netherlands (2014).

[5] Dingledine, R. and Mathewson., N.: Design of a blocking-resistant anonymity system, Tor Project Technical Report (online), available from ⟨https://svn.torproject.org/svn/projects/design-paper/blocking.html⟩ (accessed 2014-12).

[6] Dingledine, R., Mathewson, N. and Syverson, P.: Tor: The Second-generation Onion Router, *Proc. 13th Conference on USENIX Security Symposium*, *SSYM '04*, Vol.13, pp.303–320, Berkeley, CA, USA, USENIX (2004).

[7] Dyer, K.P., Coull, S.E., Ristenpart, T. and Shrimpton, T.: Protocol Misidentification Made Easy with Format-transforming Encryption, *Proc. 2013 ACM SIGSAC Conference on Computer and Communications Security*, *CCS '13*, pp.61–72, New York, NY, USA, ACM (2013).

[8] Hintz, A.: Fingerprinting Websites Using Traffic Analysis, *Proc. 2nd International Conference on Privacy Enhancing Technologies*, *PET '02*, pp.171–178, Springer-Verlag, Berlin, Heidelberg (2002).

[9] Iran blocks Tor; Tor releases same-day fix: The Tor Blog (online), available from ⟨https://blog.torproject.org/blog/iran-blocks-tor-tor-releases-same-day-fix⟩ (accessed 2014-12).

[10] Johnson, A., Wacek, C., Jansen, R., Sherr, M. and Syverson, P.: Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries, *Proc. 2013 ACM SIGSAC Conference on Computer and Communications Security*, *CCS '13*, pp.337–348, New York, NY, USA, ACM (2013).

[11] Kadianakis, G.: Implications of switching to a single guard node: Some conclusions, The Tor Project (online), available from ⟨https://lists.torproject.org/pipermail/tor-dev/2014-March/006458.html⟩ (accessed 2014-12).

[12] Ling, Z., Fu, X., Yu, W., Luo, J. and Yang, M.: Extensive Analysis and Large-Scale Empirical Evaluation of Tor Bridge Discovery, *Proc. 31th IEEE International Conference on Computer Communications*, *INFOCOM '12*, pp.2381–2389, Orlando, FL, USA, IEEE (2012).

[13] McLachlan, J. and Hopper, N.: On the Risks of Serving Whenever You Surf: Vulnerabilities in Tor's Blocking Resistance Design, *Proc. 8th ACM Workshop on Privacy in the Electronic Society*, *WPES '09*, pp.31–40, New York, NY, USA, ACM (2009).

[14] Moghaddam, H.M., Li, B., Derakhshani, M. and Goldberg, I.: SkypeMorph: Protocol Obfuscation for Tor Bridges, *Proc. 2012 ACM Conference on Computer and Communications Security*, *CCS '12*, pp.97–108, New York, NY, USA, ACM (2012).

[15] Øverlier, L. and Syverson, P.: Locating Hidden Servers, *Proc. 2006 IEEE Symposium on Security and Privacy*, *SP '06*, pp.100–114, Washington, DC, USA, IEEE Computer Society (2006).

[16] Panchenko, A., Niessen, L., Zinnen, A. and Engel, T.: Website Fingerprinting in Onion Routing Based Anonymization Networks, *Proc. 10th Annual ACM Workshop on Privacy in the Electronic Society*, *WPES '11*, pp.103–114, New York, NY, USA, ACM (2011).

[17] Shi, Y. and Matsuura, K.: Fingerprinting Attack on the Tor Anonymity System, *Information and Communications Security*, LNCS 5927, pp.425–438, Springer-Verlag, Berlin, Heidelberg (2009).

[18] Smits, R., Jain, D., Pidcock, S., Goldberg, I. and Hengartner, U.: BridgeSPA: Improving Tor Bridges with Single Packet Authorization, *Proc. 10th Annual ACM Workshop on Privacy in the Electronic Society*, *WPES '11*, pp.93–102, New York, NY, USA, ACM (2011).

[19] Snader, R. and Borisov, N.: A Tune-up for Tor: Improving Security and Performance in the Tor Network, *Proc. Network and Distributed Security Symposium*, *NDSS '08*, Internet Society (2008).

[20] Stem: Stem Docs (online), available from ⟨https://stem.torproject.org/index.html⟩ (accessed 2014-12).

[21] Tor Bridge Specification: The Tor Project (online), available from ⟨https://gitweb.torproject.org/torspec.git/tree/attic/bridges-spec.txt⟩ (accessed 2014-12).

[22] Tor bridgedb: The Tor Project (online), available from ⟨https://gitweb.torproject.org/bridgedb.git/tree⟩ (accessed 2014-12).

[23] Tor Directory Protocol: The Tor Project (online), available from ⟨https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt⟩ (accessed 2014-12).
[24] Tor Path Specification: The Tor Project (online), available from ⟨https://gitweb.torproject.org/torspec.git/plain/path-spec.txt⟩ (accessed 2014-12).
[25] Vasserman, E., Jansen, R., Tyra, J., Hopper, N. and Kim, Y.: Membership-concealing Overlay Networks, *Proc. 16th ACM Conference on Computer and Communications Security*, *CCS '09*, pp.390–399, New York, NY, USA, ACM (2009)
[26] Weinberg, Z., Wang, J., Yegneswaran, V., Briesemeister, L., Cheung, S., Wang, F. and Boneh, D.: StegoTorus: A Camouflage Proxy for the Tor Anonymity System, *Proc. 2012 ACM Conference on Computer and Communications Security*, *CCS '12*, pp.109–120, New York, NY, USA, ACM (2012).
[27] Wilde, T.: Great Firewall Tor Probing Circa 09 DEC 2011, GitHub Gist (online), available from ⟨https://gist.github.com/da3c7a9af01d74cd7de7⟩ (accessed 2014-12).
[28] Winter, P. and Lindskog., S.: How the Great Firewall of China is Blocking Tor, *Proc. 2nd USENIX Workshop on Free and Open Communications on the Internet*, *FOCI '12*, Berkeley, CA, USENIX (2012).
[29] Wright, M., Adler, M., Levine, B. and Shields, C.: Defending anonymous communications against passive logging attacks, *Proc. 2003 IEEE Symposium on Security and Privacy*, *SP '03*, pp.28–41, Washington, DC, USA, IEEE Computer Society, (2003).

Editor's Recommendation

The authors evaluate current Tor bridge mechanisms under different adversarial models by developing a Tor bridge path simulator. This paper shows the novel results with the high coverage of adversarial models. In addition, the proposed method can be easily implemented and it is practical. Thus, we expect many readers will be interested.

(Program Chair of Computer Security Symposium 2014,
Daisuke Inoue)

Fei Feng received her M.Eng. degree from the Graduate School of Information Science and Technology at the University of Tokyo, Japan, in 2015. She received her B.Eng. degree, from the School of Information Security Engineering at Shanghai Jiao Tong University, China, in 2011. Her main research interests include anonymous communication and information security.

Kanta Matsuura received his B.Eng., M.Eng., and Ph.D. degrees from the University of Tokyo, Japan, in 1992, 1994, and 1997, respectively. He is currently Professor at Institute of Industrial Science, the University of Tokyo. His research interests include cryptography, network security, and security management. In 2008, he won Distinguished-Service Award from the IEICE Communications Society. He has been serving as a member of Editorial Board of Designs, Codes and Cryptography since 2010. He is a senior member of ACM, IEEE, IEICE, and IPSJ.