修士論文

Stronger Bridge Mechanisms of Tor Considering Exhaustive Adversarial Models

(網羅的な攻撃者モデルを考慮した Torブリッジ機構の強化)

指導教員　　松浦幹太　教授

東京大学大学院 情報理工学系研究科 電子情報学専攻

48-136432　　馮　菲 **(Fei Feng)**

平成27年2月4日提出

Abstract

Nowadays, there is an increasing concern about the protection of anonymity and privacy in the online world. 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. Effective attacks to block or attack the bridge mechanism are being found in existing works, and also have been conducted in the wild. However, the vulnerabilities of the current bridge mechanism have not been thoroughly investigated yet.

In this paper, we first introduce the background knowledge of anonymous communication systems, Tor, and metrics that evaluate the anonymity of Tor in related works. Then, we exhaustively summarize possible adversarial models. Next, we thoroughly investigate the vulnerabilities of the current design under these adversarial models. After that, we propose two alternative designs of bridge mechanism, and compare them with the current design through simulation. We conclude that one of our proposals is effective against malicious bridge model and bridge set fingerprinting model. We also show it does not negatively affect the performance.

However, it cannot mitigate the enumeration attack of bridges, which is a serious threat to the bridge mechanism, because bridges are effective against censors only as long as an adversary cannot easily enumerate their IP addresses. To counteract this situation, we continue to evaluate two anti-enumeration defenses for Tor bridges. We show one of them is effective through simulation, and do further analyses and discussions on its concrete design to encourage future research on this aspect.

Finally, we conclude this research and future work.

Contents

Chapter 1 Introduction

1.1 Anonymous Communication System

1.1.1 Terminology

Pfitzmann and Kohntopp [27] define anonymity as follows:

> *Anonymity is the state of being not identifiable within a set of subjects, the anonymity set.*

In a system where senders send messages to recipients using a communication network like Figure 1.1, anonymity is the stronger, the larger the respective anonymity set is and the more evenly distributed the sending or receiving, respectively, of the subjects within that set is.

In such a system, sender anonymity, recipient anonymity and relationship anonymity are respectively defined as follows:

- Sender anonymity: a particular message is not linkable to any sender and that to a particular sender.
- Recipient anonymity: a particular message cannot be linked to any recipient an that to a particular recipient.
- Relationship anonymity: it is untraceable who communicates with whom. In other words, sender and recipient are unlinkable.

Anonymous communication is a privacy enhancing technology which is designed for the unlinkability of communication parties. In other words, most anonymous communication systems only ensure the relationship anonymity, which is considered sufficient in most practical cases.

1.1.2 Motivation of Anonymity Communication System

The Internet brings us convenience, but also hurts our privacy. In today's expanding online world, there is an increasing concern about the protection of anonymity and privacy in electronic services. With some tools, it is not difficult to eavesdrop other Internet users. Encryption solves parts of but not all of the problem. It can hide the communication contents such as data payloads, but it cannot hide data packet

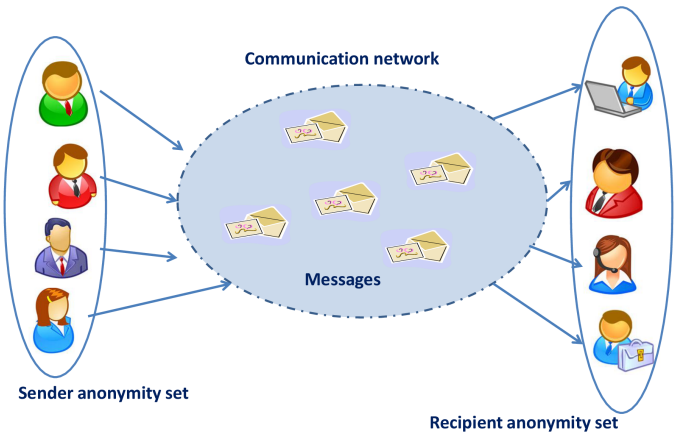


Figure 1.1: A communication network

headers, which leaks the identity of communication parties. Anonymous communication systems provide foundation for users to share information over public networks without having their privacy compromised.

Individuals use anonymous communication tools to keep websites from tracking them and their family members, or to connect to news sites, instant messaging services, or the like when these are blocked by their local Internet providers. Individuals also use anonymous communication tools for socially sensitive communication, e.g. chat rooms and web forums for people with illnesses. Journalists use anonymous communication tools to communicate more safely with whistle-blowers and dissidents. Activists use anonymous communication tools as a mechanism for maintaining civil liberties online.

Moreover, big organizations such as embassies use anonymous communication systems to exchange information with their home country. Non-governmental organizations (NGOs) use them to allow their workers to connect to their home website while they are in a foreign country, without notifying everybody nearby that they are working with that organization. Corporations use them as a safe way to conduct competitive analysis, and to protect sensitive procurement patterns from eavesdroppers. They also use it to replace traditional VPNs, which reveal the exact amount and timing of communication.

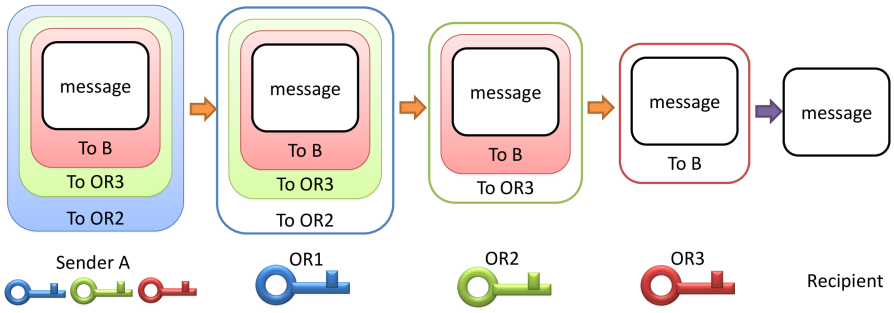


Figure 1.2: An overview of onion routing

1.2 Background on Tor

Nowadays, Tor[7], which is an implementation of onion routing, is the most popular anonymous system. It is a low-latency anonymity, privacy and censorship resistance 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.

1.2.1 Onion Routing

The first published, as well as the first deployed distribution system for low-latency anonymous Internet communication was onion routing. Onion routing is a technique for anonymous communication over the Internet. It was developed by Goldschlag *et al.*[12], and patented by the United States Navy in US Patent No.6266704[29]. Tor is the predominant technology that employs onion routing.

First, the Tor client exchanges a session key with the entry guard, middle relay and exit relay respectively by an authenticated Diffie-Hellman exchange. Then the three relays utilize their shared session keys with the client for encrypting all their correspondence in symmetric encryption, which is significantly more efficient than using asymmetric encryption. Next, messages are repeatedly encrypted by the Tor client and then sent through the three onion routers. Like someone peeling an onion, each onion router decrypts a layer of encryption to uncover routing instructions, and sends the message to the next router according to the instruction (like Figure 1.2).

In this way, every router only knows its previous and next communication party and none of the three relays knows the relation between the sender and the recipient. The advantage of onion routing is that it is not necessary to trust each cooperating router; if one router on the path is malicious or compromised, anonymous communication can still be achieved.

Loose Source Routing. Since the earliest versions of Onion Routing[12], the protocol has provided the feature: "loose source routing". The "strict source routing" and "loose source routing" are defined in RFC0791[28]. In strict source routing, the source of a message chooses every node on the message's path. But in loose source routing, the message traverses the selected nodes, but may also traverse other nodes as well. In other words, the client selects nodes N_a, N_b, and N_c, but the message may in fact traverse any sequence of nodes $N_1 ... N_j$, as long as $N_1 = N_a$, $N_x = N_b$, and $N_y = N_c$, for $1 < x < y$. Tor has retained this feature, but has not yet used it.

1.2.2 Design of Tor

The Tor network, the implementation of the second generation of the 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 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 the list, which is called consensus file. The client then selects three of these relays, and builds an encrypted channel, which secured by a session key established through an authenticated Diffie-Hellman 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 1.3 shows the structure of the Tor network.

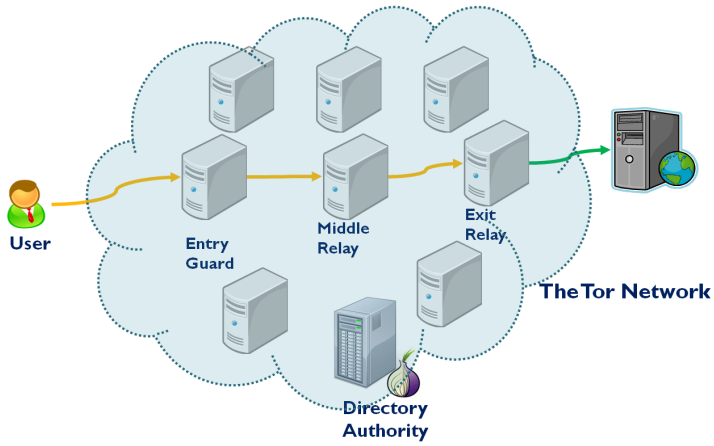


Figure 1.3: The structure of the Tor network

1.2.3 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 entry guard. Entry guards were first proposed by Wright *et al.*[44]. Only relays considered fast and stable can receive Guard flag. Details are explained in Section 1.2.4.

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[39], there are four kinds of relays to select in 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 types of relays, they will be weighted by an additional factor during path selection. This algorithm will be explained in details in Section 1.2.7.

1.2.4 Entry Guard Relays

Entry guards were first proposed by Wright *et al.*[44]. Instead of choosing a new guard every time, every Tor client maintains a list, which is called guard list, of several (by default, three) pre-selected guards. Each client ensures that the number of guards

(both online and offline) in its guard list is at least the default number at all the time. If a guard becomes offline, either temporarily or permanently, and there are fewer than two online guards in the list, a new entry guard will be selected, but each previous guard will be retried periodically. 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. Since Tor 0.2.4, the range has been extended to 60-90 days. After that period of time, the expired guards will be dropped and repopulated, which is called guard rotation. When a path is constructed by the Tor client, the entry relay to be used is selected uniformly at random from the client's guard list.

Since all Tor relays are run by volunteers, it is hard to know which ones can be trusted. An adversary can also donate some routers and participate in paths. The problem occurs when the adversary can watch the traffic passing both the entry guard and exit relay. Tor does not intentionally delay traffic or insert dummy packets, which results an almost unchanged traffic pattern through the path. It was demonstrated that by observing both ends of a path, timing patterns are enough to link senders with recipients[14]. Although the probability of correlating any single path is very low, if clients choose relays randomly every time, the adversary will see some paths from each client with a probability of 1 over time.

The entry guard mechanism turns the attack from a deterministic one to a probabilistic one. The client has a nonzero chance of remaining safe during the period of guard rotation. While the rotation lets these 'unlucky' clients to regain some privacy. It is controversial whether it is better to be compromised with some probability all the time until next guard rotation or either to be completely safe. Øverlier and Syverson[25] proved that the latter is better and hence the guard mechanism was introduced to Tor.

This entry guard design protects users in the following three ways according to the Tor Project's official blog[15].

First, entry guards can mitigate the predecessor attack[45]. Without entry guards, the client has to select a new relay for every path, eventually an attacker who runs a few relays would be the first and last relay of the path. Since Tor selects relays weighted according to bandwidth, an attacker with sufficient resource can increase the possibility and the rate that he compromises paths. With entry guards, the risk of end-to-end correlation for any given path is the same, but the cumulative risk for all the paths of the user over time is controlled.

Second, entry guards help to mitigate the "denial of service as denial of anonymity" attack [2]. During such an attack, the attacker fails any path in which he is a part of but does not occupy the both ends to force the client to generate more paths and thus increasing the overall opportunity of compromising paths. Entry guards decrease the

risk because the client never chooses the first relay outside of those pre-selected ones in the guard list.

Third, entry guards increase the initiative cost for an adversary who runs malicious relays for the purpose of de-anonymize users. Without entry guards, the attacker can donate several relays and immediately start having opportunities to observe the traffic at the entry position. However, with the mechanism of entry guards, new malicious relays will not have the Guard flag so they will not be chosen as the first relay of any path for quite a long time. Even once these malicious relays earn the Guard flag, clients who have already selected guards will not select new guards until next guard rotation. Details of the life cycle of a new relay are explained by a Tor Blog[35].

1.2.5 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[42, 43].

To counteract this situation, designers of Tor introduced a new method of accessing the Tor network: bridges[6], 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 guards 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.

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[37] distributes bridge information through the web and e-mail, 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[3] or send an email to bridges@bridges.torproject.org with the line "get bridges" in the body of the

mail.

1.2.6 Bandwidth

The bandwidth available is very important to the Tor network. More bandwidth means better efficiency and a greater number of users, and may contribute to a reduction in latency. However, a single relay with very high bandwidth will cause bias in path selection, because the current algorithm selects relays from a distribution that favors higher-bandwidth relays but also allows low-bandwidth relays to be utilized to some extent. As a result, there is an upper bound of bandwidth which is currently 10MBps. The purpose is to avoid a certain high-bandwidth relay attracts too much traffic.

Bandwidth is one major factor in the path selection algorithm in Tor. Initially, it was self-reported; however, recognizing that malicious relays can lie, it was changed to a consensus bandwidth.

1.2.7 Path Selection Algorithm

The path selection algorithm first selects an exit relay, then the guard relay (from the list of guards that the client pre-selected), finally the middle relay. There are additional constraints on the path, such as avoiding using more than one relay from a given /16 network or the same relay family. Details are available in the Tor Path Specification[39].

When the design paper[7] was written, Tor used Uniform Selection, in which relay is selected from a set of relays with uniform probability. But this approach creates terrible bandwidth bottlenecks: a relay that would allow 10 times as many bytes per second as another would still get the same number of paths constructed through it. The algorithm Tor employs now is Adjusted Bandwidth-Weighted Random Selection, where weights specified in the Tor Directory Protocol[38] are applied to the bandwidths, and relays are selected with probability proportional to those weighted bandwidths.

The weighting factors are multiplied by the bandwidths of relays. The purpose of the weighting factors is to discourage the use of scarce resources in other position in times of shortage, i.e. if the Tor network lacks exit relays, the weighting factors make it more likely an exit relay will be used at the exit position rather than middle.

The weights are in the form of W_{xy} where x is the position (guard (g), middle (m), exit (e)) of the relay, and y is the type of relay (guard relays are denoted by g, exit relays by e, guard + exit relays by d, and non-flagged relays by m). G denotes the total bandwidth of guard relays; M denotes the total bandwidth of non-flagged relays;

E denotes the total bandwidth of exit relays; D denotes the total bandwidth of guard + exit relays. Details are in the Tor Directory Protocol[38].

The weights are supposed to satisfy several balancing constraints. For example, if neither guard nor exit is scarce, the weighting factors will be

$W_{gd} = 1/3;$

$W_{ed} = 1/3;$

$W_{md} = 1/3;$

$W_{ee} = (\text{E} + \text{G} + \text{M}) / (3 * \text{E});$

$W_{me} = 1 - W_{ee};$

$W_{mg} = (2 * \text{G} - \text{E} - \text{M}) / (3 * \text{G});$

$W_{gg} = 1 - W_{mg}.$

1.3 Metrics for Evaluating Anonymity of Tor

The general consensus from the literature is that the further a path selection algorithm deviates from uniform selection, the lower anonymity it provides. It means that Tor sacrificed its anonymity to some extent for a better performance to achieve its low-latency feature. Following this intuition, existing works adopt the following metrics for evaluating the anonymity of Tor.

1.3.1 Shannon Entropy

One simple metric for anonymity system security is the size of the anonymity set, but this does not capture the non-uniformity. For this reason, Danezis and Serjantov[30] proposed entropy of the anonymity set as the effective size, and Diaz *et al.*[5] proposed normalised entropy as the system's degree of anonymity. Their proposals have been used extensively in the study of high-latency remailers, although over users rather than the paths.

Bauer *et al.*[1] adopt normalised Shannon entropy as their definition of anonymity of Tor's path selection. i.e. if the probability of selecting relay x_i is q_i, for $1 \leq i \leq n$, the entropy is

$$H = \sum_{i=1}^{n} q_i \log_2 q_i \tag{1.1}$$

H is maximal at $\log_2 n$ when the selection probability is uniform over all relays, hence entropy can be normalised, giving a quantity $0 \leq S \leq 1$ representing how skewed the probability distribution is:

$$S = \frac{H}{\log_2 n} \tag{1.2}$$

1.3.2 Gini Coefficient

Snader and Borisov[33] adopted a different metric, the Gini coefficient, but it also measures the deviation from uniform path selection. 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:

$$G = \frac{1}{n}\left(n + 1 - 2\left(\frac{\sum_{i=1}^{n}(n+1-i)y_i}{\sum_{i=1}^{n} y_i}\right)\right) \tag{1.3}$$

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. When used as a measure of income inequality, the most unequal society will be one in which a single person receives 100% of the total income and the remaining people receive none (G = 1); and the most equal society will be one in which every person receives the same income (G = 0).

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.

1.3.3 Fraction of Possible Snooping

Tor does not delay messages or introduce dummy traffic intentionally, which result in the almost unchanged traffic patterns. It was demonstrated that by observing both ends of a circuit like Figure 1.4, timing patterns are enough to link senders with receivers, which is called end-to-end correlation attack. This attack has also been shown to work even if a small proportion of packets (e.g. 1 in 2000) can be observed [24].

As a result, fraction of possible snooping is another metric that has been widely researched into. Bauer *et al.*[1] and Murdoch *et al.*[23] studied the probability of an adversary, who can control several malicious nodes, observing the both ends, by using

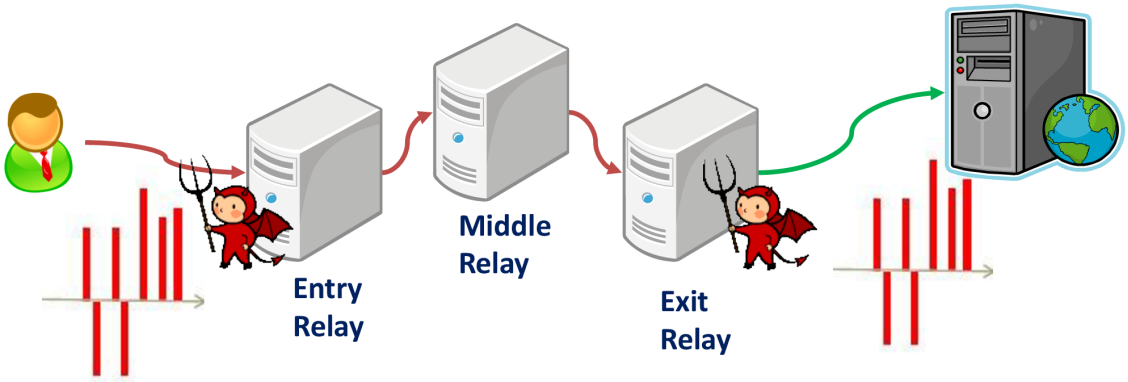


Figure 1.4: The end-to-end correlation attack.

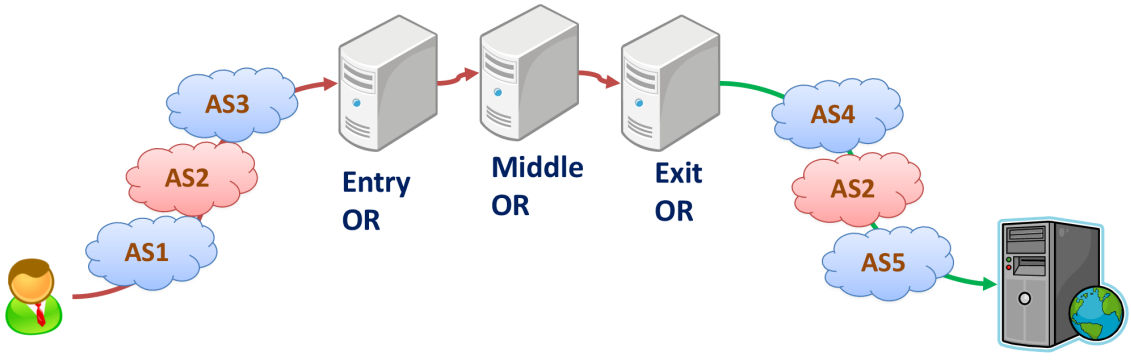


Figure 1.5: In this case, AS2 can observe both ends of the circuit to de-anonymize the client.

the number of malicious nodes and capacity as parameters. Others like Feamster *et al.*[11] and Edman *et al.*[10] studied the probability of an AS observing both ends of a circuit, like Figure 1.5.

1.4 Motivation

In today's expanding online world, there is an increasing concern about the protection of anonymity and privacy in electronic services. Anonymity is an important issue in electronic payments, electronic voting, electronic auctions, and also for email and web-browsing. Anonymous communication is a privacy enhancing technology which is designed for the unlinkability of communication parties. The Tor[7] network is the most popular anonymous communication system. It is a low-latency anonymity, privacy and censorship resistance 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 the 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 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*[6], 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 an ISP is filtering connections to all the known Tor relays, it probably will not be able to block all the bridges.

Effective attacks[19, 42, 43] to block or attack the bridge mechanism are being found and also have been conducted in the wild. The vulnerabilities of the current bridge mechanism, however, have not been thoroughly investigated yet.

Winter and Lindskog[43] have investigated how the Great Firewall of China is blocking Tor, in which they investigated how and how long Tor bridges are blocked. However, they have not quantitatively investigated how much bridges will be blocked if there are certain number of bridge users within the adversary's network bounds. Ling *et al.*[19] have analyzed the effectiveness of bridge-discovery approaches and discovered 2369 bridges by only one Tor middle relay in 14 days. Their work conducted experiments on the real Tor network, which could harm real Tor users' anonymity. We

consider it a better choice to do simulation experiments with Tor's historical data. A Tor blog[18] indicates that entry guards' fingerprinting problem may also related to bridges, but no further works have been done.

These motivate us to quantitatively investigate the vulnerabilities of Tor's bridge mechanism through simulation, and propose stronger bridge mechanisms against those attacks.

1.5 Contributions

Our first contribution is having developed a Tor path simulator, which accepts Tor's historical data as input and precisely simulate Tor clients, Tor bridge clients and Tor paths. It is the technological base for this research. Furthermore, because of this simulator's extensibility, it can also be utilized in other researches. We have used it to conduct a research on attacks against the Tor network (Publication <1>), and a research on Tor's entry guard mechanism (Publication <2>). It can also be utilized to simulate other Tor-related phenomena that do not involve actual network traffic. Examples include analysis of client path diversity, research on Tor's path selection algorithm, and assessing whole-network effects of heterogeneous client configurations.

The second contribution of this work is that we thoroughly summarize four known attacks against Tor bridges, and quantitatively investigate the vulnerabilities of the current Tor bridge mechanism under these four adversarial models.

Then we propose two bridge mechanisms as alternatives for the current round-robin method. Through simulations, the top one method is proved to be effective in mitigating two of these adversarial models. This is our third contribution.

However, the top one method cannot mitigate the adversarial model of enumeration of bridges by malicious middle relays, so we continue to evaluate two anti-enumeration defenses for Tor bridges. The evaluation results prove that letting bridge has guards can effectively mitigate the enumeration attack, which is the fourth contribution of this work. We also do some further discussions in hoping to encourage further research on this aspect.

1.6 Organization

In the following chapters, we summarize four adversarial models in Chapter 2. Then we investigate the vulnerabilities of the current Tor bridge mechanism. The experiment setup and results are described and discussed in Chapter 3. In Chapter 4 we propose two alternative designs and compare them with the current design. After

that, we evaluate and discuss two anti-enumeration defenses in Chapter 5. We finally summarize our conclusions in Chapter 6.

Chapter 2 Adversarial Models

Effective attacks to block or attack the bridge mechanism [19, 42, 43] 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 the bridge users. As a result, they also conduct attacks in different means - passively or actively. In order to propose 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.

2.1 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[42]. 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 attack procedure is illustrated by Figure 2.1. The details of how the Great Firewall blocks are described by Winter and Lindskog[43]. Iran has also used this strategy to block Tor twice[16].

2.2 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.*[19]. 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 doesn't have the Guard flag and rejects to be an exit relay, the only position it will end up is the middle one. Then nodes that build paths to connect to this relay are normal relays and bridges.

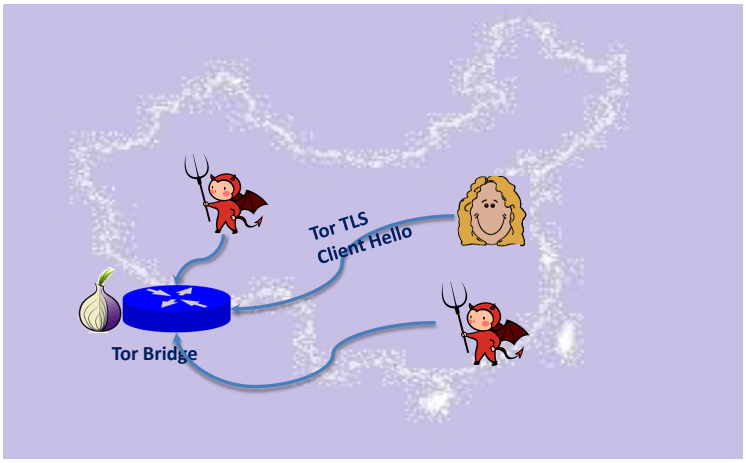


Figure 2.1: The procedure of how censors block a Tor bridge.

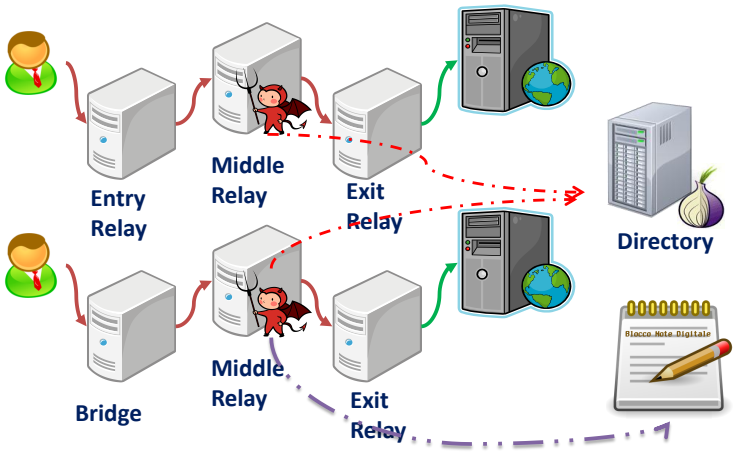


Figure 2.2: The procedure of how malicious middle relays enumerate Tor bridges.

The adversary can easily identify whether the node, which connected to his malicious middle relay, is bridge or not, by referring to the public consensus files which contains all IP address of relays. The attack procedure is illustrated by Figure 2.2. This attack can also be conducted by operating a relay and actively scan the port of each client that connects to it, to confirm which ones are running services of Tor protocol[21, 40].

2.3 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

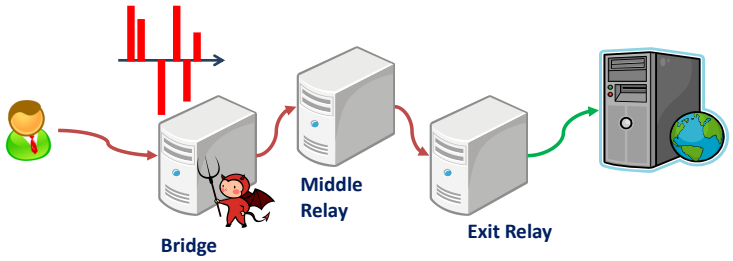


Figure 2.3: The illustration of malicious bridge compromising a Tor bridge user.

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 statistical profiling attack or fingerprinting attack[13, 26, 31] on the user. The illustration of this attack is shown by Figure 2.3. 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.

2.4 Bridge Set Fingerprinting

It has been discussed in the Tor community that the sets of entry guards might be the fingerprint for a user[8, 18]. 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 Tor bridge, and it is even worse because there is guard rotation for normal Tor clients, while there is no rotation of bridges. The illustration of this attack is shown by Figure 2.3.

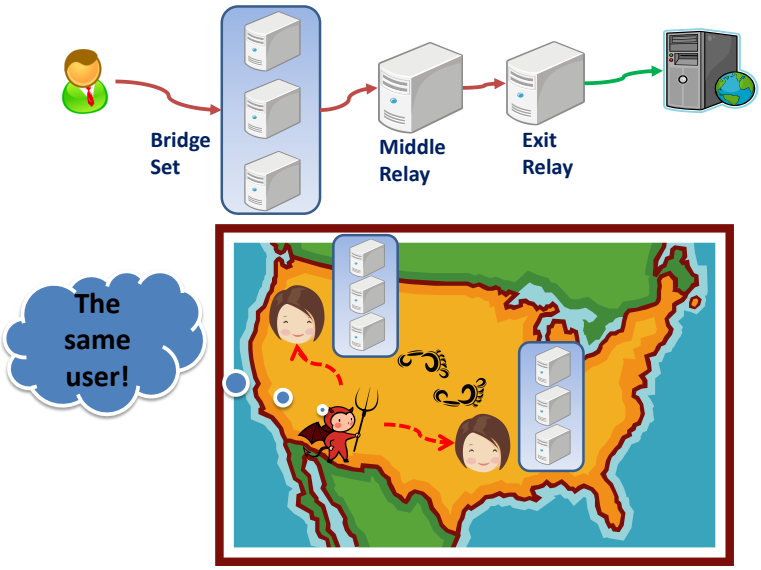


Figure 2.4: The illustration of how an attacker links a Tor bridge user across destinations.

Chapter 3 Vulnerabilities of the Current Bridge Mechanism

To investigate how vulnerable the current bridge design is under these aforementioned adversarial models, and the chance that an adversary blocks or compromises Tor bridges, we conduct simulation experiments with a Tor path simulator developed by ourselves, which accepts publicly available data provided by the CollecTor[4] as inputs.

3.1 Experiment Design

We have developed a Tor bridge path simulator for bridge users by using Stem[34] which is a Python controller library for Tor. The simulator is based on information from Tor Path Specification[39], Tor Directory Protocol[38], Tor Bridge Specification[36] and the Tor source code. The simulator 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, because 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[4]. However, the sanitizing does not affect our simulation results 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 which mainly utilizes port 80 and 443. When simulating a path, our simulator adopts the Adjusted Bandwidth-Weighted Random Selection, which is the algorithm utilized by Tor now. Weighting factors are multiplied by the bandwidths of relays, and relays are selected with probability proportional to those weighted bandwidths. Details are introduced in Section 1.2.7.

It is worth mentioning that, in our simulator, clients use the measured bandwidth in consensus files when choosing relays, while evaluates the performance that users

experience with bridges' or relays' advertised bandwidth recorded in descriptors, which is the same way as what the current Tor clients do. As mentioned in the Tor source code, when weighting bridges, they enforce 20KB/s as lower and 100KB/s as upper bound of believable bandwidth, because there is no way for them to verify a bridge's bandwidth currently.

Our simulator not only simulates paths but also bridge users' clients, in all of which there are N configured bridges chosen from all the bridge network statuses of February 2014. 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, to simulate the bridge users and adversary on that day.

All the historical data used in our experiment are downloaded from CollecTor[4]. We run the simulator on a 8-core 3.20GHz Intel Core i7 machine with 23.5GB of memory on Ubuntu 12.04 with the 3.2.0 Linux kernel.

3.2 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 the clients, there are 4 to 12 bridges configured (the number of bridges is chosen uniformly at random). 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 3.1.

For not all the bridges recorded in February's bridge network statuses are online on 28th February, some clients may have no access to the Tor network even if there is no censorship event. As shown by Table 3.1, there are 17.9% clients that 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 3.1. 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.

Table 3.1: 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%

3.3 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 to be an exit relay, the only position it will end up is the 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 3.2 are average values of the results of the three rounds.

On 28th February, there are 3014 bridges online. As shown by Table 3.2, when the adversary controls 200 malicious relays, over 80% unique bridges running on that day will be enumerated. This attack can be conducted with comparatively low cost because the adversary can rent IPs on Amazon EC2.

3.4 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

Table 3.2: Simulation results of 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	1286	42.67%
200	1617	53.65%

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 of a Tor path. Then the adversary may perform statistical profiling attack or fingerprinting attack[13, 26, 31] on the user.

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 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 3.3. This attack could also be performed by renting IPs on Amazon EC2 with comparatively low cost.

Table 3.3: Simulation results of clients compromised by malicious bridge.

Number of Malicious Bridges	Number of Compromised Clients	Percentage of Compromised Clients
0	0	0.00%
50	548	2.74%
100	1112	5.56%
150	1611	8.06%
200	2094	10.47%

3.5 Bridge Set Fingerprinting

It has been discussed in the Tor community that the sets of entry guards might be the fingerprint for a user[8, 18]. 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 bridge, and it is even worse because there is guard rotation for normal Tor clients, while there is no rotation of bridges.

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). 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 set, 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 3.4. It is clear from these results that there is trade-off between availability and privacy in terms of the possibility of being fingerprinted. We consider 7 is the 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 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.

Table 3.4: 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	49416	14295	50.6%	7.00
4	39430	22371	60.6%	4.47
5	30936	31133	69.1%	3.21
6	24669	39453	75.3%	2.53
7	19486	47780	80.5%	2.09
8	15367	55263	84.6%	1.81
9	12229	61733	87.8%	1.62
10	9655	68103	90.3%	1.47
11	7683	73065	92.3%	1.37
12	5965	77888	94.0%	1.28

Chapter 4 Countermeasures

In Chapter 3, we investigate the vulnerabilities of the current bridge mechanism under four different adversarial models. In this Chapter, we propose two alternative methods of round-robin over all bridges, and investigate whether these two proposals make the bridge mechanism more robust against the aforementioned attacks compared to the current round-robin method.

4.1 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 sticking to 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 an usable bridge sequentially from the top of the list, The client will return to use the ones that precede in the list, when one of them becomes online again.
- **Top three method:** The second method is randomly choosing 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 first two bridges. This method imitates the current entry guard mechanism.

4.2 Comparison

We conduct simulation experiments to compare the current round-robin method with the two alternatives under these adversarial models except the censorship model. It is because our proposals can not mitigate the censorship model in which the adversary identifies Tor protocol. It is the weakness of Tor but not weakness of Tor's bridge mechanism. Since one of our major research goals is to propose stronger bridge mechanism, despite the fact that the censorship model is a serious threat, it is out of the scope of this research. Related researches have been done on how to make Tor's traffic undetectable [9, 22, 32, 41].

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 4.1 shows when the round-robin method is used, clients are most likely to be compromised. And sticking to the first one is the safest method under this adversarial model. It is easy to understand that sticking to the first one can decrease the possibility of a client exposed to malicious bridges.

Table 4.1: Comparison of three methods under malicious bridge model.

	Number of Compromised Clients	Percentage of Compromised Clients
Round-robin	1019	5.10%
Top one	523	2.62%
Top three	934	4.67%

Then we change the number of malicious bridges under three methods respectively, and find the linear approximation of the relation between the number of malicious bridges and percentage of compromised clients in Figure 4.1. Figure 4.1 shows that results are almost linear, and it also shows that under the adversarial model of malicious bridges, sticking to the first one bridge is the safest method, while the current round-robin method is most vulnerable one.

Next, we compare the three methods under bridge set fingerprinting model. We simulate 20,000 clients and 5 paths for every client respectively. The results in Table 4.2 show that when using the top one method, by 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 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 sticking to 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 sticking to the first one, the bridge set that a client has will be its first bridge, so many clients share the same bridge set. In contrast, in the round-robin method, 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 expected anonymity set,

Table 4.2: 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	12655	1.58	10775
Top one	3004	6.66	86
Top three	12642	1.58	10765

that is the number of expected users per bridge set, for the three methods respectively, in order to analyze the probability of being fingerprinted. As Figure 4.2 shows, when there are 20,000 bridge users, sticking to the first one can ensure the median user an anonymity set of 6 users. On the other hand, in the round-robin or top three bridge methods, over 85% bridge sets only has one user and over 53.8% 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 4.3 shows the number of bridges enumerated by malicious relays, 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 evaluate and discuss defenses for this attack in Chapter 5.

Table 4.3: Comparison of three methods under enumeration of bridges by malicious middle relays model.

	Number of Bridges Enumerated (Round 1)	Number of Bridges Enumerated (Round 2)	Number of Bridges Enumerated (Round 3)
Round-robin	826	848	577
Top one	692	965	1782
Top three	525	408	1423

Finally, we 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 3.1, the simulator weights bridges by enforcing lower and upper bound of their bandwidth in the network status file, but evaluates the performance by their advertised bandwidth from bridge descriptor files. We use average bandwidth of all online bridges configured in a client as the metric for evaluating performance for the round-robin method, the bandwidth of the first bridge for the top one bridge 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. Table 4.4 shows that the top one method does not negatively affect the performance, while top three method causes a slight degradation in performance. The reason of this slight degradation is still unknown, and we consider it as a future task.

Table 4.4: 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	50747	50781	50696
Top one	50251	50903	50785
Top three	49042	49109	49289

4.3 Discussions

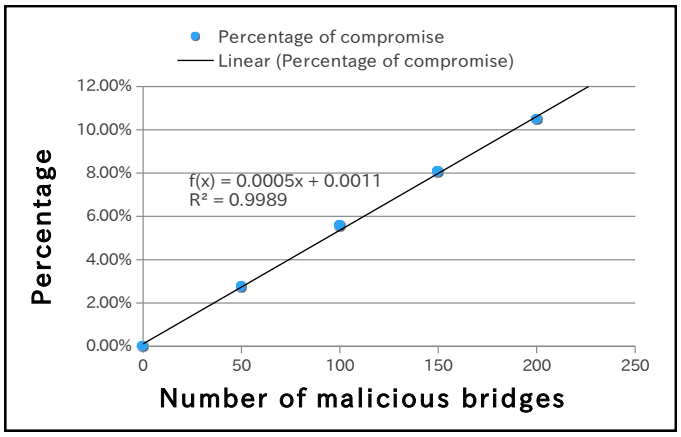
Utilizing the top one method does not mean that the client will use the same bridge forever. Rotation should be introduced if the top one method is adopted in Tor. The rotation of bridge can imitate that of entry guard.

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. Since Tor 0.2.4, the range has been extended to 60-90 days. After that period of time, the expired guards will be dropped and repopulated, which is called guard rotation. This has been described in Section 1.2.4.

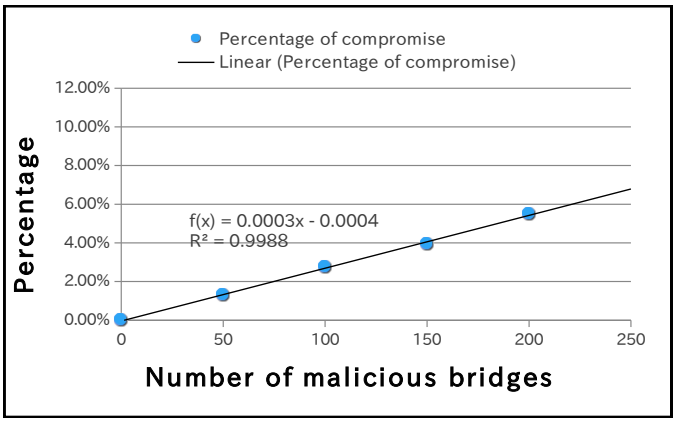
The guard rotation serves several purposes. First, as described in Section 1.2.4,

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[8, 17] have suggested that the current parameters for guard rotation, including the period of rotation time, may exposure users to more privacy loss. Dingledine *et al.*[8] suggest extending the period of guard rotation to 9 months.

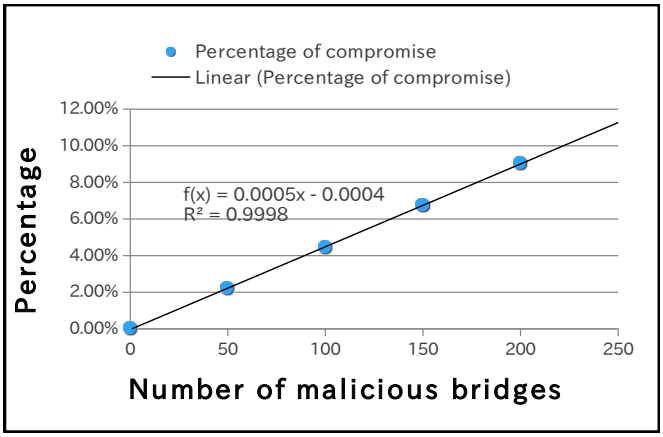
Although bridges and guards are similar in that they all serve as the entries into the Tor network, they are not the same. We thus consider the best period of bridge rotation may not be set the same value as that of guard rotation without research, so we regard it as another future task.



(a) Round-robin.

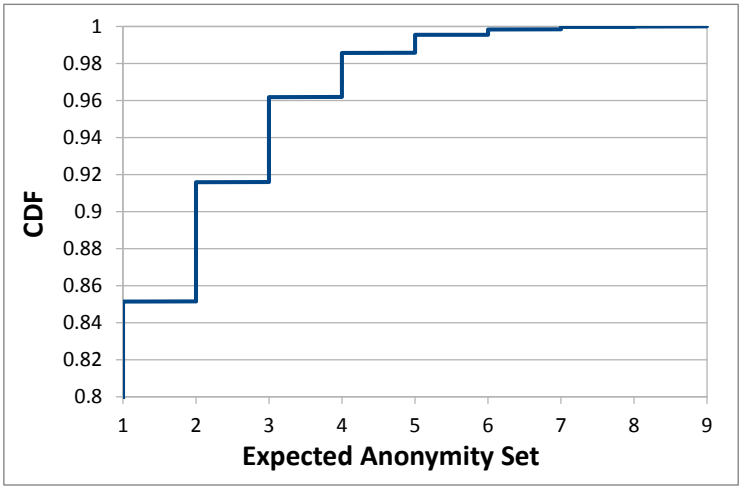


(b) Top one bridge.

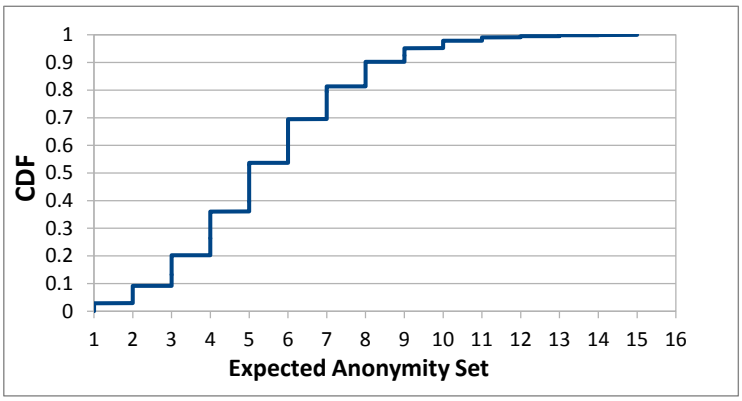


(c) Top three bridges.

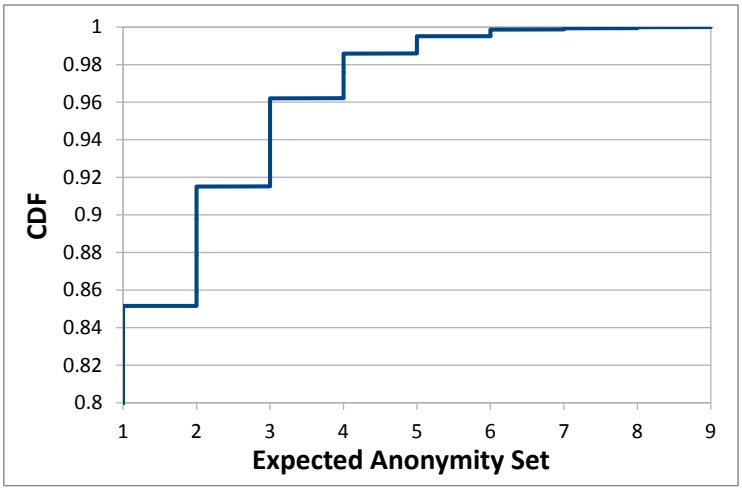
Figure 4.1: Linear approximation of the relation between the number of malicious bridges and percentage of compromised clients under three different methods.



(a) Round-robin.



(b) Top one bridge.



(c) Top three bridges.

Figure 4.2: Cumulative distribution of anonymity set size under three different methods.

Chapter 5 Evaluation of Anti-enumeration Defenses for Tor Bridges

In Chapter 4, we investigate the current Tor network's bridge mechanism's vulnerabilities to four known attacks against Tor bridges. Two alternative mechanisms are proposed. However, the results suggest that our proposals can not mitigate the adversarial model of enumeration of bridges by malicious middle relays.

The enumeration attack is a serious threat against the bridge mechanism, because bridges are effective against censors only as long as an adversary cannot easily list up their IP addresses, or the adversary could block connections to bridges just as he blocks normal Tor relays. As a countermeasure, the Tor community has discussed the idea of bridge guards[20], but the effectiveness of bridge guards has not yet been thoroughly verified and evaluated. What's more, the concrete design of bridge guards has not yet been proposed. This motivates us to evaluate two possible designs and discuss concrete design in this chapter.

5.1 Defenses

5.1.1 Bridge Guard

To counteract the enumeration attack of Tor bridges, bridge guard has been discussed in the Tor community[20]. The idea is to let bridges use guard nodes, which means to route requests from bridge clients through an additional layer of guard nodes. Then only a bridge's guard nodes can tell that it is a bridge, and the attacker needs to run or observe many more nodes in order to enumerate a large number of bridges.

Similar to the idea of entry guard introduced in Section 1.2.4, letting bridges use guards turns the attack of enumeration of bridges by malicious relays from a deterministic one to a probabilistic one. As long as there is no malicious guards in the guard list, the bridge can stay safe during the period of guard rotation.

Further more, if the adversary runs malicious relays by himself to conduct this attack, to make his malicious relays be used at the entry position, Guard flags will be needed. This is not as easy as to be used at the middle position. The directory authorities monitor the availability and bandwidth capacity of all relays in the Tor

network, and they assign Guard flags to relays. The current algorithm for assigning Guard flag has three requirements[38]:

1. The relay needs to have appeared longer than 12.5% of all the relays, or 8 days ago, whichever is shorter.

2. The relay needs to advertise at least the median bandwidth in the network, or 250Kb/s, whichever is smaller.

3. The relay needs to have at least median weighted-fractional-uptime (WFU) of relays in the Tor network, or 98% WFU, whichever is smaller. The WFU measures the fraction of uptime of a relay in the past. WFU values are discounted by factor 0.95 every 12 hours.

These requirements only allow routers that have been running for a long time and provide high bandwidth to have Guard flags, so letting bridges use guards can also increase the initiative attack cost for the adversary.

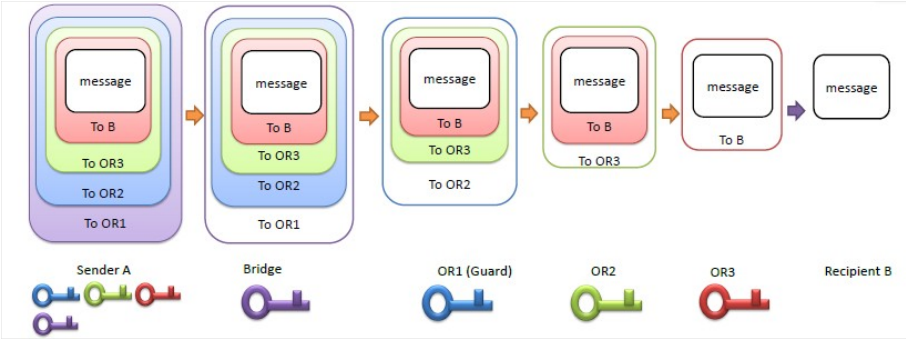
5.1.2 Designs

There are two designs that have been discussed.

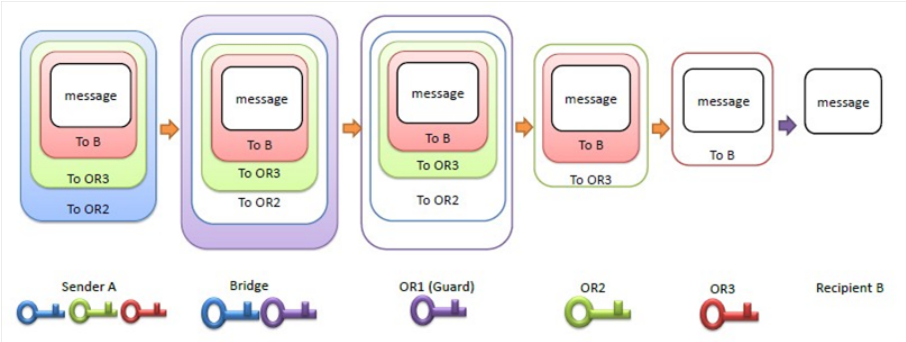
- **Client chooses guards.** If the loose source routing is not adopted, we could have the client choose three guards as its possible next hop after the bridge, which adopts strict source routing. When a path is being constructed, the client chooses one of the guards from its guard list uniformly at random. Then it chooses one bridge from its bridge list by using the current round-robin method.

- **Bridge has guards.** If the loose source routing is adopted, we could let bridge itself choose guards that all paths it handles will use. Specifically, when a bridge receives an extend cell from a client to extend a path, it should first extend to its guard, and then relay that extend cell through the guard to the next hop. The bridge should add an additional layer of encryption to outgoing cells on that path corresponding to the encryption that the guard will remove, and remove a layer of encryption on incoming cells on that path corresponding to the encryption that the guard will add.

Figure 1.2 shows the multi-encryption of onion-routing of normal Tor paths, while Figure 5.1a and Figure 5.1b show the illustrations of multi-encryption of these two different designs. Figure 5.1a shows that strict source routing just conducts the encryption for four times instead of three times. In contrast, when loose source routing



(a) Client chooses guards (strict source routing).



(b) Bridge has guards (loose source routing).

Figure 5.1: The illustration of two different designs of anti-enumeration defenses.

is adopted, the bridge shares a symmetric key with its bridge, so this additional layer seems transparent to the client.

We evaluate and discuss these two designs in the following sections.

5.1.3 Evaluation through Simulation

The experiment setup has been described in Section 3.1. What is different is that in the following experiments, all the guards configured in the clients' or bridges' guard lists are chosen from all the consensus files of September 2014. In the following experiments, we use the bridge network status and relay consensus of the Tor network on 30th September 2014 as input, to simulate the bridge users and adversary on that day. As described in Section 1.2.4, each client ensures that the number of guards in its guard list is at least the default number at all the time. If a guard becomes offline, either temporarily or permanently, and there are fewer than two online guards in the list, a

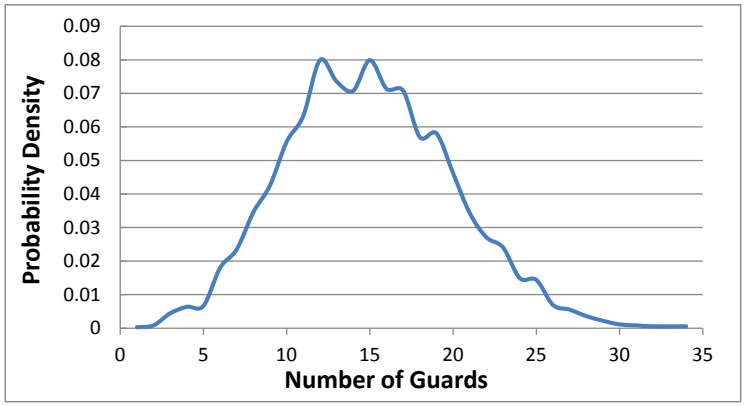
new entry guard will be selected. Our simulator first randomly chooses three guards for every client or bridge, and then checks whether these three guards are online on 30th September 2014. If there are less than two online guards, new guards will be selected.

We first compare the two designs in terms of security. We simulate 20,000 Tor bridge clients and 5 Tor paths for every clients in all the following simulations. Since a Tor client changes its path every 10 minutes, it could be considered that the adversary performs the attack for 50 minutes in this scenario. 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 guard relays. By changing the number of malicious guards controlled by an adversary, we investigate the number and percentage of bridges enumerated by the adversary.

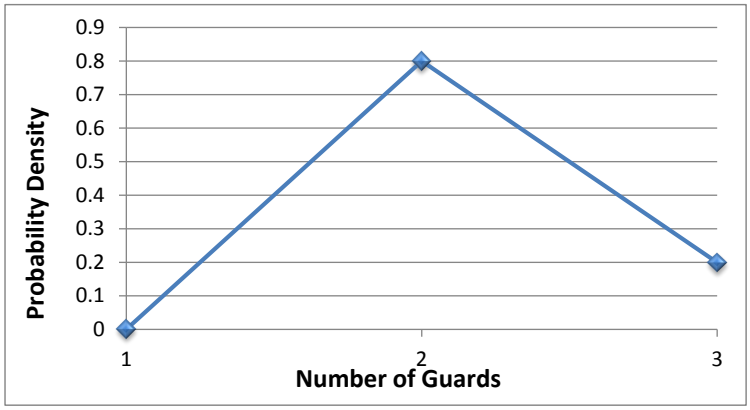
Table 5.1: Simulation results of enumeration of bridges by malicious guards.

	Number of Malicious Guards	Number of Enumerated Bridges	Percentage of Enumerated Bridges
Client Chooses Guards	50	324	8.96%
	100	727	20.11%
	150	1044	28.87%
	200	1315	36.37%
Bridge Has Guards	50	56	1.55%
	100	129	3.57%
	150	164	4.54%
	200	191	5.28%

It is obvious from Table 5.1 that letting a client choose guards increasingly exposes the bridges configured in the client to malicious guards. In contrast, letting bridge having guards can effectively mitigate this attack. This is because in this way every bridge exposes to fewer guards , and is thus less likely to encounter malicious ones. We run simulations to investigate the exposure by simulating 20,000 Tor bridge clients and 5 Tor paths for every clients under these two different designs. The average number of guards that a bridge exposes to is 14.94 when client chooses guards. In contrast, when bridge has guards, a bridge only exposes to 2.20 guards by average. We plot the probability density functions of how many guards that every bridge exposes to under the two designs respectively in Figure 5.2, which show the significant difference between these two designs in terms of exposure to guards.



(a) Client chooses guards.



(b) Bridge has guards.

Figure 5.2: The probability density functions of how many guards that every bridge exposes to.

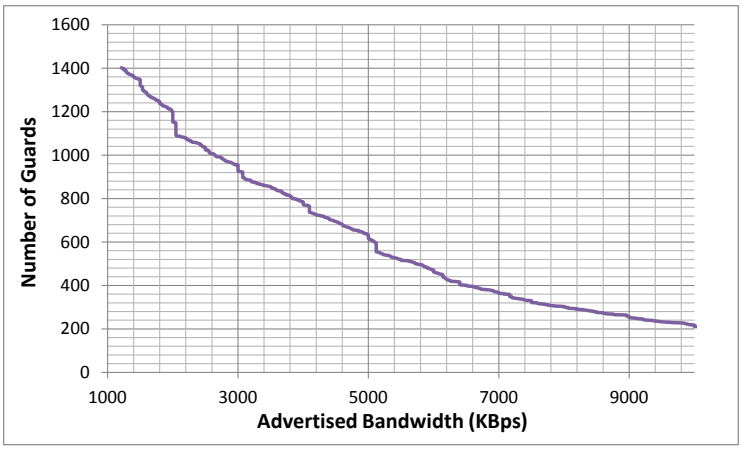


Figure 5.3: Number of guards remaining as the threshold increases.

All these results and analyses suggest that if bridge guard mechanism is adopted into Tor as countermeasure against enumeration of bridges, adopting the loose source routing to let bridge have guards is a good solution. We further discuss the concrete design of letting bridge have guards in following section.

5.2 Further Discussions

5.2.1 Fast Guards

Although the results suggest the idea that adopting the loose source routing feature of onion routing to implement bridge guards can mitigate the adversarial model of enumeration of bridges. The disadvantage is also obvious. It forces bridge clients construct four-hop paths, while normal Tor clients use three-hop paths. Bridge clients thus have to suffer worse performance.

One solution is to only use guards with high bandwidth. There are 1403 running guards on 30th September 2014. Figure 5.3 shows the number of guards remaining as the threshold increases. Figure 5.4 shows the fraction of remaining guard bandwidth if the threshold increases and the number of guards decreases. It can be seen if we only use the top 70% fast guards, the total guard bandwidth only reduces by 9.1%, so we assume it practical to increase the bandwidth threshold of guards. All the bandwidth here is advertised bandwidth, because Tor's directory servers assign Guard flag according to relays' advertised bandwidth.

To verify our assumption, we conduct simulations. Dingledine *et al.*[8] utilize *the bottleneck bandwidth* as their metric for performance. The bottleneck bandwidth is the minimum bandwidth of the guard, middle and exit relays. We adopt *the percentage*

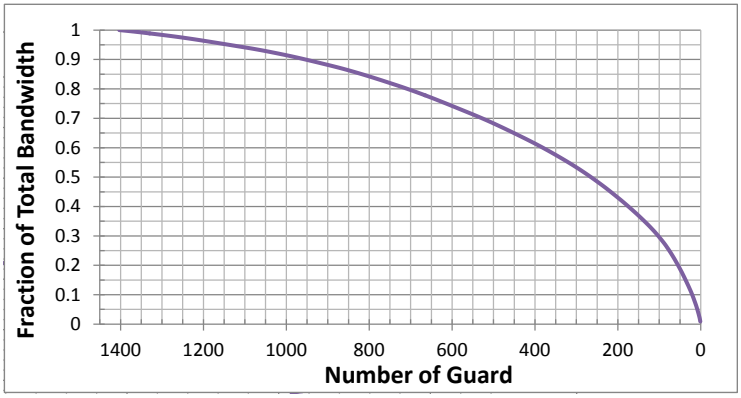


Figure 5.4: Fraction of remaining guard bandwidth as the number of guards decreases with higher bandwidth threshold.

Table 5.2: Simulation results of guards with higher bandwidth threshold.

Minimum of Guard Bandwidth Rank	Number of Remaining Guards	Average Bandwidth at Guard Position (KB/s)	Percentage of Guard Bottleneck	Entropy over Guard Position
0%	1403	7767	55.87%	10.23
10%	1263	8792	52.15%	10.07
20%	1122	9384	49.90%	9.89
30%	982	10434	45.65%	9.64

of guard bottleneck as our performance metric, which is defined as the percentage of paths whose bandwidth bottleneck is at guard position.

Results in Table 5.2 suggest that while only using the top 70% fast guards will decrease the number of guards to under 1000, the average bandwidth at guard position increase by 34.3%, which could be considered worthwhile. It also shows if we use the top 80% fast or faster guards, the percentage of the guard position being performance bottleneck reduces to under 50%.

Another concern when increasing bandwidth threshold for guards is the diversity of paths will decrease. We use *Shannon entropy* as the diversity metric. Bauer *et al.*[1] adopt Shannon entropy as their definition of anonymity and diversity of Tor's path selection. There is obviously a trade-off between better diversity of guard position and higher threshold for guards. Table 5.2 shows that cutting the slowest 30% guards results in a 5.77% reduce in entropy over guard position, which we consider is acceptable considering the 34.3% increasement in bandwidth.

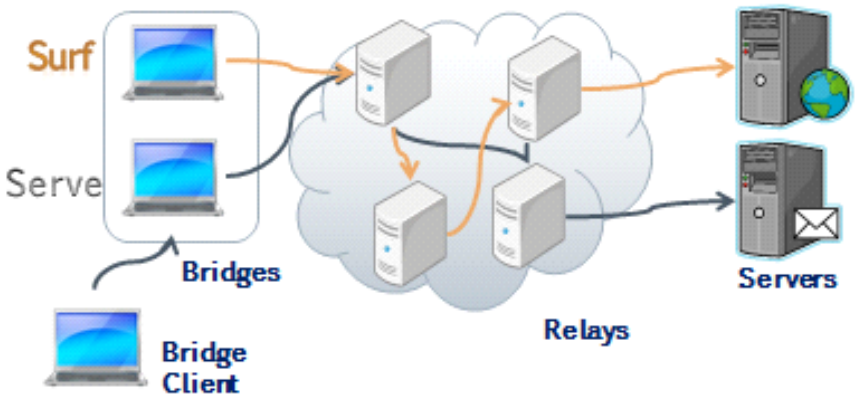


Figure 5.5: The two states of bridges: surfing and serving.

However, we do not conclude that bridge guards should be the top 70% fast guards. Our point is that through this analysis, we consider the current entry guard selection, evaluation and rotation mechanisms may not be suitable for bridge guards. Bridge guards need to be more sophisticated, which is considered as a future avenue for research.

5.2.2 Separating Bridge-guards and Client-guards

Since bridges are donated by volunteers, they may also use Tor to surf the Internet as a client. We distinguish two states of a bridge node in the same way as described by McLachlan and Hopper[21]. A bridge is *surfing* whenever it is relaying streams over Tor that originate from its own operator. A bridge is *serving* whenever it accepts connections from third party bridge clients and relays those connections over Tor. Figure 5.5 shows the two states of bridges: surfing and serving. When a bridge node is surfing, it chooses its first hop from its guard list. The problem comes if we let bridges using bridge guards.

In Section 5.1.2, it is not specified whether bridges should use the same list of guard relays for extending paths for client when they serve, as they use as entry guards for their own paths when they surf.

One of the advantages of keep separate guard lists is to prevent the bridge client of the bridge from being able to enumerate the guards that the bridge uses to protect its own traffic, by extending a path through bridge to a relay it controls. Having separate guard lists can also prevent the attack proposed by McLachlan and Hopper[21]. However, having separate guard lists would probably make the two kinds of circuits

more easily distinguishable.

It is not clear whether separating bridge-guards and client-guards is a good idea or not. We hope our discussions could encourage future research on this aspect.

Chapter 6 Conclusions

Tor is the most popular anonymous communication system in the world. Nowadays, there is an increasing set of Tor users use it as a tool for Internet censorship resistance. Its bridge mechanism is designed to help censored users to access the Tor network. Meanwhile, serious threats against Tor bridge mechanism exist. In this research, we have proposed stronger bridge mechanisms, discussed their concrete designs and analyzed their influences.

We first investigate the vulnerabilities of the current bridge mechanism under four different adversarial models. Existing works indicate these attacks are threats to Tor's bridge mechanism. However, as stated in Section 1.4, some of them just conduct research under one of the adversarial models, and do experiments on the real Tor network, which may harm the anonymity and privacy of real Tor users. Others just mention there might be a threat against Tor bridge mechanism but without further works. This work is the first one, which thoroughly summarizes and quantitatively researches into all these aforementioned adversarial models. We do experiments through simulation, which does not harm the anonymity and privacy of real Tor users.

We have developed a Tor path simulator to do experiments, which is the technical base of this research. This simulator accepts the existing historical bridge descriptors and bridge network statuses as input. It simulates not only Tor paths, but also Tor bridge clients. Because of this simulator's extensibility, it can also be utilized in other researches. We have used it to conduct a research on attacks against the Tor network (Publication <1>), and a research on Tor's entry guard mechanism (Publication <2>). It can also be utilized to simulate other Tor-related phenomena that do not involve actual network traffic. Examples include analysis of client path diversity, research on Tor's path selection algorithm, and assessing whole-network effects of heterogeneous client configurations.

Then we compare the current bridge mechanism with our two proposals. Experiment results show that our proposal of sticking to the first bridge (top one method) can effectively mitigate the attacks under *malicious bridge model* and *bridge set fingerprinting model*. This proposal is easy to implement and proved not to negatively affect the performance, so we consider it practical for the top one method to be adopted in Tor. As described in Section 4.3, 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 an open research question.

Our proposals, however, cannot mitigate the enumeration attack against Tor bridges. To counteract the enumeration attack of Tor bridges, we evaluate two anti-enumeration defenses, which implement *bridge guards* in different methods. Although the idea of bridge guard has been discussed in Tor community, its effectiveness has not yet been quantitatively verified and evaluated. This part of our work fills the gap. We also discusses the concrete design. We discover that letting client choose guards may have the risk of increasingly exposing bridges, while letting bridge have guards can effectively mitigate the enumeration attack. After that, we do further discussions on bridge guards about using fast guards and whether to separate bridge-guards and client-guards. As mentioned in Section 5.2, through analyses, we discover the current entry guard selection, evaluation and rotation mechanisms may not be suitable for bridge guards. Bridge guards need to be more sophisticated, which is considered as a future avenue of research. What's more, an investigation into whether to separate bridge-guards and client-guards or not is also worth conducting.

Tor bridges are designed to help censored users all over the world, so researches on Tor bridges can have great social significance. This work contributes to propose stronger Tor bridge mechanism, which is expected to encourage related researches on both anonymous communication system Tor and Internet censorship resistance technologies.

Acknowledgements

本研究を遂行するにあたり、日頃から常にご指導を頂きました東京大学生産技術研究所の松浦幹太教授に心から感謝致します。松浦先生には研究の進め方だけでなく、研究に対する姿勢や着眼点を含む基礎的な部分から教えて頂き、また学会参加など多数の各種活動の機会を与えて頂いたことで、研究生 1 年間と修士 2 年間を非常に有意義に過ごすことができました。

松浦研究室打ち合わせでの発表において様々な的確な助言をくださったり、研究内容に関して議論していただいた東京大学の山口利恵先生、元警察庁の北條孝佳さん、警察庁情報通信局情報技術解析課の田村研輔さん、The Karlsruhe Institute of Technology の Andreas Gutmann さん、中央大学研究開発機構の北川隆さんを始めとする、今まで研究生 1 年間と修士 2 年間の間にお世話になった松浦研究室打ち合わせの参加者の皆様に感謝致します。

そして、私たちの研究活動が円滑に進むように日頃から尽力してくださっている教授室秘書の小倉華代子さん、仲野小絵さんにも改めて深く感謝致します。

また、松浦研の技術職員である細井琢朗さん、松浦研究室メンバーの村上隆夫さん、Bongkot Jenjarrussakul さん、大畑幸矢さん、石坂理人さん、横手健一さん、碓井利宣さん、高木哲平さん、中田謙二郎さん、篠田詩織さん、孫達さんにも、研究室や松浦研定期ミーティングにおいて議論をしたり、適切な助言をいただきました。改めて感謝致します。皆様のおかげで松浦研究室での 3 年間は素晴らしいものとなりました。

そのほか、留学生活において大変お世話になりましたチューターの大畑幸矢さん、パナソニックスカラシップの関係者の皆様にも深く御礼申し上げます。お陰様で日常生活のことで困らずに研究に没頭できました。

最後に、遠くにいながらも、常日頃から私を支えてくれた両親に心から感謝します。

Bibliography

[1] Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. "Low-resource Routing Attacks Against Tor". In: *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society*. WPES '07. Alexandria, Virginia, USA: ACM, 2007, pp. 11–20.

[2] Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. "Denial of Service or Denial of Security?" In: *Proceedings of the 14th ACM Conference on Computer and Communications Security*. CCS '07. Alexandria, Virginia, USA: ACM, 2007, pp. 92–102.

[3] *Bridge DB*.
`https://bridges.torproject.org/`, Accessed December 2014.

[4] *CollecTor*.
`https://collector.torproject.org/index.html`, Accessed December 2014.

[5] Claudia Díaz, Stefaan Seys, Joris Claessens, and Bart Preneel. "Towards Measuring Anonymity". In: *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies*. PET '02. San Francisco, CA, USA: Springer-Verlag, 2003, pp. 54–68.

[6] Roger Dingledine and Nick Mathewson. *Design of a blocking-resistant anonymity system.* Tech. rep. `https://www.torproject.org/svn/trunk/doc/design-paper/blocking.html`, Accessed December 2014. Nov. 2006.

[7] Roger Dingledine, Nick Mathewson, and Paul Syverson. "Tor: The Second-generation Onion Router". In: *Proceedings of the 13th Conference on USENIX Security Symposium*. Vol. 13. SSYM '04. San Diego, CA: USENIX, 2004, pp. 303–320.

[8] Roger Dingledine, Nicholas Hopper, George Kadianakis, and Nick Mathewson. "One Fast Guard for Life (or 9 months)". In: *Proceedings of the 7th Workshop on Hot Topics in Privacy Enhancing Technologies*. HotPETs 2014. Amsterdam, Netherlands, July 2014.

[9] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. "Protocol Misidentification Made Easy with Format-transforming Encryption". In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*. CCS '13. Berlin, Germany: ACM, 2013, pp. 61–72.

[10] Matthew Edman and Paul Syverson. "As-awareness in Tor Path Selection". In: *Proceedings of the 16th ACM Conference on Computer and Communications Security*. CCS '09. Chicago, Illinois, USA: ACM, 2009, pp. 380–389.

[11] Nick Feamster and Roger Dingledine. "Location Diversity in Anonymity Networks". In: *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*. WPES '04. Washington DC, USA: ACM, 2004, pp. 66–76.

[12] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. "Hiding Routing Information". In: *Proceedings of Information Hiding: First International Workshop*. Springer-Verlag, LNCS 1174, May 1996, pp. 137–150.

[13] Andrew Hintz. "Fingerprinting Websites Using Traffic Analysis". In: *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies*. PET '02. San Francisco, CA, USA: Springer-Verlag, 2002, pp. 171–178.

[14] Nicholas Hopper, Eugene Y. Vasserman, and Eric Chan-Tin. "How Much Anonymity Does Network Latency Leak?" In: *ACM Transactions on Information and System Security* 13.2 (Mar. 2010), pp. 1–28.

[15] *Improving Tor's anonymity by changing guard parameters.* `https://blog.torproject.org/blog/improving-tors-anonymity-changing-guard-parameters`, Accessed December 2014.

[16] *Iran blocks Tor; Tor releases same-day fix.* `https://blog.torproject.org/blog/iran-blocks-tor-tor-releases-same-day-fix`, Accessed December 2014.

[17] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. "Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries". In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*. CCS '13. Berlin, Germany: ACM, 2013, pp. 337–348.

[18] George Kadianakis. *Implications of switching to a single guard node: some conclusions.* `https://lists.torproject.org/pipermail/tor-dev/2014-March/006458.html`, Accessed December 2014.

[19] Zhen Ling, Xinwen Fu, Wei Yu, Junzhou Luo, and Ming Yang. "Extensive Analysis and Large-Scale Empirical Evaluation of Tor Bridge Discovery". In: *Proceedings of the 31th IEEE International Conference on Computer Communications*. INFOCOM '12. Orlando, FL, USA: IEEE, Mar. 2012, pp. 2381–2389.

[20] Nick Mathewson. *Bridge Guards and other anti-enumeration defenses.* `https://www.torproject.org/svn/trunk/doc/design-paper/blocking.html`, Accessed December 2014. Oct. 2011.

[21] Jon McLachlan and Nicholas Hopper. "On the Risks of Serving Whenever You Surf: Vulnerabilities in Tor's Blocking Resistance Design". In: *Proceedings of the 8th ACM Workshop on Privacy in the Electronic Society*. WPES '09. Chicago, Illinois, USA: ACM, 2009, pp. 31–40.

[22] Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. "SkypeMorph: Protocol Obfuscation for Tor Bridges". In: *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. CCS '12. Raleigh, North Carolina, USA: ACM, 2012, pp. 97–108.

[23] Steven J. Murdoch and Robert N. M. Watson. "Metrics for Security and Performance in Low-Latency Anonymity Networks". In: *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies*. PET '08. Leuven, Belgium: Springer-Verlag, July 2008, pp. 115–132.

[24] Steven J. Murdoch and Piotr Zieliński. "Sampled Traffic Analysis by Internet-exchange-level Adversaries". In: *Proceedings of the 7th International Conference on Privacy Enhancing Technologies*. PET '07. Ottawa, Canada: Springer-Verlag, 2007, pp. 167–183.

[25] Lasse Øverlier and Paul Syverson. "Locating Hidden Servers". In: *Proceedings of the 2006 IEEE Symposium on Security and Privacy*. SP '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 100–114.

[26] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. "Website Fingerprinting in Onion Routing Based Anonymization Networks". In: *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*. WPES '11. Chicago, Illinois, USA: ACM, 2011, pp. 103–114.

[27] Andreas Pfitzmann and Marit Köhntopp. "Anonymity, Unobservability, and Pseudeonymity: a Proposal for Terminology". In: *International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*. Berkeley, California, USA: Springer-Verlag, 2001, pp. 1–9.

[28] Jon Postel. *RFC 791: Internet Protocol Dapra Internet Program Protocol Specification*.
`ftp://ftp.rfc-editor.org/in-notes/rfc791.txt`. Sept. 1981.

[29] Michael G. Reed, Paul F. Syverson, and David M. Goldschlag. *Onion routing network for securely moving data through communication networks*.
`http://www.google.com/patents/US6266704`. July 2001.

[30] Andrei Serjantov and George Danezis. "Towards an Information Theoretic Metric for Anonymity". In: *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies*. PET '02. San Francisco, CA, USA: Springer-Verlag, 2003, pp. 41–53.

[31] Yi Shi and Kanta Matsuura. "Fingerprinting Attack on the Tor Anonymity System". In: *Information and Communications Security*. Berlin, Heidelberg: Springer-Verlag, LNCS 5927, 2009, pp. 425–438.

[32] Rob Smits, Divam Jain, Sarah Pidcock, Ian Goldberg, and Urs Hengartner. "BridgeSPA: Improving Tor Bridges with Single Packet Authorization". In: *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*. WPES '11. Chicago, Illinois, USA: ACM, 2011, pp. 93–102.

[33] Robin Snader and Nikita Borisov. "A Tune-up for Tor: Improving Security and Performance in the Tor Network". In: *Proceedings of the Network and Distributed Security Symposium*. NDSS '08. Internet Society, Feb. 2008.

[34] *Stem*.
`https://stem.torproject.org/index.html`, Accessed December 2014.

[35] *The lifecycle of a new relay*.
`https://blog.torproject.org/blog/lifecycle-of-a-new-relay`, Accessed December 2014.

[36] *Tor Bridge Specification*.
`https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=attic/bridges-spec.txt`, Accessed December 2014.

[37] *Tor bridgedb*.
`https://gitweb.torproject.org/bridgedb.git/tree`, Accessed December 2014.

[38] *Tor Directory Protocol*.
`https://gitweb.torproject.org/torspec.git/blob/HEAD:/dir-spec.txt`, Accessed December 2014.

[39] *Tor Path Specification*.
`https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=path-spec.txt`, Accessed December 2014.

[40] Eugene Vasserman, Rob Jansen, James Tyra, Nicholas Hopper, and Yongdae Kim. "Membership-concealing Overlay Networks". In: *Proceedings of the 16th ACM Conference on Computer and Communications Security*. CCS '09. Chicago, Illinois, USA: ACM, pp. 390–399.

[41] Zachary Weinberg, Jeffrey Wang, Vinod Yegneswaran, Linda Briesemeister, Steven Cheung, Frank Wang, and Dan Boneh. "StegoTorus: A Camouflage Proxy for the Tor Anonymity System". In: *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. CCS '12. Raleigh, North Carolina, USA: ACM, 2012, pp. 109–120.

[42] Tim Wilde. *Great Firewall Tor Probing Circa 09 DEC 2011*. `https://gist.github.com/da3c7a9af01d74cd7de7`, Accessed December 2014.

[43] Philipp Winter and Stefan Lindskog. "How the Great Firewall of China is Blocking Tor". In: *Proceedings of the 2nd USENIX Workshop on Free and Open Communications on the Internet*. FOCI '12. Bellevue, WA: USENIX, 2012.

[44] Matthew Wright, Micah Adler, Brian Levine, and Clay Shields. "Defending anonymous communications against passive logging attacks". In: *Proceedings of the 2003 IEEE Symposium on Security and Privacy*. SP '03. Washington, DC, USA: IEEE Computer Society, May 2003, pp. 28–41.

[45] Matthew Wright, Micah Adler, Brian Levine, and Clay Shields. "The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems". In: *ACM Transactions on Information and System Security* 7.4 (Nov. 2004), pp. 489–522.

Publications

Domestic Conference

<1> 馮菲, 松浦幹太. Tor ネットワークに対する戦略的攻撃とその脅威の検証, Towards an Analysis of a Strategic Attack against the Tor Network, コンピュータセキュリティシンポジウム 2013 論文集, CD-ROM, 高松，10 月，2013 年.

<2> 馮菲, 松浦幹太. Towards Better Parameters of Tor's Entry Guard Mechanism, 暗号と情報セキュリティ・シンポジウム (SCIS2014), CD-ROM, 鹿児島，1 月，2014 年.

<3> 馮菲, 松浦幹太. 網羅的な攻撃者モデルを考慮した Tor ブリッジ機構の強化, Stronger Bridge Mechanisms of Tor Considering Exhaustive Adversarial Models, コンピュータセキュリティシンポジウム 2014 論文集, CD-ROM, 札幌，10 月，2014 年.
（CSS2014 優秀論文賞受賞）.

<4> 馮菲, 松浦幹太. Evaluation of Anti-enumeration Defenses for Tor Bridges, 暗号と情報セキュリティ・シンポジウム (SCIS2015), CD-ROM, 北九州，1 月，2015 年.

Journal

<5> Fei Feng, Kanta Matsuura. Stronger Bridge Mechanisms of Tor Considering Exhaustive Adversarial Models. 情報処理学会論文誌,「社会に浸透していくコンピュータセキュリティ技術」特集. (Recommended paper. Submitted).