TorBrix: Blocking-Resistant Tor Bridge Distribution

Abstract

Tor is currently the most popular network for anonymous Internet communication. It critically relies on volunteer nodes called *bridges* to relay Internet traffic when a user's ISP blocks connections to Tor. Unfortunately, current methods for distributing bridges are vulnerable to malicious users who obtain and block bridge addresses. In this paper, we propose TorBrix, a protocol for privacy-preserving distribution of Tor bridges to n users, even when an unknown number $t < n$ of these users are controlled by a malicious adversary. TorBrix distributes $O(t \log n)$ bridges and guarantees that all honest users can connect to Tor with high probability after $O(\log t)$ rounds of communication with the distributor. Our empirical evaluations show that TorBrix requires at least 20x fewer bridges and two orders of magnitude less running time than the state-of-the-art.

1 Introduction

Mass surveillance and censorship increasingly threaten democracy and freedom of speech. A growing number of governments around the world restrict access to the Internet to protect their domestic political, social, financial, and security interests [31, 27]. Countering this trend is the rise of anonymous communication systems, which strive to foil censorship and preserve the anonymity of individuals in cyberspace. Tor [14] is the most popular of such systems with more than 2.5 million users on average per day [2]. Tor relays Internet traffic via more than 6,500 volunteer nodes called *relays* spread across the world [3]. By routing data through random paths in the network, Tor can protect the private information of its users such as their identity, geographical location, and content accessed.

Since the list of all relays is publicly available, governments can block access to them. When access to Tor is blocked, users can use *bridges*, which are volunteer relays not listed in Tor's public directory [13]. Bridges serve only as entry points into the rest of the Tor network, and their addresses are carefully distributed to the users, with the hope that they will not be learned by censors. As of March 2016, about 3,000 bridge nodes were running daily in the Tor network [5].

Currently, bridges are distributed to users based on strategies such as CAPTCHA-enabled email-based distribution [13]. Unfortunately, censors now use sophisticated attacks to obtain and block bridges, rendering Tor unavailable for many users [12, 22, 33]. Additionally,

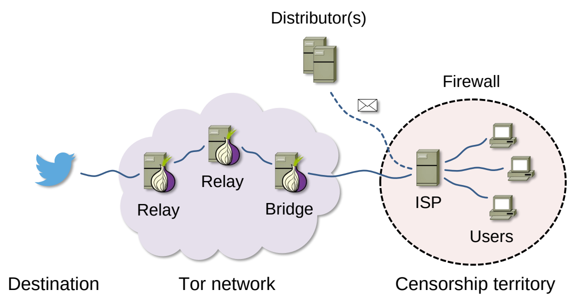


Figure 1: Our network model

current techniques for bridge distribution either (1) cannot provably guarantee that all *honest*[1] users can access Tor [32, 24, 30]; (2) only work when the number of dishonest users is known in advance [23]; (3) require fully trusted distributors [24, 23, 30]; and/or (4) cannot resist malicious attacks from the distributors [32, 24, 23, 30].

In this paper, we describe TorBrix, a bridge distribution protocol that guarantees Tor access to all honest users with high probability, even when there is an unknown number of corrupt users that can block access to all bridges they receive. TorBrix distributes $O(t \log n)$ bridges, where n is the total number of users, and $t < n$ is the number of corrupt users. This significantly improves over prior work by Mahdian [23], which distributes $O(t^2 \log n / \log \log n)$ bridges, and also requires knowledge of t in advance.

Additionally, TorBrix uses secure multi-party computation protocols to ensure that distributors do not learn user-bridge assignments, even when up to a $1/3$ fraction of the distributors are controlled by an adversary. Finally, we stress that TorBrix can run independently from Tor so that the Tor network can focus on its primary purpose of providing anonymity.

The rest of this paper is organized as follows. In Section 1.1, we describe our network and threat model. In Section 1.2, we state our main result as a theorem. We review related work in Section 2. In Section 3, we describe our protocol for reliable bridge distribution; we start from a basic protocol and improve it as we continue. We describe our implementation of TorBrix and our simulation results in Section 5. Finally, we summarize and state our open problems in Section 6.

[1]By honest users, we mean the users that are not controlled by the censor to obtain the bridge addresses assigned to them.

1.1 Our Model

We now define our problem model, which is depicted, at a high-level, in Figure 1.

We assume there are *n users* who want to obtain bridge addresses to access Tor. Initially, we assume a single trusted server called the *distributor*, which has access to a reliable supply of bridge addresses. Later, we generalize to multiple distributors.

We assume an adversary (or *censor*) which can control up to t of the users. We call these adversarially-controlled users *corrupt*. The adversary is *adaptive* in that it can corrupt users at any point of the protocol, up to the point of taking over t users. The adversary has the ability to *block* any bridges received by any of the corrupt users. He is not required to block bridges immediately upon receipt, but may rather strategically decide the best time to block a bridge.

Users which are not corrupt are called *honest*. Each honest user seeks to obtain one bridge that is not blocked.

We make the standard assumption that there exists a rate-limited channel, such as email, that allows users to send requests for bridges to the distributor, and the distributor to send bridges to the users.[2] The distributor runs our bridge distribution protocol locally and sends bridge assignments back to the users via the same channel. We assume the adversary has no knowledge of the private random bits used by our protocol.

Bridge Reachability. We assume the distributor learns which bridges are blocked using scanning algorithms deployed outside the censored countries. Efficient scanning algorithms are described in recent work by Dingledine [11], Ensafi et al. [17], and Burnett and Feamster [9].

1.2 Our Result

Below is our main theorem, which we prove in Section 3.

Theorem 1. *There exists a bridge distribution protocol that can run among m distributors and guarantee the following properties with probability $1-1/n^c$, for some constant $c \geq 1$, in the presence of a malicious adversary corrupting at most $\lfloor m/3 \rfloor$ of the distributors:*

1. *The number of bridges distributed is $O(t \log n)$;*
2. *All honest users can connect to Tor after $\lceil \log \lceil (t+1)/32 \rceil \rceil + 1$ rounds of communication with the distributors;*
3. *Each user receives m messages in each round;*
4. *Each distributor sends/receives $O(m^2+n)$ messages;*
5. *Each message has length $O(\log n)$ bits.*

[2]Completely blocking a service such as email would likely impose significant economic consequences for censors. However, unfortunately, email alone does not enable real-time interaction with the Web.

We simulate a proof-of-concept prototype of TorBrix to measure the running time and bridge cost of the protocol. We discuss our simulation results in Section 5.

2 Related Work

Proxy Distribution. The bridge distribution problem has been studied under the name *proxy distribution*, where a set of proxy servers outside a censorship territory are distributed among a set of users inside the territory.

The work closest to our own is that of Mahdian [23]. To the best of our knowledge, this is the only other result that gives theoretical guarantees against an omniscient adversary. Mahdian's work assumes that the number of corrupt users, t, is known in advance. His algorithms may use up to $O(t^2 \log n / \log \log n)$ bridges.

Remaining related work on proxy distribution uses three main approaches. First, proof-of-work based schemes, including the system of Feamster et al. [18]. Second, social networks based schemes, including the Kaleidoscope system of Sovran et al. [30], and the Proximax system of McCoy et al. [24]. Finally, reputation based schemes, including the rBridge system proposed by Wang et al. [32]. Our approach is essentially orthogonal to these schemes, in that proof-of-work, social networks, and reputation management can potentially be heuristically incorporated into the TorBrix system.

Handling DPI and Active Probing. The Tor Project has developed a variety of tools known as *pluggable transports* [4] to obfuscate the traffic transmitted between users and bridges. This makes it hard for the censor to perform *deep packet inspection (DPI)* attacks, since distinguishing actual Tor traffic from legitimate-looking obfuscated traffic is hard.

The censor can also block bridges using *active probing* [16]: he can passively monitor the network for suspicious traffic, and then actively probe dubious servers to block those identified as running the Tor protocol. Depending on the sophistication of the censor, TorBrix may be used in parallel with tools that can handle DPI and active probing to provide further protection against blocking.

Resource-Competitive Analysis. Our analytical approach to bridge distribution can be seen as an application of the *resource-competitive analysis* introduced by Gilbert et al. [19, 6], which measures the performance of a system with respect to the unknown resource budget of an adversary.

3 Our Protocol

We first construct a bridge distribution protocol that is run locally by a trusted distributor. Then, we extend this protocol to multiple distributors, where no subset of less than a $1/3$ fraction of the distributors learns any information about the user–bridge assignments.

We say an event occurs *with high probability*, if it occurs with probability at least $1 - 1/n^c$, for some constant $c \geq 1$. We denote the set of integers $\{1,...,n\}$ by $[n]$, the natural logarithm of any real number x by $\ln x$, and the logarithm to the base 2 of x by $\log x$. We denote a set of n users participating in our protocol by $\{u_1,...,u_n\}$. We define the *latency* of our protocol as the maximum number of rounds of communication that any user has to perform with the distributor(s) until he obtains at least one unblocked bridge.

3.1 Basic Protocol

The most naive approach to distribute a set of bridges is to assign a unique bridge to each user. Unfortunately, this does not scale: while the number of Tor users has nearly tripled in the past two years [1], the number of bridges in the network has at best remained the same [5].

Thus, TorBrix assigns each bridge to multiple users. In particular, we start with a "small" set of bridges, and assign each user a bridge selected uniformly at random from this set. But how do we choose the size of this set? If the set is too small, an adversary can corrupt a small number of bridges and easily prevent any users from accessing Tor. If the set is too large, then we are wasting precious bridges.

The key idea is to *adjust* the number of bridges distributed in each round based on the number of bridges that have been blocked. Our protocol is divided into rounds incremented by i. We advance to the next round when the number of bridges blocked in the current round (b_i) exceeds a geometrically increasing threshold. In each round, we increase geometrically the size of the set of bridges that we assign (this size is the value d_i). In this way, we ensure that the number of bridges TorBrix uses is a slowly growing function of the number of bridges blocked.

The number of bridges distributed in every round is determined based on the threshold in that round as depicted in Figure 2. The exponential growth of the number of bridges distributed in each round allows us to achieve a logarithmic latency (in t) until all users can connect to Tor with high probability (see Lemma 2). In Lemma 1, we show that if one instance of steps 1–13 of Algorithm 1 is executed, then it guarantees that all users can connect to Tor with some *constant probability*. Therefore, if we run $3\log n$ instances in parallel, we can guarantee that all users connect to Tor *with high probability*.

3.2 Some Modifications

Reusing Bridges. In every round, TorBrix only distributes unblocked bridges. A heuristic to reduce the total number of bridges required is to use unblocked bridges from previous rounds in the current distribution round. This can be done by removing blocked bridges from the pile of previously used bridges and adding a sufficient number of new bridges to accommodate the new load. One may choose to further reduce the number of bridges used by assigning new bridges only to those users who still do not have an unblocked bridge.

Algorithm 1 TorBrix – Basic Protocol

Goal: Distributes a set of $O(t \log n)$ bridges among a set of users $\{u_1,...,u_n\}$.

```
Run $3\log n$ instances of the following algorithm in parallel with disjoint sets of bridges:
1:  $i \leftarrow 1$
2:  while true do
3:      $d_i \leftarrow 2^{i+4}$
4:      $\{B_1,...,B_{d_i}\} \leftarrow d_i$ unblocked bridges
5:      for each $j \in [n]$ do          $\triangleright$ Distribute $d_i$ bridges
6:          Pick $k \in [d_i]$ uniformly at random
7:          Send bridge $B_k$ to user $u_j$
8:      end for
9:      while $b_i < 0.6 \times 2^{i+4}$ do
10:         $b_i \leftarrow$ # blocked bridges in $\{B_1,...,B_{d_i}\}$
11:     end while
12:     $i \leftarrow i+1$
13: end while
```

Handling Serialization Attacks. If the $3\log n$ instances run completely independently, then the adversary can take advantage of this to increase the latency of the algorithm by a factor of $3\log n$ using a *serialization attack*. In this attack, the adversary can strategically coordinate with its corrupt users to block the assigned bridges in such a way that the instances proceed to the next round one at a time. TorBrix prevents this attack by maintaining a single round counter, i, for all instances: whenever the number of blocked bridges in *any* of the instances exceeds the threshold for the current round, all instances are taken to the next round. Since all instances are run by the distributor locally, i can be easily synchronized between them.

Handling User Churn. Algorithm 1 can only distribute bridges among a fixed set of users. A more realistic scenario is when users join or leave the algorithm frequently. One way to handle this is to add the new users to the algorithm at the beginning of the next round (i.e., after i increments). This, however, introduces two challenges. First, the adversary can arbitrarily delay the next round, causing a denial of service attack. Second, our proof of robustness (Lemma 1) would not necessarily hold if n changes, because the algorithm is repeated $3\log n$ times

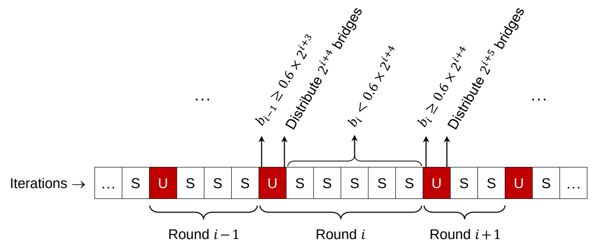

…
$b_{i-1} \geq 0.6 \times 2^{i+3}$
Distribute 2^{i+4} bridges
$b_i < 0.6 \times 2^{i+4}$
$b_i \geq 0.6 \times 2^{i+4}$
Distribute 2^{i+5} bridges
…
Iterations →
… S U S S S U S S S S S U S S U S …
Round $i-1$
Round i
Round $i+1$

Figure 2: Number of bridges distributed in round i of Algorithm 1. S and U indicate successful and unsuccessful iteration of the while-loop in Algorithm 1. An iteration is called successful when *all* users are able to connect to Tor in that iteration. Otherwise, it is called an unsuccessful iteration.

to ensure it succeeds with high probability.

To resolve these challenges, TorBrix can assign $3\log n$ random bridges from the set of bridges used in the last round (i.e., the last time i was incremented) to every new user. If the total number of users, n, is doubled since the last round, we use $3 \times 2^{i+4}$ unblocked bridges and assign 3 of them randomly to each user. This ensures that the number of parallel instances always remains $3\log n$ even if n changes, because $\log n$ is increased by one when n is doubled. Therefore, each existing user must receive 3 new bridges so that Lemma 1 holds in the setting with churn. Our remaining lemmas hold if users leave the system; thus, we only need to update n when nodes leave.

Since distributing new bridges among existing users is done only after the number of users is doubled, the latency is increased by at most a $\log n$ term, where n is the largest number of users in the system during a complete run of the algorithm.

3.3 Privacy-Preserving Bridge Distribution

We now consider a *multiple distributors model*, where a group of $m \ll n$ distributors collectively distribute bridge addresses among the users.

We assume that the distributors are connected to each other pairwise with private and authenticated channels. In this model, the adversary not only can corrupt an unknown number of the users, t but can also maliciously control and read the internal state of up to $\lfloor m/3 \rfloor$ of the distributors. The corrupt distributors can deviate from our protocols in any arbitrary manner, e.g., by sending invalid messages or remaining silent.

The key tool to handle such a scenario is *secure multiparty computation*. In this problem, the goal is to compute a function over private inputs distributed over many nodes, even when up to a $1/3$ fraction of the nodes are controlled by an adversary, and to do so without revealing any information about the private input held by any node. Seminal work by Goldreich, Micali and Wigderson [20] described a protocol to solve secure multiparty computation for any function. Recent results have improved on this seminal work in terms of bandwidth and latency costs, and practicality [8, 10, 7] (see also [21, 28] for surveys).

We can use secure multiparty computation to solve our multiple distributors problem in the following manner. Initially, each bridge address is divided into m shares that are given to each distributors in such a way that 1) no subset of less than a $1/3$ fraction of the distributors can learn the bridge address by sharing their shares; and 2) any subset of a $2/3$ fraction of the distributors can reconstruct the bridge address with their shares. Standard approaches using Shamir secret sharing [29] and Reed-Solomon codes [26] can achieve this.

Next, we use any secure multiparty computation protocol to essentially compute the function in Algorithm 1. In particular, after running this secure multiparty computation, for each user, each distributor learns a share of the appropriate bridge to be sent to that user, and sends that share to the user. In this way, 1) no coalition of less than a third of the distributors will learn which bridges map to which users; and 2) all users will receive enough correct shares to reconstruct the bridges assigned to them.

4 Protocol Analysis

We now prove Theorem 1. Before stating our first lemma, we define the following variables:

- b_i: number of bridges blocked in round i.
- d_i: number of bridges distributed in round i.
- t_i: number of corrupt users that have blocked at least one bridge in round i.

Lemma 1 (**Robustness**). *In round i of Algorithm 1, if $b_i < 0.6 \times 2^{i+4}$, then all honest users can connect to Tor with high probability.*

Proof. We first consider the execution of only one of the $3\log n$ instances of Algorithm 1. For each user, the algorithm chooses a bridge independently and uniformly at random and assigns it to the user. Without loss of generality, assume the corrupt users are assigned bridges first.

For $k = 1, 2, ..., t_i$, let $\{X_k\}$ be a sequence of random variables each representing the bridge assigned to the k-th corrupt user. Also, let Y be a random variable corresponding to the number of *bad* bridges, i.e., the bridges that are assigned to at least one corrupt user that has blocked a bridge in this round. Since each user is assigned a fixed bridge with probability $1/d_i$, the probability that a bridge is assigned to at least one such corrupt user is $1-(1-1/d_i)^{t_i}$. Thus, by linearity of expectation,

$$E[Y] = \left(1-(1-1/d_i)^{t_i}\right) d_i < (1-e^{-(t_i+1)/d_i}) d_i.$$

We know $t_i < 2^{i+4}$, because in each round $d_i = 2^{i+4}$ bridges are distributed and each corrupt user is assigned exactly one bridge. Hence,

$$\mathrm{E}[Y] < (1 - 1/e^{1+1/2^{i+4}})d_i \le (1 - 1/e^2)d_i \qquad (1)$$

Therefore, in expectation at most a constant fraction of the bridges become bad in each instance of the algorithm.

The sequence $\{Z_k = \mathrm{E}[Y|X_1,...,X_k]\}$ defines a Doob martingale [15, Chapter 5], where $Z_0 = \mathrm{E}[Y]$. Since $|Z_{k+1} - Z_k| \le 1$, $Z_0 = \mathrm{E}[Y]$, and $Z_{t_i} = Y$, by the Azuma-Hoeffding inequality [15, Theorem 5.2],

$$\Pr(Y > \mathrm{E}[Y] + \sqrt{d_i}) \le e^{-2d_i/t_i} < 1/e^2. \qquad (2)$$

The last step holds since $t_i < d_i$. Hence, with probability $1 - 1/e^2$, any user is assigned a bad bridge with probability at most.

$$\begin{aligned}\frac{\mathrm{E}[Y] + \sqrt{d_i}}{d_i} &< \frac{(1 - 1/e^{1+1/2^{i+4}})d_i + \sqrt{d_i}}{d_i} \\ &= 1/e^{1+1/2^{i+4}} + 1/\sqrt{d_i}, \qquad (3)\end{aligned}$$

where the first step is achieved using (1).

Now, let $p_1 = \Pr(Y > \mathrm{E}[Y] + \sqrt{d_i})$, and let p_2 be the probability that a fixed honest user is assigned a bad bridge in a fixed instance and a fixed round. From (2) and (3), we have

$$p_1 < 1/e^2 \quad \text{and} \quad p_2 < 1/e^{1+1/2^{i+4}} + 1/\sqrt{d_i}.$$

Thus, the probability that a fixed user fails to receive a good bridge in a fixed instance and a fixed round is equal to $p_1 + (1 - p_1)p_2$, which is at most 0.6.

Over the $3\log n$ instances, the probability that a user only receives bad bridges is at most $0.6^{\lceil 3\log n\rceil} \le 1/n^2$. By a union bound, the probability that any of the n users receives only bad bridges in a round is at most $1/n$. □

Lemma 2 (**Latency**). *By running Algorithm 1, all honest users can connect to Tor with high probability after at most* $\lceil \log \lceil (t+1)/32 \rceil \rceil + 1$ *iterations of the while loop.*

Proof. Fix one of the parallel instances of Steps 1–13 of Algorithm 1. The adversary must block at least $0.6 \times 2^{i+4}$ bridges in round i to force the algorithm to proceed to the next round. Let ℓ be the smallest integer such that $2^\ell \ge t$. In round ℓ, the adversary has enough corrupt users to take the algorithm to round $\ell + 1$. However, in round $\ell + 1$, the adversary can block at most $2^\ell < 2^{\ell+1}$ bridges. Thus, by Lemma 1, at the end of round $\ell + 1$, *all* honest users can connect to Tor with high probability. Since $2^\ell \ge t$, and the algorithm starts by distributing 32 bridges, $\ell + 1 \le \lceil \log \lceil (t+1)/32 \rceil \rceil + 1$. □

Lemma 3 (**Bridge Cost**). *The total number of bridges used by Algorithm 1 is at most* $\min[(10t + 96)\log n, 2n]$.

Proof. Consider one of the $3\log n$ instances of Algorithm 1. The algorithm starts by distributing 32 bridges. In every round $i > 0$, the algorithm distributes a new bridge only to replace a bridge blocked in round $i - 1$. Let M_i be the total number of bridges used until round i; and let a_i be the number of new bridges distributed in round i. Then,

$$M_i = a_i + \sum_{j=0}^{i-1} b_j. \qquad (4)$$

In round i, $a_i \le 2^{i+4}$ and $b_i < 0.6 \times 2^{i+4}$. Thus,

$$M_i < 2^{i+4} + 0.6 \sum_{j=1}^{i-1} 2^{j+4} = 9.6(2^i - 2) + 2^{i+4}.$$

From Lemma 2, it is sufficient to run the algorithm $k = \lceil \log \lceil (t+1)/32 \rceil \rceil + 1$ rounds. Then,

$$M_k < 9.6(2^k - 2) + 2^{k+4} \le 3.2t + 32.$$

Summing over all $3\log n$ instances, we get that the total number of bridges is at most $(10t + 96)\log n$. If, when the number of bridges to be distributed in the current round across all instances becomes larger than n, the algorithm sends one bridge to each user, we get that the total number bridges used is at most $\min[(10t + 96)\log n, 2n]$. □

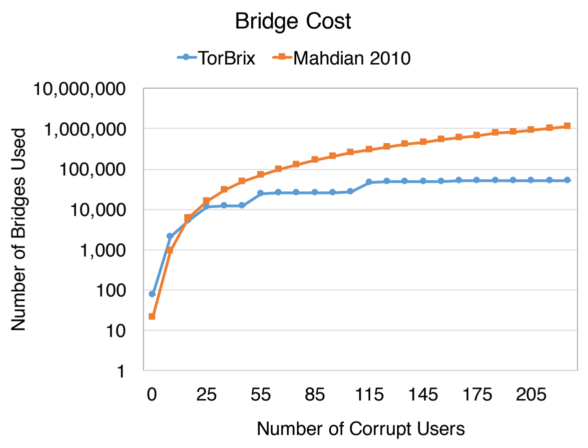
Algorithm 1 does not necessarily assign the same number of users to each bridge. However, in the following lemma, we show that each bridge is assigned to almost the same number of users as other bridges with high probability providing a reasonable level of load-balancing.

Lemma 4 (**Bridge Load-Balancing**). *Let X be a random variable representing the maximum number of users assigned to any bridge, Y be a random variable representing the minimum number of users assigned to any bridge,* $\mu = n/d_i$ *be the average number of users per bridge, and* $z = \Theta\left(\frac{\ln n}{\ln\ln n}\right)$. *Then, we have*

$$\Pr(X \ge \mu z) \le 2/n \quad \textit{and} \quad \Pr(Y \le \mu z) \le 2/n.$$

Proof. Each round of Algorithm 1 can be seen as the classic balls-and-bins process: n balls (users) are thrown independently and uniformly at random into d_i bins (bridges). Then it is well known that the distribution of the number of balls in a bin is approximately Poisson with mean $\mu = n/d_i$ [25, Chapter 5].

Let X_j be the random variable corresponding to the number of users assigned to the j-th bridge, and let $\tilde{X}_j$ be the Poisson random variable approximating X_j. We have $\mu = \mathrm{E}[X_j] = \mathrm{E}[\tilde{X}_j] = n/d_i$. We use the following Chernoff bounds from [25, Chapter 5] for Poisson random



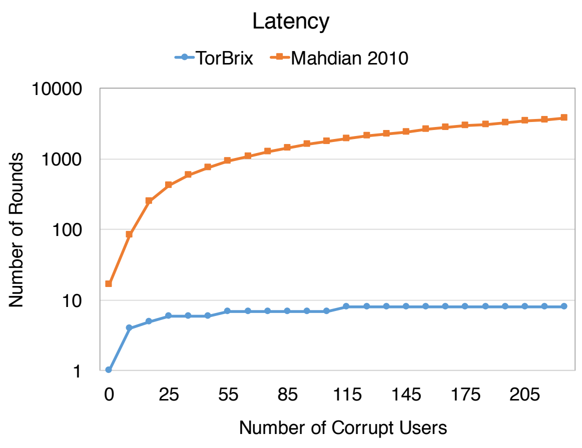


Figure 3: Simulation results for $n = 8,192$ and variable number of corrupt users showing bridge cost (left) and latency (right) of TorBrix and the dynamic bridge distribution algorithm of Mahdian [23].

variables:

$$\Pr(\tilde{X}_j \geq x) \leq e^{-\mu}(e\mu/x)^x, \text{ when } x > \mu \qquad (5)$$
$$\Pr(\tilde{X}_j \leq x) \leq e^{-\mu}(e\mu/x)^x, \text{ when } x < \mu \qquad (6)$$

Let $x = \mu y$, where $y = ez$. From (5), we have

$$\Pr(\tilde{X}_j \geq \mu y) \leq \left(\frac{e^{y-1}}{y^y}\right)^{\mu}$$
$$\leq \frac{e^{y-1}}{y^y} = \frac{1}{e}\left(\frac{1}{z^z}\right)^e < \frac{1}{n^2}. \qquad (7)$$

The second step is because $y^y > e^{y-1}$ (since $z > 1$) and $\mu > 1$. The last step is because $z = \Theta\left(\frac{\ln n}{\ln \ln n}\right)$ is the solution of $z^z = n$.

It is shown in [25, Corollary 5.11] that for any event that is monotone in the number of balls, if the event occurs with probability at most p in the Poisson approximation, then it occurs with probability at most $2p$ in the exact case. Since the maximum and minimum bridge loads are both monotonically increasing in the number of users, from (7) we have

$$\Pr(X_j \geq \mu y) \leq 2\Pr(\tilde{X}_j \geq \mu y) < 2/n^2.$$

By applying a union bound over all bridges, the probability that the number of users assigned to any bridge will be more than μz is at most $2/n$. The bound on the minimum load can be shown using inequality (6) in a similar way. □

5 Evaluation

We implemented a proof-of-concept prototype of TorBrix and tested it in a simulated environment with 8,192 users and one distributor. We assume the adversary blocks bridges aggressively meaning that it blocks every bridge it receives immediately. Experiments with other adversarial strategies gave similar results to those presented here."We also implemented the dynamic bridge distribution algorithm of Mahdian [23] to compare with TorBrix. We set the parameters of both protocols in such a way that it fails with probability at most 10^{-5}. We increase the number of corrupted users, t, from 0 to 225 with increments of 10 and measure the bridge cost and latency of both schemes. The results are shown in Figure 3. All numbers are averaged over 10 runs of each protocol.

Figure 3 shows that the number of bridges used by TorBrix is less than those used by Mahdian's algorithm when $t \geq 15$. Moreover, this number scales significantly better for TorBrix than Mahdian's algorithm for larger values of t. For example, TorBrix requires 20 times fewer bridges for $t = 200$. Figure 3 also shows it takes TorBrix significantly less time to guarantee robust bridge assignments. For example, TorBrix requires at least two orders of magnitude less time than [23] to guarantee every user receives at least one unblocked bridge. In the TorBrix plots, sharp increases can be seen around $t = 5, 10, 25, 55$, and 115 that are due to the increase in the number of rounds required until the protocol converges.

6 Conclusion

We described TorBrix, a bridge distribution system that allows all honest users to connect to Tor in the presence of an adversary that corrupts an unknown number of users, and can hide the bridge assignments from a colluding adversary. Empirical evaluations show that TorBrix requires at least 20x fewer bridges and two orders of magnitude less running time than the state-of-the-art. As a future work, the current algorithm uses a relatively large number of bridges when the number of corrupt users is large. Is it possible to make the bridge cost sublinear in t with practical constant terms?

References

[1] TorMetrics: Directly connecting users. Available at `https://metrics.torproject.org/userstats-relay-country.html`.

[2] The Tor Project metrics: Direct users connecting between January 1, 2015 and March 31, 2015, 2015.

[3] The Tor Project metrics: Relays in the network between January 1, 2015 and March 31, 2015, 2015.

[4] The Tor Project: Pluggable transport, 2015.

[5] The Tor Project metrics: Bridges in the network between March 1, 2016 and March 31, 2016, 2016.

[6] M. A. Bender, J. T. Fineman, M. Movahedi, J. Saia, V. Dani, S. Gilbert, S. Pettie, and M. Young. Resource-competitive algorithms. *ACM SIGACT News*, 46(3):57–71, Sept. 2015.

[7] P. Bogetoft, D. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. Nielsen, J. Nielsen, K. Nielsen, J. Pagter, et al. Secure multiparty computation goes live. *Financial Cryptography and Data Security*, pages 325–343, 2009.

[8] E. Boyle, K.-M. Chung, and R. Pass. Large-scale secure computation: Multi-party computation for (parallel) ram programs. In *Annual Cryptology Conference*, pages 742–762. Springer, 2015.

[9] S. Burnett and N. Feamster. Encore: Lightweight measurement of web censorship with cross-origin requests. *SIGCOMM Comput. Commun. Rev.*, 45(4):653–667, Aug. 2015.

[10] V. Dani, V. King, M. Movahedi, and J. Saia. Quorums quicken queries: Efficient asynchronous secure multiparty computation. In *Distributed Computing and Networking*, volume 8314 of *Lecture Notes in Computer Science*, pages 242–256. Springer Berlin Heidelberg, 2014.

[11] R. Dingledine. Research problem: Five ways to test bridge reachability, 2011.

[12] R. Dingledine. Research problems: Ten ways to discover Tor bridges, 2011.

[13] R. Dingledine and N. Mathewson. Design of a blocking-resistant anonymity system. Technical report, The Tor Project Inc., 2006.

[14] R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, Berkeley, CA, USA, 2004.

[15] D. P. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 2009.

[16] R. Ensafi, D. Fifield, P. Winter, N. Feamster, N. Weaver, and V. Paxson. Examining how the Great Firewall discovers hidden circumvention servers. In *Internet Measurement Conference (IMC)*. ACM, 2015.

[17] R. Ensafi, J. Knockel, G. Alexander, and J. R. Crandall. Detecting intentional packet drops on the Internet via TCP/IP side channels. In *Proceedings of the 15th International Conference on Passive and Active Measurement - Volume 8362*, PAM 2014, pages 109–118, New York, NY, USA, 2014. Springer-Verlag New York, Inc.

[18] N. Feamster, M. Balazinska, W. Wang, H. Balakrishnan, and D. Karger. Thwarting web censorship with untrusted messenger discovery. In R. Dingledine, editor, *Privacy Enhancing Technologies*, volume 2760 of *Lecture Notes in Computer Science*, pages 125–140. Springer Berlin Heidelberg, 2003.

[19] S. Gilbert, J. Saia, V. King, and M. Young. Resource-competitive analysis: A new perspective on attack-resistant distributed computing. In *Proceedings of the 8th International Workshop on Foundations of Mobile Computing*, FOMC '12, pages 1:1–1:6, New York, NY, USA, 2012. ACM.

[20] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, STOC '87, pages 218–229, New York, NY, USA, 1987. ACM.

[21] Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality*, 1(1):5, 2009.

[22] Z. Ling, J. Luo, W. Yu, M. Yang, and X. Fu. Extensive analysis and large-scale empirical evaluation of tor bridge discovery. In *INFOCOM, 2012 Proceedings IEEE*, pages 2381–2389, March 2012.

[23] M. Mahdian. Fighting censorship with algorithms. In P. Boldi and L. Gargano, editors, *Fun with Algorithms*, volume 6099 of *Lecture Notes in Computer Science*, pages 296–306. Springer Berlin Heidelberg, 2010.

[24] D. McCoy, J. A. Morales, and K. Levchenko. Proximax: Measurement-driven proxy dissemination.

In *Proceedings of the 15th International Conference on Financial Cryptography and Data Security*, FC'11, pages 260–267, Berlin, Heidelberg, 2012. Springer-Verlag.

[25] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[26] I. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics (SIAM)*, pages 300–304, 1960.

[27] D. Rushe. Google reports 'alarming' rise in censorship by governments. The Guardian, June 2012.

[28] J. Saia and M. Zamani. Recent results in scalable multi-party computation. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 24–44. Springer, 2015.

[29] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[30] Y. Sovran, A. Libonati, and J. Li. Pass it on: Social networks stymie censors. In *Proceedings of the 7th International Conference on Peer-to-peer Systems*, IPTPS'08, pages 3–3, Berkeley, CA, USA, 2008. USENIX Association.

[31] K. Turner. Mass surveillance silences minority opinions, according to study. The Washington Post, March 2016.

[32] Q. Wang, Z. Lin, N. Borisov, and N. Hopper. rbridge: User reputation based tor bridge distribution with privacy preservation. In *Network and Distributed System Security Symposium*, NDSS 2013. The Internet Society, 2013.

[33] P. Winter and S. Lindskog. How the great firewall of China is blocking Tor. In *2nd USENIX Workshop on Free and Open Communications on the Internet*, Berkeley, CA, 2012.