Troll Patrol: Anonymous User Reporting of Bridge Censorship

Vecna
University of Waterloo
Waterloo, Ontario, Canada
vvecna@uwaterloo.ca

Ian Goldberg
University of Waterloo
Waterloo, Ontario, Canada
iang@uwaterloo.ca

Abstract

Bridges are circumvention proxies that provide routes around censorship. In order to be effective, most bridges must be kept secret from censors, lest they be added to censors' blocklists.

Reputation-based bridge distribution systems including rBridge, Hyphae, and Lox have been proposed with the goal of enabling regular users to learn about bridges while preventing censors from learning about and blocking these bridges. These three examples use anonymous credentials to preserve user anonymity while still allowing the anonymous users to gain reputation within the system if the bridges distributed to them remain unblocked for some period of time and penalizing them by reducing their reputations if the bridges distributed to them become blocked by censors. Performing these reputation changes requires these systems to have knowledge of which bridges have been discovered and blocked by censors and which are still accessible.

The obvious way to test whether a given bridge is accessible from a given region is by attempting to connect directly to that bridge from within that region; however, this approach carries risks, including that these scans may inadvertently reveal previously undiscovered bridges to a censor. Rather than actively scanning all bridges, we favor soliciting users to submit reports when their bridges are inaccessible and scanning only the bridges for which user reports have been received, to validate those reports. This approach reduces the set of bridges that must be actively scanned but also introduces new risks if implemented naïvely; namely, censors or malicious users might submit inaccurate reports in order to disrupt the bridge distribution system or induce additional scans. It might be tempting to tie users' reports to their identities in order to penalize users who submit inaccurate reports, but doing so would violate user anonymity. Building on the anonymous credential systems used in, e.g., Hyphae and Lox, we design Troll Patrol, a scheme for users to submit reports that is resilient against malicious behavior, both by censors and by regular users, while still preserving the anonymity of the users who submit them.

Keywords

measuring censorship, bridges, anonymous credentials, Tor, Lox

1 Introduction

Bridges [17], first designed for use with Tor [18], are proxies intended to enable people to circumvent censorship and access otherwise restricted resources. While not all types of bridges must remain secret from censors, most bridges rely on secrecy. Unless access to a bridge is entangled with access to other resources the censor is unwilling to block (as with bridges such as meek that use domain fronting [26]), a censor that discovers the bridge can simply block access to it. This work is specifically concerned with settings in which these bridges must remain secret.

When a user is given information about how to connect to a bridge, we say the bridge has been *distributed* to the user. We refer to a system for distributing bridges to users as a *bridge distribution system*, and we call the server component of such a system (which knows the information needed to connect to the bridges being distributed and decides which bridges to distribute to which users) the *bridge distributor*. Distributing bridges to honest users while preventing censors from learning about and blocking those bridges is a challenge that is currently best addressed by *reputation-based* bridge distribution systems. In these systems, users learn about bridges and then earn higher or lower reputations over time based on whether or not the bridges distributed to them become blocked by censors. If a user learns information that then makes its way to a censor, this suggests that the user may be cooperating with the censor. rBridge [80], Hyphae [40], and Lox [75] operate on this basic principle, while using anonymous credentials [3, 10, 52] to protect user anonymity.

Of course, it is not possible to enact these changes in reputation level without insight into which bridges have been blocked and which have not. Unfortunately, gaining this insight is not as simple as it may seem. While measuring censorship is a well-studied problem [23, 28, 49, 56, 79, 87], the vast majority of prior work is concerned with learning whether or not *publicly known* resources (such as news websites) are accessible. The problem of detecting bridge censorship is subtly more challenging because it carries an additional risk that must be mitigated: the censorship measurements could themselves expose the identities of bridges to censors observing the measurements.

A number of sources of data on whether a bridge can be accessed from a region have been discussed [15, 39, 66, 67]. Generally, these ideas fall into three categories: active scans (attempting or inducing a connection between a device in the region and the bridge), bridge usage statistics (having the bridge estimate and report the number of distinct users in the region that connect to it), and user reports (having users in the region report whether or not they are able to connect to the bridge). Active scans carry various risks. If they are performed by volunteers, a censor may retaliate against those volunteers, and the process of actively scanning bridges could reveal those secret bridges to a censor. Thus, we aim to limit the number of active scans that must be performed by only scanning bridges for which we have some other indication of possible censorship.

In this work, we explore the other two categories as indicators of which bridges might be blocked (and thus should be actively

Proceedings on Privacy Enhancing Technologies 2026(4), 964–982

https://doi.org/10.56553/popets-2026-0154

scanned for verification), with the goal of reducing the number of active scans that must be performed. Using real-world Tor data, we evaluate the utility of bridge usage statistics with the use case of reputation-based bridge distribution systems in mind. We expect user reports to be useful in principle, but we consider that there may be possible attacks by censors and malicious users, as well as privacy risks to honest users who submit reports. To address these risks, we present Troll Patrol,[1] an abuse-resistant reporting scheme that uses anonymous credentials to allow users to anonymously submit reports that bridges are inaccessible.

Our contributions are as follows:

- We discuss bridge-reported usage statistics, positive reports (in which users report that they *are* able to access their bridges), and negative reports (in which users report that they *are not* able to access their bridges) as possible early indicators of censorship and ultimately argue that negative reports are the only one of these three indicators likely to be useful for bridges distributed by Lox or similar reputation-based bridge distribution systems.
- Given a bridge distribution system that uses anonymous credentials (such as Lox) and a separate mechanism for actively testing whether a bridge is accessible in a region (which we leave to future work), we describe a scheme to allow users to submit reports that bridges are inaccessible, such that only the bridges reported as inaccessible must be actively tested. Our reporting system preserves user anonymity while protecting against malicious behavior from both censors and regular users. In particular, a report that a bridge is inaccessible cannot be submitted by someone who has not been given information about how to access the bridge, and we (privately) track the number of inaccurate reports submitted by users, allowing us to disallow reports from users who have previously submitted too many inaccurate reports.
- We implement a prototype of our reporting scheme as an extension to the Tor Project's implementation of Lox and evaluate its performance, finding that our new reporting protocols perform comparably to existing Lox protocols and that our changes have only a small impact on the performance of existing Lox protocols.

2 Assumptions and Threat Model

In this section, we discuss the assumptions that must be true in order for Troll Patrol to be effective, as well as our threat model.

2.1 Assumptions

Our assumptions address important problems that must be considered when implementing our proposed reporting system in practice. We do not prescribe the specifics of how these assumptions should be fulfilled, but we do briefly discuss implementation considerations.

Assumption 1. *Users are in definable regions, and a user's client can determine in which region to consider itself.*

[1]The Norwegian fairy tale "Three Billy Goats Gruff" features a troll who lives under a bridge and eats would-be crossers of the bridge. Our work mobilizes users to "patrol" to find such "trolls" that prevent people from using bridges.

We model users in regions, which could be as broad as entire countries or as granular as individual autonomous systems. Our goal is to determine whether users in a region are able to access particular resources. For the sake of report submission, a user's client must be able to indicate from which region its connection attempts succeeded or failed. This region information could be determined through some automated procedure or through manual input from the user.

Assumption 2. *Users can anonymously contact the bridge distributor, even if their secret bridges are inaccessible.*

We assume that there is some (possibly high-latency or low-throughput) bi-directional *signaling channel* that can be used to send small amounts of data between a user and the bridge distributor, even if the user's secret bridges have all been discovered and blocked. We note that such a signaling channel is already a prerequisite for the bridge distribution system, where users who do not already have working bridges must have some way to learn about bridges and can subsequently use their bridges to establish a low-latency, high-throughput *messaging channel*. Various censorship-resistant signaling channels have been proposed, such as email as in SWEET [90], domain fronting [26], and Amazon Simple Queue Service [55].

Furthermore, we assume that this signaling channel can be used in a way that protects user anonymity when contacting the bridge distributor. If a user's secret bridges are all inaccessible, they might submit their reports over an anonymity network, connecting via a different type of bridge that does not require secrecy (but may have low bandwidth). As a concrete example, users might connect to Tor via meek, which uses domain fronting but imposes a limit on bandwidth due to costs associated with running the bridge [44]. This mechanism may be accessible to users and usable as a signaling channel but too slow for regular Internet use.

Assumption 3. *It is possible to perform active scans to confirm whether or not bridges are accessible from any region of interest.*

This paper focuses on early indicators to infer which bridges may be inaccessible and assumes that there is some separate way to verify these inferences, namely through active scans. Developing such a verification system that can be used in all cases is challenging; we discuss logistics of performing these scans in Section 7.2 but do not ultimately provide a solution to this problem. Such a system will need to be separately developed for the scheme presented in this paper to be deployable. Without scanning infrastructure, we can still make inferences about which bridges are blocked (for example, based on reports from users); however, we cannot verify these inferences (for example, to determine whether the users who submitted reports for a bridge should be allowed to continue submitting reports).

2.2 Threat Model

We consider a censor to be a powerful adversary with control over a region's networks. In particular, a censor can block connections based on defined rules, and it can make exceptions in these rules for its own connections to succeed without issue. While censors are technically powerful, they also face social and economic pressures.

We assume that a censor will allow *some* benign-looking communications, even if these communications could be used to circumvent censorship intended by the censor. Complete Internet shutdowns do occur; for example, Iran has recently blocked nearly all Internet use in response to war [12] and to suppress information about the country's human rights violations during protests [4, 81]. However, these extreme restrictions of Internet use are outside our threat model and better addressed by other work [35, 37, 54, 58].

We consider that a censor may attempt to learn information about bridges in order to block those bridges. It may identify bridges by requesting them from the bridge distributor or receiving an invitation from an existing user (as a regular user would do), by identifying bridge users or the infrastructure used to test access to bridges and watching the resources they attempt to contact, or by fingerprinting the protocols used to carry traffic between bridges and users. If a censor blocks a bridge, it may attempt to hide this fact from the bridge distributor. Inversely, if a censor cannot block a bridge because it has not discovered the bridge's identity, it may nonetheless attempt to mislead the bridge distributor into believing that the bridge is inaccessible in order to cause disruptions.

Importantly, we note that a bridge may be inaccessible due to censorship or simply offline, e.g., for maintenance. Users of reputation-based bridge distribution systems should lose trust when censors block their bridges, but they should not be penalized for their bridges being offline. In either case, we aim to learn that the bridge is inaccessible, but it is important that we also learn *why*: whether the bridge is blocked in a region or offline.

While we leave the method of verifying early inferences to future work, direct scans (in which some device within the censored region attempts a direct connection to the bridge) are an obvious direction. If the device used for direct scans is known to the censor, then the censor can monitor connections from that scanner, potentially learning additional bridges that the scanner also tests. Thus, we aim to keep the identity of such a scanner secret from the censor. However, we assume that the censor may still discover the scanner and that its probability of doing so is some monotonically increasing function of the number of scans performed. (It may be the case, for example, that each scan has some chance of drawing the attention of the censor or that the censor will notice the scanner if its volume of scans exceeds some threshold.) Thus, we aim to minimize the number of scans that must be performed, to limit the chance of the censor discovering the scanning infrastructure.

Beyond disrupting bridge use, a censor may attempt to identify bridge users and their friends. A censor may attempt measures such as intercepting reports submitted by users or compromising the bridge distributor to learn this information.

We assume that the bridge distributor is not itself acting as or colluding with a censor; however, we do not expect users to assume that the bridge distributor is not curious about their identities and social relationships or that it will never be compromised by someone who is. Thus, we consider that the bridge distributor may attempt to identify users or learn their social graphs; i.e., which users are friends with which other users. Bridge distribution systems such as Hyphae and Lox use anonymous credentials to protect users' identities and social graphs from the bridge distributor; our reporting scheme must not violate these protections.

We also consider that users may themselves act with malice (or have compromised devices that behave maliciously without user intent). Malicious behavior by users (or their devices) may be viewed as a reduced version of that of a censor. Users cannot generally restrict other users' access to bridges (though they may be able to inform censors of bridges to be blocked), but they may still claim that bridges are accessible when they are not or vice versa. In particular, users regularly reporting that their bridges are inaccessible may lead to their bridges being regularly scanned. As discussed, this creates additional risk, as increasing the number of scans increases the risk of the scanning infrastructure being discovered by the censor and may in turn expose additional bridges to the censor. Thus, if some users demonstrate a pattern of submitting inaccurate reports, we would like to ignore reports from them in the future, to reduce the number of scans and the associated risks.

Finally, we note that a user submitting a report that a bridge is inaccessible and a subsequent scan determining that the bridge is accessible from the user's region (or vice versa) do not necessarily imply malice. It is possible that the user's report was a genuine representation of the user's view and that the user's view is not appropriately representative of their assigned region. The user may have a particularly restrictive local firewall configured, or regions may be defined too broadly in general. For example, if regions are defined on a per-country basis but censorship is applied on a per-ISP basis (as is the case in India [36, 60], for example), one user in a country may be able to access resources that another user in the same country cannot. Thus, if a user submits reports that cannot be validated, we do not assume that user to be acting maliciously. Nevertheless, we may wish to ignore future reports from them.

3 Bridge Usage Statistics

One possible source of data on bridge accessibility is statistics sourced from bridge operators about the users of their bridges. For example, the Tor Project collects daily statistics about Tor bridges and publishes this information through the CollecTor [71] service. In the case of a Tor bridge, this information includes a `bridge-ips` field containing a list of the country codes from which the bridge saw a non-relay connection during the measured period and the number of unique IP addresses (rounded up to a multiple of 8) used to connect from each such country (based on GeoIP lookups in a local database).

A 2011 report [39] by Loesing proposed that a bridge may be considered blocked in a country if the count of connecting users from that country falls below an absolute threshold, below a relative threshold compared to other countries' connection counts, or outside of an estimated interval based on past data on connections to that bridge from that country. For simplicity, Loesing adopted an absolute threshold of 32, meaning that a bridge that receives 32 or fewer connections from a country in a day should be considered blocked, and evaluated bridge censorship in China in the first half of 2010. Loesing's report only evaluated bridges that reported receiving connections from at least 100 (due to rounding, this is 97 or more) different Chinese IP addresses during a single day in the evaluation period, noting that "[b]ridges with fewer users than that have a usage pattern that makes it much more difficult to detect blockings at all."

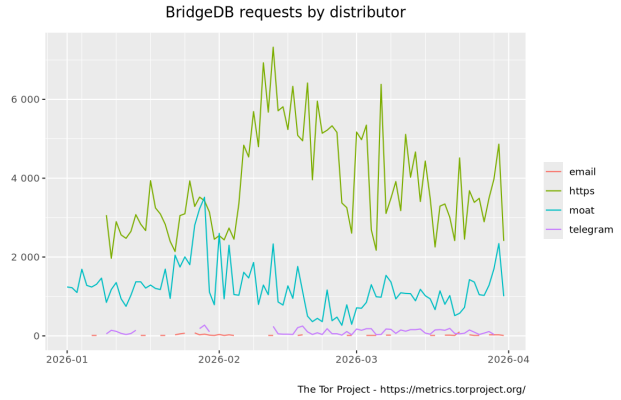

Figure 1: Approximate number of daily requests (by users in any country) for bridges from each distributor during the period January 2026 through March 2026. Graph from the Tor Project [69]. Gaps in the data appear to be due to a bug [45], rather than representing zero requests.

This is a fundamental limitation of using (absolutely or relatively) low connection counts to detect censorship: this approach requires a sufficient volume of consistent bridge use in order to reasonably use low bridge use as a signal for censorship. A bridge simply being unpopular or many users choosing not to use their bridges on a given day (e.g., a holiday) does not imply censorship. We also identify a second fundamental limitation: bridge connection counts can be deceptively increased by a censor that blocks regular users' access to the bridge but makes its own connections from several different IP addresses. Thus, a bridge's connection count exceeding the threshold does not imply the absence of censorship.

We explored the feasibility of using these bridge statistics to measure censorship, using a case study of Tor bridge censorship in Belarus in 2020–2021. In this incident, email-distributed bridges that used the obfs4 pluggable transport were blocked, while other bridges were apparently accessible. However, our attempts at developing algorithms to infer this censorship were unsuccessful, particularly because the bridges that were actually blocked were used by too few users in Belarus to meaningfully infer censorship from low connection counts, regularly receiving 0 connections from Belarus, even on days when they were *not* blocked. The details of this case study are provided in Appendix A.

It is unsurprising that the algorithms we used were not effective for detecting censorship of bridges with such low usage. If 0 connections is a common value during normal times (when the bridge is not blocked), it would be very difficult to detect censorship using a low connection count (even 0) on a single day. It may be possible to detect censorship based on an abnormally long period of low connection counts (e.g., if the bridge reports 0 connections for several consecutive days), but this would significantly increase the delay between a censorship event and detection of that event. In the context of bridge distribution, users having to wait (at minimum) a day for their bridges to be detected as blocked (and potentially replaced) is already a burden; increasing this delay to several days would not be reasonable.

It seems likely that the observed low regular use of bridges distributed via email is because it is more challenging and may feel more daunting for most users to request bridges via email than via HTTPS or Moat, even though users are more likely to have uncensored access to Gmail than to the bridges.torproject.org website. We note that this observation is consistent with current trends in global requests for bridges. Figure 1 shows that users are far more likely to request bridges via HTTPS or Moat than via email or the relatively new Telegram bot.

We similarly expect low regular connection counts for the bridges distributed by reputation-based systems, which have much higher barriers for access. Learning a bridge in one of these systems requires a prospective new user to know an existing user who trusts the prospective user enough to invite them or informally share bridges with them. Some systems, such as Salmon [19] and Lox [75], take a hybrid approach, in which bridges are initially distributed via some less-restrictive mechanism and later become invitation-only. Even with these hybrid systems, we expect each individual bridge to be distributed to a small number of users. For example, we note that the Tor Project's current implementation of Lox initially distributes each bridge to only 10 users [73]. These bridges eventually become aggregated into invitation-only "buckets" containing two other bridges (each also distributed to 10 users), meaning we expect 30 users *globally* (not necessarily in the same region) to know each bridge when it becomes invitation-only.

Motivated by this Belarus case study and the primary goal of supporting reputation-based bridge distribution systems (likely with a small user base for each bridge), we choose not to rely on bridge usage statistics, which (if low) do not reliably indicate censorship (particularly for low-use bridges) and (if high) do not reliably indicate lack of censorship (as they may have been manipulated by a censor to avoid detection). Instead, we assume that some active scans will be necessary, and we focus on designing a reporting system to indicate which bridges might be inaccessible so that only those bridges must be scanned. We discuss user reports in the next two sections.

4 Positive Reports

We use the phrase *positive report* to refer to a report indicating that a user has successfully connected to a bridge. We identify two ways that positive reports might be used. Tracking drops in the rate of positive reports over time could be used to infer censorship, or the presence of positive reports could be used as evidence that a bridge is accessible (counteracting evidence that it has been blocked, e.g., negative reports, to avoid false detections).

Unfortunately, positive reports have the same fundamental limitations as bridge usage statistics. Unless a bridge has a sufficiently high and consistent positive report submission rate (something we cannot assume for bridges distributed by reputation-based bridge distribution systems), it is not possible to reliably infer censorship based on a drop in positive report submissions. If positive reports can be used to counteract evidence of censorship, then censors (or malicious users) can abuse this by submitting positive reports even after a bridge has been blocked.

It may be tempting to add restrictions to positive reports to address the latter problem. However, censors have all the technical

capabilities of regular users. Restrictions will either be ineffective against a censor or effective against the censor but also overly restrictive of honest users. We discuss two strawman restrictions, explaining why both are insufficient. First, we could require positive reports for a bridge to include some evidence that the bridge was successfully contacted, such as a signed token fetched from the bridge. This would prevent users who cannot access the bridge from submitting positive reports maliciously. It would not, however, prevent a censor (which is presumed to have the power to allow its own connections to the bridge) from contacting the bridge and submitting positive reports. Second, given that we are building upon a reputation system, we could restrict positive report submission to users with sufficient reputation. This may be effective at preventing a censor from submitting positive reports, but it also prevents low-reputation-but-honest users from submitting these reports. As we expect most bridge users early on to have low reputations, this would significantly limit the utility of positive reports.

Based on these limitations, we do not include positive reports in our strategy for detecting blocked bridges.

5 Negative Reports

Drawing on ideas from prior work [15, 39, 66, 67], we have so far discussed active scans, bridge usage statistics, and positive reports as possible sources of data to indicate which bridges are accessible and which are not. We argued that active scans are necessary but dangerous, and their use should be limited, as well as that bridge usage statistics and positive reports are not useful for our setting. We thus design our system around *negative reports*, which indicate that a user was unable to connect to a bridge. We use these negative reports as an initial signal of which bridges might be blocked. The bridges reported as blocked should be actively scanned for validation.

Anecdotally, users informing Tor Project developers that their bridges are inaccessible has led to investigations confirming that the bridges are indeed blocked [7, 31]; our goal is to standardize this reporting process so that investigations (by way of active scans) can be automated and to build in abuse mitigations. We note that per Assumption 2, even if users' secret bridges are blocked, there is still some (possibly high-latency and low-bandwidth) anonymous channel for contacting the bridge distributor to submit reports.

While our design is not limited to Lox, we use Lox as our running example and include specific details for implementing negative reports for Lox when appropriate.

5.1 Design Goals

We have two broad goals when designing negative reports. First, these reports must not identify the users who submit them. Reports should reveal no more information about a user than the region from which they attempted a connection to a bridge and the list of bridges their credential permits them to use. We elaborate our assumptions related to this list of bridges in Section 5.2 but briefly note here that the bridge distributor should not know which users have been distributed which bridges and that each list should be assigned to multiple users such that it is not a unique fingerprint. We allow the bridge distributor to link multiple reports submitted by the same user within a period of time to each other but not to link this set of reports to any non-report-related user actions.

Second, there must be mechanisms for preventing abuse by censors and users. In particular, users must only be able to submit reports for bridges that were distributed to them, each user should only be able to submit one report per bridge at a time, and users who submit reports for bridges that are determined to be accessible from the user's region should lose the ability to submit reports in the future. As discussed in Section 2.2, we do not assume malice from submission of unverifiable reports. Thus, aside from losing the ability to submit more reports, users should not be penalized for submitting these unverifiable reports. Finally, a censor that intercepts a valid report should not learn any information that would allow it to block the bridge being reported.

5.1.1 A Simple Construction. We first provide a simple construction that achieves most of our described goals but does not provide a defense against users who submit inaccurate reports. In order to restrict the act of reporting a bridge to users who have knowledge of the bridge without requiring them to uniquely identify themselves, we require the report to include a hash of its *bridge line*: the information required to access the bridge (such as its IP:port pair and cryptographic material). This design prevents a censor that intercepts a report from learning the bridge line and using the information in it to block the bridge. To prevent replay attacks, we additionally require the hash input to include a nonce and the current date, and we reject reports with previously observed nonces or prior dates.

This scheme is simple and does not depend on any particular bridge distribution system; as such, it might be useful for bridges that are not distributed by reputation-based bridge distribution systems. However, it does not provide any defense against users who submit many reports for their bridges. Thus, a malicious user may submit inaccurate reports at every opportunity, leading to their bridges being regularly scanned, potentially drawing the attention of the censor to the scanning infrastructure and exposing additional bridges. Defending against this attack without violating user anonymity requires additional complexity; the rest of this section describes a reporting scheme that fulfills these goals.

5.2 Preliminaries

We assume that each user has an anonymous credential that they can use to prove statements about their status within the bridge distribution system. In particular, these credentials have various attributes, and a user can reveal some attributes and hide others while constructing a zero-knowledge proof about their credential's validity and its hidden attributes (in a process called a credential *show*). Among these attributes, we assume that each credential has a unique ID that is revealed during credential shows (but hidden during issuance such that the events of a credential issuance and a subsequent credential show cannot be linked together). This ID can be revealed only once, and when the user shows a credential with the ID revealed, they should be issued a new credential with a new (hidden) ID to replace their old one. The bridge distributor should reject credential shows using IDs that have already been revealed.

We also assume that the credential has an attribute that somehow conveys an ordered list of the bridges the user should be allowed

to access (and thus allowed to report as blocked) and that the list corresponding to this attribute is known to the bridge distributor. In Lox, this is the "bucket" attribute β, which consists of an index and a key, allowing the user to retrieve and decrypt their list of bridges. We will generically assume that the user has such an ordered list of bridges and refer to the attribute that conveys this list as β, but we do not prescribe the mechanism by which β conveys this list. We assume that multiple users are assigned the same list of bridges and that two users with identical lists of bridges have identical β values; in other words, having a particular β is not a unique fingerprint for a user. (We note that this is true of Lox.)

Troll Patrol works with either single-show or multi-show credential schemes (since we use the credentials in a single-show way), and it works with either public-verification or keyed-verification anonymous credentials (since we model the bridge distributor as both issuer and verifier). We note that the credential scheme used by Lox [10, 52] satisfies all of our requirements.

When the bridge distributor receives a valid report, it should first establish whether the bridge is online at all by attempting to connect to it. (We assume that the bridge distributor is itself located in a region that does not block bridges; thus, this connection attempt does not risk exposing the bridge's identity to a censor that will block it.) If the bridge distributor's connection fails, the bridge should be considered offline (not blocked). If the connection succeeds, then the bridge distributor should determine whether or not to initiate a scan based on some criteria, e.g., if a requisite number of reports for a particular bridge have been received from a particular region in a short period of time (which we envision in the range from 1 to 24 hours) and a recent scan has not already been initiated. As we expect bridges to be initially distributed to a small number of users, likely in different regions, a single report should initially be sufficient to initiate a scan. Once there is reason to believe that a bridge has been distributed to more users in the same region (e.g., after 44 days for Lox, enough time for a user to advance from level 0 to level 2, gaining the ability to invite friends), it may make sense to increase this threshold.

In the bridge distributor's view, reports are either *pending* or *resolved*. A report is resolved when the bridge specified in the report is found to be offline, when a scan has concluded that the bridge is either blocked or not blocked in the specified region, or when the requisite period of time has elapsed without a scan being initiated from that region. In the case that a bridge is scanned and found not to be blocked in a region, all pending reports for that bridge from that region are classified as *false reports*. Each anonymous credential tracks the number of false reports it has been used to submit, and we revoke the ability to submit more reports from credentials that have already been used to submit too many false reports. This process is described in Section 5.3.

5.3 Design

To facilitate reporting, we require credentials to contain the following attributes:

- ID: a unique ID, revealed only once
- β: a value that indicates the list of bridges the user is authorized to access
- f: the number of false reports the user has previously submitted
- p: the *pending* status of the credential

We discussed the need for a unique ID and the β attribute in Section 5.2. f and p are new attributes we introduce. Reporting occurs in sessions, which are initiated and resolved by the user. When a user does not have an active reporting session, their pending attribute $p = 0$. During a session, p has a random value defined by the bridge distributor, which is split into a prefix and a suffix. The suffix consists of the lowest k bits of p (we use $k = 8$ in our implementation), and it is used as a counter to track the number of operations that have occurred in the current session and ensure that the operations within the session are performed in an appropriate order. The prefix is randomly generated by the bridge distributor and used as a session identifier. This session identifier allows the bridge distributor to associate multiple reports submitted during a single reporting session but not to link multiple reporting sessions for a single user to each other or to link report-related actions by a user to non-report-related actions by the same user.

Notably, the bridge distributor and user must both know whether there are still pending reports associated with the user's session and how many resolved reports so far are false reports for which the user's credential's f value should increase. This information is certified by the bridge distributor and provided to the user in the form of a *resolve credential*, a new credential type that we define with the following attributes:

- p: a pending value, the same as in the main credential
- f_{new}: the number of new false reports for the user
- b: an indicator of which of the user's bridges have pending reports

Assuming the user's credential has a value β that conveys the ordered list of bridges known to the user, the ith bit of b (from least to most significant) is 1 if and only if the ith bridge in the user's list has a pending report. (Any bits that do not correspond to bridge indices are set to 0.) The user cannot end their reporting session if there are any pending reports associated with the session (in other words, if $b \neq 0$), and they cannot submit a report for a bridge if there is already a pending report for that bridge associated with their session or if their credential's f value plus f_{new} from their resolve credential exceeds an allowable false report limit, which we call m. Resolve credentials contain no information that is secret from the bridge distributor, and they are always shown and issued with all attributes revealed.

In general, other protocols can still be performed while a user's credential is in a pending (i.e., $p \neq 0$) state, and these protocols should be oblivious to the user's p value (while ensuring that it does not change). However, it may be necessary to explicitly disallow some actions while the user is in a pending state. For example, in Lox, a user can invite a friend, but the friend's credential should inherit some attributes from the first user's credential. We view f as an attribute that should be inherited, and we require the user to resolve their current session, aggregating all false reports from the session into their f count, before inviting a friend. This ensures that the friend's credential accounts for all false reports submitted by the user who invited them (thus limiting the ability of malicious users to invite themselves many times to obtain additional credentials

Table 1: Report-related protocols, including relevant inputs (the second section) and outputs (the third section). In each protocol, the user shows their old credential (denoted O) with some attributes revealed ($\mathcal{R}$) to the bridge distributor and some hidden ($\mathcal{H}$). The user is issued a new credential (denoted N) with some attributes selected ($\mathcal{S}$) by the bridge distributor, some revealed, and some hidden. The user may be required to show or be issued a resolve credential (denoted R); whenever a resolve credential is shown or issued, all of its attributes are always revealed. Constraints on inputs or outputs are listed after whether the attribute is selected by the bridge distributor, revealed, or hidden. Constraints on hidden attributes are proven in zero knowledge.

Protocol	$O.p$	$O.f$	R shown	$N.p$	$N.f$	R issued
Report Submit ($p = 0$)	$\mathcal{R} : p = 0$	$\mathcal{H} : f \leq m$		$\mathcal{S}$: low k bits $= 0$	$\mathcal{H} : f = O.f$	
Report Status	$\mathcal{R} : p \neq 0, p$ is even	$\mathcal{H}$		$\mathcal{R} : p = O.p + 1$	$\mathcal{H} : f = O.f$	Yes
Report Submit ($p \neq 0$)	$\mathcal{R} : p$ is odd	$\mathcal{H} : f + R.f_{new} \leq m$	Yes	$\mathcal{R} : p = O.p + 1$	$\mathcal{H} : f = O.f + R.f_{new}$	
Report Resolve ($b = 0$)	$\mathcal{R} : p$ is odd	$\mathcal{H}$	Yes: $b = 0$	$\mathcal{R} : p = 0$	$\mathcal{H} : f = O.f + R.f_{new}$	
Report Resolve ($b \neq 0$)	$\mathcal{R} : p$ is odd	$\mathcal{H}$	Yes: $b \neq 0$	$\mathcal{R} : p = O.p + 1$	$\mathcal{H} : f = O.f + R.f_{new}$	

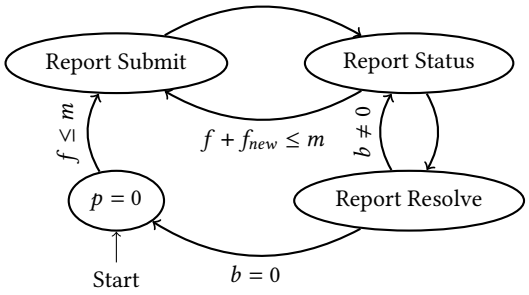

Figure 2: State transition diagram for our reporting protocols. $p = 0$ indicates that a user is not in a reporting session. $b = 0$ indicates that the user has no pending reports. In Report Status, the user is issued a resolve credential, which must be shown in the subsequent Report Submit/Resolve protocol. The user may perform at most α report-related protocols during a single reporting session (that is, between consecutive $p = 0$ states).

and submit many false reports). We also disallow operations that change a user's set of known bridges until their current reporting session has been resolved.

5.3.1 Protocols. Concretely, we introduce three protocols to manage reporting. For each, the user reveals their credential's ID and p value and is issued a new credential with its p value revealed and a new (hidden) ID. All other credential attributes (including f) are hidden but proven in zero knowledge to be the same for both the old and new credential or, in the case of f, increased appropriately. We summarize these protocols:

Report Submit. Report Submit allows the user to submit one or more reports. The user must indicate their region and the bridges being reported, and they must prove that they should be allowed to access those bridges. They do this by sending their β value times a publicly defined point H, such that the bridge distributor (which knows the set of β values in use) can determine the user's β value, but someone who intercepts the report cannot learn β from $\beta \cdot H$. The user also proves in zero knowledge that the β used to compute $\beta \cdot H$ is equal to the β attribute in their credential, and that their f value, plus f_{new} from their resolve credential if they are already in a pending state, does not exceed the allowable false report limit m.

Report Status. Report Status allows the user to obtain a resolve credential certifying the statuses of their previously submitted reports. They can use this credential to perform a subsequent Report Submit or Report Resolve action.

Report Resolve. Report Resolve allows the user to end their reporting session if $b = 0$ or simply resolve their current state, allowing them to perform Report Status again (to obtain an updated resolve credential for a future Report Submit or Report Resolve operation).

Interactions Between Protocols. Table 1 shows the inputs and outputs of each report-related protocol, and Figure 2 shows the permitted transitions between these protocols. When the bridge distributor assigns a random p value in the Report Submit protocol, it generates a random prefix and sets the k-bit suffix to 0 as described above. The bridge distributor uses the prefix as an index in a table and stores available information about the user's reporting session, such as any pending reports that have been submitted in the session and the user's region. When the user's session ends, the bridge distributor can (and should) delete this data.

Each report-related operation that occurs after a session begins and before the session ends causes the user's pending value p to increase by 1. This prevents the user from reusing old resolve credentials (with previous p values) and allows the bridge distributor to enforce restrictions on both the sequence and number of operations during a session. The user must always alternate between checking the status of their existing reports (at which time the bridge distributor commits to a particular state) and resolving the state to which the bridge distributor committed (either through the Report Resolve action or through submitting additional reports, which implicitly resolves their current state).

The bridge distributor should enforce some maximum number of actions within a session (including the initial Report Submit action that starts the session and the final Report Resolve action that ends the session), which we call α. This value must be odd, and it must not exceed $2^k + 1$ (where k is the number of bits in the suffix of p). In our prototype, we set α to 101 (supporting up to 50 pairs of Report Submit/Resolve and Report Status actions, followed by a final Report Resolve action that ends the session), but we do not expect that users will actually perform this many actions in a single session. A user's client should end sessions as quickly as possible,

only allowing them to extend if the user submits multiple reports at staggered times such that there is always a pending report.

A user is not permitted to end their session while they have any pending reports; however, we observe that as described it is possible for a user to reach an unresolvable state by performing $\alpha-2$ report-related actions in the same session and then performing Report Status only to receive a resolve credential with $b \neq 0$. In this case, the user is unable to perform the final Report Resolve action. This is not only a problem for submitting reports; being perpetually trapped in a pending state prevents the user from ever again performing any non-report-related protocols that are disallowed while $p \neq 0$. In Lox, for example, this would prevent the user from ever inviting friends or migrating to a new bucket when their current bridges are blocked. To address this, the bridge distributor can reject Report Status actions where the suffix of p is $\alpha - 2$ and there are still pending reports associated with the user's session (indicating to the client that it should try again later). Alternatively, the bridge distributor can make an exception to allow Report Resolve actions when $b \neq 0$ and the suffix of p is $\alpha - 1$ but treat each pending report as a false report.

We describe each new reporting protocol in further detail in the next section.

5.4 Protocol Details

In this section, we provide a more detailed description of our protocols for the submission of negative reports.

5.4.1 Report Submit. Report Submit requires either $p = 0$ or an odd p and a resolve credential with a matching p value. The user must indicate which bridges they are reporting and prove that they should be allowed to submit reports for the bridges being reported. We accomplish this using the β attribute, which is associated with an ordered list of bridges. To indicate which bridges are inaccessible, the user indicates which list of bridges they have (i.e., which value their credential has for the attribute β) and those bridges' indices within this list. To prove that the user should be allowed to use their credential to submit reports for those bridges, the user proves that their credential has the indicated β value and reveals either that their p value is 0 or that the b value in their submitted resolve credential has a 0 for the bit corresponding to each of the reported bridges.

Indicating β and proving that a credential has a particular β value can be trivially accomplished by simply revealing the β value, but we note that this attribute may itself be a sensitive value. In Lox, for example, a censor that learns a user's β value can learn and block all of that user's bridges. If it would be problematic for β to be learned by an adversary that observes the report, the user can instead submit a hash of β or multiply β by a public point. We use the latter approach in our implementation for Lox. We use a public point H, which is generated daily from a hash of the date. The user submits $D = \beta \cdot H$ and proves in zero knowledge that their credential contains this β. (Given knowledge of β, the bridge distributor can verify this statement in the proof. The fact that H changes each day prevents a passive observer from learning whether reports submitted on different days indicate the same or different β values without knowing the β values.) The bridge distributor knows the set of β values and precomputes all $\beta \cdot H$ values for each day, allowing it to quickly look up the β corresponding to the point D submitted by the user.

It is acceptable that the bridge distributor learns the user's β value in this process. Per Assumption 2, users can contact the bridge distributor anonymously; we assume that the user originally registered and learned their bridges anonymously (otherwise, the bridge distribution system has already failed at protecting user anonymity) and that reports are submitted anonymously. Thus, the bridge distributor does not know or learn which users know which β values. Additionally, as discussed in Section 5.2, we assume that β is not a unique fingerprint, so the user indicating β does not allow the server to associate all reporting sessions using that β value to the same (anonymous) user.

In addition to proving that their credential corresponds to the bridges being reported, the user must prove that they have not previously submitted too many false reports. The f value from the user's credential (which is not revealed) plus f_{new} from the resolve credential (if applicable) must not exceed the allowable false report limit m, and the sum is used as the f value in the user's new credential. (The user proves these statements in zero knowledge.) If the user's old credential had $p = 0$, the bridge distributor assigns a new, random p value with the low k bits equal to 0 (as described previously). Otherwise, the p value for the new credential is one more than the p value from the old credential. (Note that in either case, the new p value is even.) If the user has previously submitted reports within a session, they should not be allowed to submit reports from a different region within the same session. (While a user may change regions from time to time, they must resolve their session and start a new one before submitting additional reports in this case.)

5.4.2 Report Status. Report Status requires an even $p \neq 0$ and returns a resolve credential that summarizes the current status of the user's reports. The user's new credential is given a p value that is one higher than the p value of the user's previous credential (the new p value will be odd), and the issued resolve credential is given the same p value as the new credential. The purpose of incrementing p is to ensure that old resolve credentials cannot be reused in future operations.

Importantly, in issuing a resolve credential, the bridge distributor commits to a particular state. The user must perform a subsequent Report Submit or Report Resolve action, and the bridge distributor must accept this action according to the state committed to in the resolve credential, rather than a future state of the bridge distributor's view. For example, suppose a user has an unresolved report when they perform Report Status, so that the resulting resolve credential has $b \neq 0$. When they next perform Report Resolve, their reporting session will not end, *even if the outstanding report is resolved before they perform the Report Resolve protocol.* They must perform Report Resolve (presenting their resolve credential with $b \neq 0$), followed by another Report Status action (obtaining a resolve credential with $b = 0$ showing that their reports have all been resolved), and then a final Report Resolve action to end their session.

5.4.3 Report Resolve. Report Resolve requires an odd p and a resolve credential with the same p value. The user is issued a new credential, with f equal to the old credential's f plus f_{new} from the resolve credential. If the resolve credential showed that the user

had no pending reports (i.e., $b = 0$), then the user's new credential has $p = 0$, ending the user's reporting session. Otherwise, the p value for the user's new credential is one more than the p value from the old credential (in which case it will be even).

5.4.4 Storage. The bridge distributor must store some information about reporting sessions, namely the region associated with a reporting session (which is not permitted to change during the session) and information about the reports submitted as part of the session. In particular, the bridge distributor must track pending reports and resolved false reports that the user has not yet aggregated into their credential's f value. When a user performs Report Status and receives a resolve credential indicating the count of resolved false reports, the bridge distributor can stop tracking these resolved reports (however, it must continue to track all pending reports). The resolve credential tracks this information, and the user *must* present it during the next report-related protocol.

When the user successfully performs Report Resolve with $b = 0$, their reporting session ends. At this point, all of their submitted reports must have been resolved (otherwise b would not be 0), and the count of their new false reports must have been added to their credential's f value (otherwise Report Resolve would not succeed). At this time, the bridge distributor can purge all information related to that particular reporting session; the only information about the reporting session that is needed in the future is the user's total false report count, which is tracked by the user's credential (but authenticated by the bridge distributor such that the user cannot falsify it).

5.4.5 Allowing f to Reset or Decay. As described, f can only ever increase. However, this may be problematic in practice. In particular, "false" reports may be submitted for reasons unrelated to malice, such as a misconfiguration, an overly restrictive local firewall, or a temporary bridge outage that is resolved before the bridge distributor makes its connection to test whether the bridge is online. As such, it may be desirable to allow f to gradually decay over time, or to provide some mechanism for resetting f after it grows beyond m, allowing users to regain the ability to submit reports. We describe a mechanism for resetting f in Appendix B.

5.5 Privacy Leakage

We aim to leak as little additional information as possible, beyond what the underlying bridge distribution system leaks. In this section, we discuss the additional leakage of both our simpler reporting scheme described in Section 5.1.1 and our full scheme described in Sections 5.3 and 5.4. Per Assumption 2, we assume that users connect anonymously to the bridge distributor when submitting reports, and so the concern is just the information that the reports themselves reveal. Any reporting scheme must minimally associate the locations of users with the bridges that are inaccessible to them, along with the time when the report is received by the bridge distributor. If the number of users of a particular bridge in a single region is assumed to be one, then the bridge distributor can assume that all reports for that bridge from that region come from the same user; however, the identity of that user will still be unknown. Additionally, if bridges are only considered blocked in a region as a result of users submitting reports (which are validated by subsequent scans), then users who do not submit reports can infer from a bridge being classified as blocked in their region that some other user(s) in their region submitted reports. The simpler scheme leaks only the information we have already identified, while the full scheme leaks additional information (detailed in the remainder of this section) but provides a better defense against malicious users submitting many reports.

When a user submits multiple reports during a single session in our full scheme, the bridge distributor learns that those reports were all submitted by the same user (but still does not learn which person that user is). (As stated above, the bridge distributor may already be able to infer in some cases that there is only one user of the bridge in that region.) The bridge distributor can also infer information about the number of users (or, at least, the number of credentials) in each region that have access to each bridge. For example, if the bridge distributor receives n reports for a given bridge from a given region simultaneously, then it learns that there must be n different credentials corresponding to that bridge, held by users who are (or claim to be) in that region, which have not previously been used to submit too many false reports. (We note that when Lox users invite their friends, the bridge distributor does not learn to which bucket the new user is assigned; thus, a bound on the number of users per bucket is additional information leaked by our scheme.)

Users may also learn information about other users from our scheme. If a user submits a report, they may learn (from how their report gets resolved, e.g., whether it times out) whether other users in the same region also submitted reports for their bridge. While the bridge distributor does not know the social relationships between users, these likely *are* known by users (who invite each other to join the system). Thus, users may be able to infer, for example, whether or not their friends are trying and failing to use bridges at some particular times, based on inferred reporting behavior.

Ultimately, we consider this additional leakage to be acceptable, and we believe that it is worth the benefit of defending against users who submit many false reports.

6 Performance Evaluation

We extended the Tor Project's Lox system to include our reporting protocols described in the previous section. In this section, we evaluate the performance of our system.

6.1 Implementation and Experiment Setup

We base our Rust prototype of Troll Patrol on a development branch[2] of the Tor Project's Lox code. Our code is publicly available,[3] as are scripts for reproducing our results.[4]

We use the same performance metrics as were used in the original Lox paper [75], namely, for each protocol, we measure (averaged over 100 runs) the serialized request size (in bytes), the time required for the client to compute the request, the serialized bridge distributor response size, the time required to compute the bridge

[2]https://gitlab.torproject.org/onyinyang/lox/-/tree/lox-extension; we forked that repo at commit 88d9fcf56017a28737a7d18d5115efa48525f11e, dated 2025-12-30.
[3]https://git-crysp.uwaterloo.ca/vvecna/lox-troll-patrol-extension
[4]https://git-crysp.uwaterloo.ca/vvecna/troll-patrol-artifact

Table 2: Performance statistics for the Tor Project development branch of Lox on which we based on our code. We use 3,600 bridges (1,800 untrusted open-entry one-bridge buckets and 600 hot spare three-bridge buckets) over 100 runs. All times are reported in milliseconds (ms), and sizes are reported in bytes. Results for the check blockage protocol are reported for 5, 50, and 100% of trusted (invitation-only) bridges being blocked as the performance changes accordingly.

Protocol	Request Size	Request Time	σ	Response Size	Response Time	σ	Response Handling Time	σ
Open Invitation	241	0.165	0.002	539	0.92	0.01	0.660	0.007
Trust Promotion(0→1)	1677	2.68	0.03	252181	753	1	1.51	0.05
Trust Migration (0→1)	845	1.411	0.009	237	2.84	0.04	0.629	0.009
Level Up (1→4)	2253	4.04	0.04	237	7.357	0.006	0.599	0.001
Issue Invitation	1101	2.51	0.02	397	4.509	0.009	1.05	0.01
Redeem Invitation	1037	1.94	0.03	237	3.73	0.05	0.601	0.009
Check Blockage 5%	685	1.161	0.002	4381	14.608	0.002	0.606	0.001
Check Blockage 50%	685	1.18	0.06	42181	126.5	0.5	0.73	0.02
Check Blockage 100%	685	1.16	0.03	84181	250.7	0.6	1.09	0.08
Blockage Migration	973	1.95	0.04	237	3.51	0.07	0.60	0.01

Table 3: Performance statistics for our modified version of Lox, including the Troll Patrol protocols. We use 3,600 bridges (1,800 untrusted open-entry one-bridge buckets and 600 hot spare three-bridge buckets) over 100 runs. All times are reported in milliseconds (ms), and sizes are reported in bytes. Results for the check blockage protocol are reported for 5, 50, and 100% of trusted (invitation-only) bridges being blocked as the performance changes accordingly. The Report Submit request size includes the size of the optional resolve credential that only needs to be presented if the user's credential has $p \neq 0$.

Protocol	Request Size	Request Time	σ	Response Size	Response Time	σ	Response Handling Time	σ
Open Invitation	241	0.166	0.004	539	1.04	0.02	0.72	0.01
Trust Promotion(0→1)	1805	2.89	0.06	252181	753	1	1.50	0.04
Trust Migration (0→1)	973	1.677	0.007	237	3.32	0.01	0.657	0.002
Level Up (1→4)	2445	4.58	0.07	237	8.17	0.07	0.600	0.001
Issue Invitation	1229	2.84	0.02	397	5.09	0.07	1.08	0.01
Redeem Invitation	1133	2.21	0.04	237	4.19	0.05	0.630	0.003
Check Blockage 5%	877	1.569	0.006	4381	15.8	0.7	0.65	0.06
Check Blockage 50%	877	1.57	0.03	42181	127.3	0.4	0.724	0.008
Check Blockage 100%	877	1.562	0.005	84181	251.6	0.6	1.1	0.1
Blockage Migration	1101	2.206	0.008	237	3.981	0.005	0.629	0.001
Report Submit	1300	2.72	0.03	237	4.56	0.06	0.543	0.008
Report Status	877	1.96	0.03	397	3.86	0.03	1.080	0.004
Report Resolve	1069	2.174	0.003	205	3.75	0.01	0.541	0.002

distributor's response, and the time required by the client to process this response. Each experiment was run on a single core of an Intel Xeon Platinum 8380 at 2.30 GHz in a Docker container running Debian 13. The Lox paper presented performance statistics for Lox protocols initialized with 1,800 untrusted open-entry one-bridge buckets and 600 hot spare three-bridge buckets (meaning 3,600 bridges total), based on the total number of Tor bridges at the time. We run our performance evaluation using the same bridge pool size, also allocating half of the bridges to open-entry one-bridge buckets and half to hot spare buckets. We evaluate these performance metrics for each existing Lox protocol and for our new reporting protocols. As we added new attributes to the primary Lox credential, the communication and computation costs for existing protocols are affected.

We also evaluated the same performance metrics (for existing Lox protocols) for the version of the Lox code from the original Lox paper.[5] The results of this evaluation can be found in Appendix C.

[5]https://git-crysp.uwaterloo.ca/iang/lox

6.2 Results

We present our results in Tables 2 and 3. We show that our new protocols for reporting are comparable to the existing Lox protocols in terms of communication and computation. The addition of the p and f attributes to Lox credentials results in a small increase in the request size (around 100–200 bytes) and a small increase in computation cost for protocols unrelated to reporting.

7 Limitations and Discussion

The obvious effective way to determine which bridges are blocked would be to periodically perform active scans of all bridges of interest; however, this approach may draw the attention of the censor both to the volunteers performing the scans (if applicable) and to the bridges being tested, causing more harm than good. As a compromise, Troll Patrol aims to drastically decrease the number of active scans that must be performed by asking users to submit reports when their bridges are blocked and only scanning those bridges for which users submit reports, only from the regions where

users indicate the bridges are inaccessible. Additionally, users who submit reports for bridges that are accessible in order to induce active scans will quickly have their reporting privileges revoked.

In Section 5, we presented a design for negative user reports, using anonymous credentials to track the number of times a user has submitted inaccurate information and forbidding reports from users who have submitted too much inaccurate information, while protecting user privacy by not requiring users to identify themselves or link together multiple reports submitted at different points in time. We do not prescribe precisely how users should submit these reports or how subsequent active scans should be performed, but we discuss some ethical and practical considerations for these actions here, along with some limitations of our work.

7.1 Report Submission

Reports should be submitted by a user's client only if the user grants consent. The user should be informed of the privacy implications of submitting reports in an understandable way. For example, the consent dialog might explain that the report would contain their region, the bridge, and proof that the bridge being reported has been distributed to them, but nothing that individually identifies the user. The user should also be informed that if they submit too many reports for bridges that are determined to be accessible from their region, they will lose the ability to submit new reports; however, this will neither reveal their identity nor affect their reputation within the rest of the system. Per Assumption 2, users have some way to contact the bridge distributor anonymously to submit reports. The user's client should handle report submission seamlessly (provided the user grants consent) and ensure that the anonymous channel is used properly.

The user's client should try to end reporting sessions as soon as possible, to minimize the number of reports that can be linked together; in the case that the user still has pending reports when they are prompted to submit a new report, their client should warn them that if a new report is submitted now, it will be linked to their other recent reports. (For example, the client might show the list of recent reports that can be associated together.) The client should explain that if the user is willing to wait to submit the new report, it will not be associated with any prior reports. (The user might opt to save their setting, e.g., choosing never to submit new reports when existing reports are pending.)

We argue that if the results of active scans usually align with the observations of users, then users are generally incentivized to submit reports when their bridges are inaccessible. (In Section 7.3, we discuss conditions under which active scans may not reliably align with users' observations, and in the next paragraph, we discuss nuances in incentives depending on users' reputations.) Submitting reports can help to identify bridges that are offline (not blocked) so that they can be replaced, and in some systems (including Lox), users can only migrate to new bridges if their old bridges have been identified as blocked. Even if user reports are sometimes (but not regularly) mislabeled as false reports, a user may lose the ability to submit additional reports, but their reputation and ability to migrate to new bridges will not be harmed otherwise. (As discussed in Section 5.4.5, f could be allowed to reset or decay, such that users can regain the ability to submit reports.) If a bridge is blocked by a censor but later becomes accessible again, then the censor may block the bridge again in the future. Thus, when a bridge becomes blocked, it is still better for users to report the bridge immediately so they can migrate to new bridges that have not been discovered by the censor, rather than waiting in case the bridge becomes accessible again.

However, the incentives are more complicated. If a user has enough reputation to migrate to new bridges, or if they have a high-reputation friend who can migrate and then re-invite them to use the friend's new bridges, then the best strategy is to submit reports, to have the bridges classified as blocked as quickly as possible. This is because a censor may have obtained a credential allowing it to use (and block) the bridges and to migrate if the censor's credential accrues enough reputation. Causing the censor to lose its ability to gain reputation as soon as possible minimizes the probability of the censor also being able to migrate to the same new bridges as the user and block those bridges too. However, if a user does not already have enough reputation to migrate or a friend who can help them (and the user believes that the bridge is inaccessible due to censorship, rather than the bridge being offline), then reporting that their bridges are blocked will simply prevent them from gaining reputation. Instead, their optimal strategy is to wait until they do have enough reputation to migrate, while hoping that the bridge is not discovered to be blocked (so that they can continue accruing reputation). Unfortunately, this means that the censor can also accrue reputation during this time.

There is a question of when users should be asked to give consent for the submission of reports. Asking when the user's client fails to connect to a bridge (and is able to submit new reports) is likely to be the best time, as this clearly implies context for the action. The consent dialog could invite the user to grant consent for a single report submission or to grant consent for this and future report submission. (In the latter case, it should, of course, be possible for the user to revoke consent at any time in the future.)

On the other hand, it would be desirable to learn how many users *would* be willing to submit reports, whether or not reports currently need to be submitted. If scanning occurs exclusively as a result of reports being submitted and users are unwilling to submit reports, then it may be assumed that all bridges are accessible, even if they are not. Users could be asked (e.g., after successfully using their bridges for a few days) whether or not they would be willing to submit anonymous reports if they found that their bridges were blocked. This could either be implemented as a proper consent dialog that stores their choice and uses it if their bridges are blocked (and if they have not since changed their selection) or simply as an informal signal to the bridge distributor to determine whether and how frequently to scan bridges without having received reports for those bridges (in which case, the user should, of course, be properly asked for consent when their bridges are blocked). We assume that this indicator of user willingness will be submitted via an anonymity network such as Tor.

7.2 Active Scans

If user reports suggest that a bridge cannot be accessed from a certain region, the bridge distributor should first check whether the bridge is accessible from a different region that is unlikely to

block the bridge. If the bridge is simply offline (and inaccessible everywhere), this should not be treated as a censorship event, and users should not be penalized for submitting reports that the bridge is inaccessible. (This may also be useful information for the bridge distributor. If a bridge is down or unreliable but not blocked by a censor, then users of that bridge may be given a replacement bridge without incurring any reputational penalty.)

We assume that there is some way to actively scan bridges to determine whether or not they are accessible from within a particular region of interest; however, how these scans should work is not obvious. Solving this problem is orthogonal to our work, and we do not provide a complete solution, but we describe some relevant considerations for this problem.

The most obvious solution would be to have volunteers in the region attempt to connect to the bridge. This has two major problems: it may put those volunteers at risk, and it requires exposing the secret identities of bridges to more parties, thus increasing the risk of exposing this information to censors. The former issue is an inherent problem with this approach to censorship measurement and is well known, but the latter is somewhat unique to the setting of measuring censorship of *secret* resources. Relying on volunteers in this way also requires trusting the results provided by the volunteers.

An alternative to asking volunteers to perform measurements is to rent infrastructure (such as virtual private servers) or use proxy servers in regions of interest and conduct the scans using these servers. By eliminating the need for human participants, this avoids the risks to those participants, but there are various limitations of this approach as well. This approach is only possible if there is such infrastructure available in the regions of interest (and the infrastructure must *actually* be located in the regions advertised; companies that provide access to proxies are known to lie about their servers' locations [82]). It is also possible that a censor would apply different censorship policies to such business-oriented infrastructure than to regular people using residential Internet plans; thus, successful connections from such infrastructure may not conclusively indicate that a bridge is not blocked. Additionally, collusion of the infrastructure providers with the censor could again expose secret bridge identities.

There are techniques for measuring censorship without controlling a vantage point within the region of interest, either by attempting a connection from the bridge to a resource within the region [63, 64] or by remotely inducing a connection from a resource in the region of interest to the bridge and measuring its success [23, 24]. These techniques depend on censorship being implemented in certain ways that can be measured by these scans. For example, the former approach requires that censorship is implemented symmetrically and that there is an identified world-accessible resource within the region (e.g., a public website) that can be used for testing. Both require that the censor applies the same restrictions to the resource within the region that is used for measurements as it applies to regular people.

7.3 Additional Limitations

Troll Patrol would not be effective against certain adversarial behaviors or under certain conditions. We assume that users will submit reports and that scans should only be performed in response to user reports. Consequently, if users do not submit reports (e.g., due to privacy concerns or because they are not incentivized to do so at their current reputation), then Troll Patrol would not detect that their bridges are blocked.

Furthermore, if a bridge is blocked in a region but users in that region do not attempt to connect to it, then this censorship will go undetected. We do not consider this to be a significant issue, as the users who *are* trying to use the bridge are not prevented from using the bridge. We also comment that Lox assumes that bridges will not be forgotten by censors and once blocked should not be considered unblocked later (except in special circumstances, such as protocol-level blocking). Thus, Lox's model would prevent bridges from being considered unblocked just because users give up on trying to connect to them.

We assume that censorship is uniform within a region; i.e., either all connections from the region to the bridge will be blocked, or all those connections will be unblocked. There are various ways this assumption can fail. The region boundaries may be defined too broadly, such that multiple censorship policies are enforced within the region. For example, different ISPs in India apply different censorship policies [36, 60], so the entirety of India should not be treated as a single region. Even within a correctly defined region, censorship may not be consistent. For example, a censor may apply some probability of failure to each connection for a forbidden resource, rather than blocking all connections to forbidden resources. A censor may also allow connections but throttle them to the point of being unusable [89], which may not appear as censorship. A valuable area of future work would be adapting to new information about specific regions or censorship models, such as re-dividing regions observed to apply multiple censorship policies or incorporating information about how specific censors enforce censorship (e.g., trying multiple connections or factoring in the speed of the connection).

We comment that Lox currently does not consider regions, instead modeling each bridge as either blocked (everywhere) or not blocked (everywhere). The simplest way to apply our work to Lox would be to determine whether a bridge is blocked *somewhere* and if so, have Lox consider it blocked *everywhere*. However, we comment that the assumption of a global censor may be a limitation of Lox that should be re-evaluated in the future.

8 Related Work

Bridges were proposed for Tor in 2006 by Dingledine and Mathewson [17]. Since then, much work has been done to identify ways that bridges might be discovered by censors [16, 20, 22, 29, 38, 41, 43, 76, 78, 85], and to protect bridges from discovery using pluggable transports that resist traffic analysis and active probing [5, 6, 9, 27, 30, 32, 46, 59, 61, 83, 86] and reputation-based bridge distribution [19, 40, 42, 75, 80].

In 2019, Nasr et al. presented a new framework for bridge distribution and evaluated stragies for bridge distribution and censorship, finding that their framework was more resilient against censorship than rBridge's reputation system [48]. Nasr et al. did not compare their framework to other extant reputation-based bridge distribution systems, such as Salmon [19] or Hyphae [40]. In 2026, Fares et

al. performed a similar evaluation [25] that focused on ephemeral bridges, modeling Snowflake [9] for their analysis. Fares et al. did not analyze reputation-based bridge distribution systems. Notably, both of these papers assume that the bridge distributor knows which bridges have been blocked by censors, but neither explores the question of how it learns this information.

Measuring Internet censorship is a well-studied problem. The obvious way to determine whether an action will be blocked within a region is to attempt the action from that region. By seeing if and how the connection fails, one can learn whether the action was permitted. This approach has been taken by many censorship measurement studies and projects [1, 14, 21, 28, 47, 53, 79, 87].

However, it is not always necessary to have a vantage point in a region to measure censorship in that region. In 2006, Clayton et al. used the known fact that China's firewall operated symmetrically (meaning that it blocked forbidden content from both outgoing and incoming communications) to study censorship within China from England [11]. Follow-up work has similarly measured symmetric censorship [88]. In 1998, antirez presented a technique for inducing a connection between two hosts and measuring its success [2]. The approach of inducing possibly forbidden behavior from remote hosts has been used to detect censorship without the need for vantage points within a region of interest or control of potentially censored hosts [23, 24, 49, 56, 57, 77].

In 2015, Crandall et al., Jones and Feamster, and Jones et al. discussed ethical considerations for censorship measurements and practices to minimize the potential for harm to human participants [13, 33, 34]. In 2015 and 2018, Nisar et al. argued that censorship measurement and censorship circumvention should go together [50, 51]; that is, a user who consents to submitting censorship measurements should use the same tool both for reporting that they were unable to access some resource and for circumventing the censorship to gain access to that resource.

While there has been much work on detecting censorship, this prior work has focused on censorship of known content, such as common keywords and popular websites. There has been relatively little work on detecting which *secret* bridges have been discovered and blocked, a problem that is unique in that it risks revealing those secret bridges to the censor and leading to further censorship. In 2011, Loesing discussed the use of bridge statistics for detecting censorship using a case study of bridge usage in China in the first half of 2010 [39]. Loesing concluded that these published statistics appeared to be be useful for detecting censorship of high-use bridges. Notably, this study only considered bridges that had received connections from at least 100 different Chinese IP addresses in a single day during the period evaluated. Later in 2011, Dingledine discussed sources of information that could be used to discover which bridges were accessible, identifying various types of active scans, bridge statistics, and reports from users [15]. These ideas have received some further attention in public Tor Project discussions [63, 64, 66, 67], but this problem has not been given sufficient academic attention. In a 2013 paper on censorship analysis for Tor, Winter explicitly recommended asking users to scan Tor bridges to test reachability and provided some considerations around anonymity, user consent, and the mechanics of report submission [84]. However, this paper did not sufficiently address techniques for limiting abuse of the reporting system by censors or users.

Perhaps the most similar work to our own is OONI's recently announced anonymous credential system [52, 62], which also aims to ensure the quality of user-submitted censorship measurements without violating the privacy of those users. Notably, however, OONI aims to measure censorship of publicly known resources; as discussed, this is a different problem with different constraints than measuring censorship of secret resources, such as bridges.

9 Conclusion

Reputation-based bridge distribution systems require knowledge of which bridges have been blocked by censors, in order to appropriately increase or decrease users' reputations. This requirement can be fulfilled through active scans that test whether a connection can be made between a resource in a region of interest and a bridge; however, such scans risk revealing secret information about bridges to censors and resulting in additional bridges becoming blocked. Rather than routinely scanning all bridges, a bridge distribution system can encourage users to submit reports when their bridges are inaccessible, then scan just the bridges that have been reported by users.

In this paper, we presented Troll Patrol, an anonymous reporting scheme for bridge distribution systems that use anonymous credentials. We consider and mitigate various vectors for abuse by both censors and users, as well as the possibility that users accidentally submit inaccurate reports. In particular, Troll Patrol indicates which bridges should be scanned and uses the results of those scans to validate users' reports, revoking users' ability to submit additional reports if they have previously submitted too many inaccurate reports. Troll Patrol enforces this restriction in a privacy-preserving way, without users needing to identify themselves or link together multiple reports submitted by the same user at different points in time. We provide a prototype implementation of Troll Patrol for the Tor Project's Lox bridge distribution system and demonstrate that our modifications have only a small impact on the performance of the system. If users are satisfied by the privacy protections and willing to submit reports when they are unable to connect to their bridges, then Troll Patrol can drastically reduce the need for active scanning.

Acknowledgments

We acknowledge the support of the NSERC Discovery Grant RGPIN-2023-03260. This research was undertaken, in part, thanks to funding from the Canada Research Chairs program, award CRC-2018-00135. This work benefitted from the use of the CrySP RIPPLE Facility at the University of Waterloo. We thank Lindsey Tulloch and Cecylia Bocovich at the Tor Project for providing insights about Lox.

References

[1] Collin Anderson. 2012. *The Hidden Internet of Iran: Private Address Allocations on a National Network.* Technical Report. https://arxiv.org/pdf/1209.6398v1.pdf

[2] antirez. 1998. new tcp scan method. https://seclists.org/bugtraq/1998/Dec/79 Accessed May 2026.

[3] Man Ho Au, Willy Susilo, and Yi Mu. 2006. Constant-Size Dynamic k-TAA. In *Proceedings of the 5th International Conference on Security and Cryptography for Networks*, Roberto De Prisco and Moti Yung (Eds.). Berlin, Heidelberg, Germany, 111–125.

[4] Amnesty International Australia. 2026. Iran: Internet shutdown hides violations in escalating deadly crackdown on protesters. https://www.amnesty.org.au/iran-internet-shutdown-deadly-crackdown-protesters/ Accessed May 2026.

[5] Diogo Barradas, Nuno Santos, and Luís Rodrigues. 2017. DeltaShaper: Enabling Unobservable Censorship-resistant TCP Tunneling over Videoconferencing Streams. *Proceedings on Privacy Enhancing Technologies* 2017, 4 (2017), 1–18.

[6] Diogo Barradas, Nuno Santos, Luís Rodrigues, and Vítor Nunes. 2020. Poking a Hole in the Wall: Efficient Censorship-Resistant Internet Communications by Parasitizing on WebRTC. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. Virtual Event, 35–48.

[7] Cecylia Bocovich. 2021. Possible new blocking event in Belarus. https://bugs.torproject.org/tpo/anti-censorship/censorship-analysis/40002 Accessed May 2026.

[8] Cecylia Bocovich. 2021. Tor blocking events in Belarus from 2020-2021. https://bugs.torproject.org/tpo/anti-censorship/censorship-analysis/-/blob/main/reports/2020/belarus/2020-belarus-report.md Accessed May 2026.

[9] Cecylia Bocovich, Arlo Breault, David Fifield, Serene, and Xiaokang Wang. 2024. Snowflake, a censorship circumvention system using temporary WebRTC proxies. In *Proceedings of the 33rd USENIX Security Symposium*. Philadelphia, PA, USA, 2635–2652.

[10] Melissa Chase, Sarah Meiklejohn, and Gregory M. Zaverucha. 2014. Algebraic MACs and Keyed-Verification Anonymous Credentials. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. Scottsdale, AZ, USA, 1205–1216.

[11] Richard Clayton, Steven J. Murdoch, and Robert N. M. Watson. 2006. Ignoring the Great Firewall of China. In *Proceedings of the 6th Workshop on Privacy Enhancing Technologies*. 20–35.

[12] Kevin Collier. 2025. Iran plunged into an internet near-blackout during deepening conflict. https://www.nbcnews.com/tech/internet/iran-plunged-internet-blackout-deepening-conflict-rcna213544 Accessed May 2026.

[13] Jedidiah R. Crandall, Masashi Crete-Nishihata, and Jeffrey Knockel. 2015. Forgive Us our SYNs: Technical and Ethical Considerations for Measuring Internet Filtering. In *Proceedings of the 2015 ACM SIGCOMM Workshop on Ethics in Networked Systems Research*. London, UK, 3. Full version available at https://jeffreyknockel.com/publications/nsethics2015-syns.pdf.

[14] Jakub Dalek, Bennett Haselton, Helmi Noman, Adam Senft, Masashi Crete-Nishihata, Phillipa Gill, and Ronald J. Deibert. 2013. A Method for Identifying and Confirming the Use of URL Filtering Products for Censorship. In *Proceedings of the 2013 Conference on Internet Measurement Conference*. Barcelona, Spain, 23–30.

[15] Roger Dingledine. 2011. *Five ways to test bridge reachability*. Technical Report 2011-12-001. The Tor Project. https://research.torproject.org/techreports/five-ways-test-bridge-reachability-2011-12-01.pdf

[16] Roger Dingledine. 2011. *Ten ways to discover Tor bridges*. Technical Report 2011-10-002. The Tor Project. https://research.torproject.org/techreports/five-ways-test-bridge-reachability-2011-12-01.pdf

[17] Roger Dingledine and Nick Mathewson. 2006. *Design of a blocking-resistant anonymity system*. Technical Report 2006-11-001. The Tor Project. https://research.torproject.org/techreports/blocking-2006-11.pdf

[18] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*. San Diego, CA, USA.

[19] Frederick Douglas, Rorshach, Weiyang Pan, and Matthew Caesar. 2016. Salmon: Robust Proxy Distribution for Censorship Circumvention. *Proceedings on Privacy Enhancing Technologies* 2016, 4 (2016), 4–20.

[20] Arun Dunna, Ciarán O'Brien, and Phillipa Gill. 2018. Analyzing China's Blocking of Unpublished Tor Bridges. In *Proceedings of the 8th USENIX Workshop on Free and Open Communications on the Internet*. Baltimore, MD, USA.

[21] Kathrin Elmenhorst, Bertram Schütz, Nils Aschenbruck, and Simone Basso. 2021. Web Censorship Measurements of HTTP/3 over QUIC. In *Proceedings of the 21st ACM Internet Measurement Conference*. Virtual Event, 276–282.

[22] Roya Ensafi, David Fifield, Philipp Winter, Nick Feamster, Nicholas C. Weaver, and Vern Paxson. 2015. Examining How the Great Firewall Discovers Hidden Circumvention Servers. In *Proceedings of the 2015 Internet Measurement Conference*. Tokyo, Japan, 445–458.

[23] Roya Ensafi, Jeffrey Knockel, Geoffrey Alexander, and Jedidiah R. Crandall. 2014. Detecting Intentional Packet Drops on the Internet via TCP/IP Side Channels. In *Proceedings of the 15th International Conference on Passive and Active Measurement*. Los Angeles, CA, USA, 109–118.

[24] Roya Ensafi, Jong Chun Park, Deepak Kapur, and Jedidiah R. Crandall. 2010. Idle Port Scanning and Non-interference Analysis of Network Protocol Stacks Using Model Checking. In *Proceedings of the 19th USENIX Security Symposium*. Baltimore, MD, USA, 257–272.

[25] Hassan Fares, Omkar Fulsundar, and Nicholas Hopper. 2026. The Game Has Changed: Revisiting proxy distribution and game theory. *Free and Open Communications on the Internet* 2026, 1 (2026), 14–22.

[26] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. 2015. Blocking-resistant communication through domain fronting. *Proceedings on Privacy Enhancing Technologies* 2015, 2 (2015), 46–64.

[27] Gabriel Figueira, Diogo Barradas, and Nuno Santos. 2022. Stegozoa: Enhancing WebRTC Covert Channels with Video Steganography for Internet Censorship Circumvention. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*. Nagasaki, Japan, 1154–1167.

[28] Arturo Filastò and Jacob Appelbaum. 2012. OONI: Open Observatory of Network Interference. In *Proceedings of the 2nd USENIX Workshop on Free and Open Communications on the Internet*. Bellevue, WA, USA.

[29] Sergey Frolov, Jack Wampler, and Eric Wustrow. 2020. Detecting Probe-resistant Proxies. In *Proceedings of the 27th Network and Distributed System Security Symposium*. San Diego, CA, USA.

[30] Sergey Frolov and Eric Wustrow. 2020. HTTPT: A Probe-Resistant Proxy. In *Proceedings of the 10th USENIX Workshop on Free and Open Communications on the Internet*. Virtual Event.

[31] Gustavo Gus. 2021. [Russia] Monitoring dynamic bridges health. https://bugs.torproject.org/tpo/community/support/40054 Accessed May 2026.

[32] Bridger Hahn, Rishab Nithyanand, Phillipa Gill, and Rob Johnson. 2016. Games without Frontiers: Investigating Video Games as a Covert Channel. In *Proceedings of the 2016 IEEE European Symposium on Security and Privacy*. Saarbrücken, Germany, 63–77.

[33] Ben Jones, Roya Ensafi, Nick Feamster, Vern Paxson, and Nick Weaver. 2015. Ethical Concerns for Censorship Measurement. In *Proceedings of the 2015 ACM SIGCOMM Workshop on Ethics in Networked Systems Research*. London, UK, 17–19.

[34] Ben Jones and Nick Feamster. 2015. Can Censorship Measurements Be Safe(r)?. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*. Philadelphia, PA, USA.

[35] Sina Kamali and Diogo Barradas. 2025. Anix: Anonymous Blackout-Resistant Microblogging with Message Endorsing. In *Proceedings of the 46th IEEE Symposium on Security and Privacy*. San Francisco, CA, USA, 1381–1399.

[36] Divyank Katira, Gurshabad Grover, Kushagra Singh, and Varun Bansal. 2023. CensorWatch: On the Implementation of Online Censorship in India. *Free and Open Communications on the Internet* 2023, 1 (2023), 35–45.

[37] Adam Lerner, Giulia Fanti, Yahel Ben-David, Jesus Garcia, Paul Schmitt, and Barath Raghavan. 2016. Rangzen: Anonymously Getting the Word Out in a Blackout. https://arxiv.org/abs/1612.03371

[38] Zhen Ling, Xinwen Fu, Wei Yu, Junzhou Luo, and Ming Yang. 2012. Extensive Analysis and Large-Scale Empirical Evaluation of Tor Bridge Discovery. In *Proceedings of the 31st Annual International Conference on Computer Communications*. Orlando, FL, USA, 2381–2389.

[39] Karsten Loesing. 2011. *Case study: Learning whether a Tor bridge is blocked by looking at its aggregate usage statistics: Part One*. Technical Report 2011-09-002. The Tor Project. https://research.torproject.org/techreports/blocking-2011-09-15.pdf

[40] Isis Agora Lovecruft and Henry de Valence. 2017. Hyphae: Social Secret Sharing. https://patternsinthevoid.net/hyphae/hyphae.pdf

[41] Srdjan Matic, Carmela Troncoso, and Juan Caballero. 2017. Dissecting Tor Bridges: A Security Evaluation of their Private and Public Infrastructures. In *Proceedings of the 24th Network and Distributed System Security Symposium*. San Diego, CA, USA.

[42] Damon McCoy, Jose Andre Morales, and Kirill Levchenko. 2011. Proximax: A Measurement Based System for Proxies Dissemination. In *Proceedings of the 15th International Conference on Financial Cryptography and Data Security*. Gros Islet, St. Lucia, 260–267.

[43] Jon McLachlan and Nicholas Hopper. 2009. 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*. Chicago, IL, USA, 31–40.

[44] meskio. 2025. Comment on "Question about new built in meek-azure bridge". https://forum.torproject.org/t/question-about-new-built-in-meek-azure-bridge/19978/3 Accessed May 2026.

[45] meskio. 2026. gaps on the https requests. https://bugs.torproject.org/tpo/anti-censorship/rdsys/293 Accessed May 2026.

[46] Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. 2012. SkypeMorph: Protocol Obfuscation for Tor Bridges. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. Raleigh, NC, USA, 97–108.

[47] Zubair Nabi. 2013. The Anatomy of Web Censorship in Pakistan. In *Proceedings of the 3rd USENIX Workshop on Free and Open Communications on the Internet*. USENIX, Washington, D.C., USA.

[48] Milad Nasr, Sadegh Farhang, Amir Houmansadr, and Jens Grossklags. 2019. Enemy At the Gateways: Censorship-Resilient Proxy Distribution Using Game Theory. In *Proceedings of the 26th Network and Distributed System Security Symposium*. San Diego, CA, USA.

[49] Arian Akhavan Niaki, Shinyoung Cho, Zachary Weinberg, Nguyen Phong Hoang, Abbas Razaghpanah, Nicolas Christin, and Phillipa Gill. 2020. ICLab: A Global, Longitudinal Internet Censorship Measurement Platform. In *Proceedings of the 41st IEEE Symposium on Security and Privacy*. Virtual Event, 135–151.

[50] Aqib Nisar, Aqsa Kashaf, Ihsan Ayyub Qazi, and Zartash Afzal Uzmi. 2018. Incentivizing Censorship Measurements via Circumvention. In *Proceedings of the 2018*

Conference of the ACM Special Interest Group on Data Communication. Budapest, Hungary, 533–546.
[51] Aqib Nisar, Aqsa Kashaf, Zartash Afzal Uzmi, and Ihsan Ayyub Qazi. 2015. A Case for Marrying Censorship Measurements with Circumvention. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*. Philadelphia, PA, USA.
[52] Michele Orrù, Lindsey Tulloch, Victor Snyder-Graf, and Ian Goldberg. 2026. sigma-rs: A Modular Approach for Keyed-Verification Anonymous Credentials. In *Proceedings of the 35th USENIX Security Symposium*. Baltimore, MD, USA.
[53] Jong Chun Park and Jedidiah R. Crandall. 2010. Empirical Study of a National-Scale Distributed Intrusion Detection System: Backbone-Level Filtering of HTML Responses in China. In *Proceedings of the 30th International Conference on Distributed Computing Systems*. Genoa, Italy, 315–326.
[54] Amogh Pradeep, Hira Javaid, Ryan Williams, Antoine Rault, David Choffnes, Stevens Le Blond, and Bryan Ford. 2022. Moby: A Blackout-Resistant Anonymity Network for Mobile Devices. *Proceedings on Privacy Enhancing Technologies* 2022, 3 (2022), 247–267.
[55] Michael Pu, Andrew Wang, Anthony Chang, Kieran Quan, and Yi Wei Zhou. 2024. Exploring Amazon Simple Queue Service (SQS) for Censorship Circumvention. *Free and Open Communications on the Internet* 2024, 2 (2024), 22–26.
[56] Ram Sundara Raman, Prerana Shenoy, Katharina Kohls, and Roya Ensafi. 2020. Censored Planet: An Internet-wide, Longitudinal Censorship Observatory. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. Virtual Event, 49–66.
[57] Ram Sundara Raman, Adrian Stoll, Jakub Dalek, Reethika Ramesh, Will Scott, and Roya Ensafi. 2020. Measuring the Deployment of Network Censorship Filters at Global Scale. In *Proceedings of the 27th Network and Distributed System Security Symposium*. San Diego, CA, USA.
[58] Piyush Kumar Sharma, Rishi Sharma, Kartikey Singh, Mukulika Maity, and Sambuddho Chakravarty. 2023. Dolphin: A Cellular Voice Based Internet Shutdown Resistance System. *Proceedings on Privacy Enhancing Technologies* 2023, 1 (2023), 589–607.
[59] shelikhoo and ggus. 2024. Hiding in plain sight: Introducing WebTunnel. https://blog.torproject.org/introducing-webtunnel-evading-censorship-by-hiding-in-plain-sight/ Accessed May 2026.
[60] Kushagra Singh, Gurshabad Grover, and Varun Bansal. 2020. How India Censors the Web. In *Proceedings of the 12th ACM Conference on Web Science*. Southampton, UK, 21–28.
[61] Rob Smits, Divam Jain, Sarah Pidcock, Ian Goldberg, and Urs Hengartner. 2011. BridgeSPA: improving Tor bridges with single packet authorization. In *Proceedings of the 10th ACM Workshop on Privacy in the Electronic Society*. Chicago, IL, USA, 93–102.
[62] OONI Team. 2026. Announcing OONI's New Anonymous Credential System. https://ooni.org/post/2025-announcing-ooni-new-anonymous-credential-system/ Accessed May 2026.
[63] The Tor Project. 2010. Learn which bridges can't reach baidu. https://bugs.torproject.org/tpo/core/tor/1851 Accessed May 2026.
[64] The Tor Project. 2011. Can we try to extend from the bridge to a website and learn if the website is reachable? https://bugs.torproject.org/tpo/network-health/metrics/analysis/2576 Accessed May 2026.
[65] The Tor Project. 2021. Bridge Pool Assignments 2021-02. https://collector.torproject.org/archive/bridge-pool-assignments/bridge-pool-assignments-2021-02.tar.xz Accessed May 2026.
[66] The Tor Project. 2022. detect if a bridge is blocked in a certain country by its reported usage. https://bugs.torproject.org/tpo/anti-censorship/rdsys/112 Accessed May 2026.
[67] The Tor Project. 2023. Brainstorm and analyze heuristics to guess that a bridge might be offline or blocked. https://bugs.torproject.org/tpo/anti-censorship/team/176 Accessed May 2026.
[68] The Tor Project. 2026. Bridge users from Belarus. https://metrics.torproject.org/userstats-bridge-country.html?start=2020-07-01&end=2020-10-31&country=by Accessed May 2026.
[69] The Tor Project. 2026. BridgeDB requests by distributor. https://metrics.torproject.org/bridgedb-distributor.html?start=2026-01-01&end=2026-03-31 Accessed May 2026.
[70] The Tor Project. 2026. BridgeDB requests by distributor. https://metrics.torproject.org/bridgedb-distributor.html?start=2020-07-01&end=2021-04-30 Accessed May 2026.
[71] The Tor Project. 2026. CollecTor. https://metrics.torproject.org/collector.html Accessed May 2026.
[72] The Tor Project. 2026. Directly connecting users from Belarus. https://metrics.torproject.org/userstats-relay-country.html?start=2020-07-01&end=2020-10-31&country=by&events=off Accessed May 2026.
[73] The Tor Project. 2026. Lox: A privacy preserving bridge distribution system. https://gitlab.torproject.org/tpo/anti-censorship/lox/-/blob/260dec179c0521711f92a2916680e420c0078135/crates/lox-library/src/lib.rs#L170 Accessed May 2026.
[74] Lindsey Tulloch. 2023. Make Lox Bridge Table more robust. https://bugs.torproject.org/tpo/anti-censorship/lox/7 Accessed May 2026.
[75] Lindsey Tulloch and Ian Goldberg. 2023. Lox: Protecting the Social Graph in Bridge Distribution. *Proceedings on Privacy Enhancing Technologies* 2023, 1 (2023), 494–509.
[76] twilde. 2012. Knock Knock Knockin' on Bridges' Doors. https://blog.torproject.org/knock-knock-knockin-bridges-doors/ Accessed May 2026.
[77] Benjamin VanderSloot, Allison McDonald, Will Scott, J. Alex Halderman, and Roya Ensafi. 2018. Quack: Scalable Remote Measurement of Application-Layer Censorship. In *Proceedings of the 27th USENIX Security Symposium*. Baltimore, MD, USA, 187–202.
[78] Eugene Y. Vasserman, Rob Jansen, James Tyra, Nicholas Hopper, and Yongdae Kim. 2009. Membership-concealing overlay networks. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*. Chicago, IL, USA, 390–399.
[79] John-Paul Verkamp and Minaxi Gupta. 2012. Inferring Mechanics of Web Censorship Around the World. In *Proceedings of the 2nd USENIX Workshop on Free and Open Communications on the Internet*. Bellevue, WA, USA.
[80] Qiyan Wang, Zi Lin, Nikita Borisov, and Nicholas Hopper. 2013. rBridge: User Reputation based Tor Bridge Distribution with Privacy Preservation. In *Proceedings of the 20th Network and Distributed System Security Symposium*. San Diego, CA, USA.
[81] Human Rights Watch. 2026. Iran's Internet Blackout Concealing Atrocities. https://www.hrw.org/news/2026/01/12/irans-internet-blackout-concealing-atrocities Accessed May 2026.
[82] Zachary Weinberg, Shinyoung Cho, Nicolas Christin, Vyas Sekar, and Phillipa Gill. 2018. How to Catch when Proxies Lie: Verifying the Physical Locations of Network Proxies with Active Geolocation. In *Proceedings of the Internet Measurement Conference 2018*. Boston, MA, USA, 203–217.
[83] Zachary Weinberg, Jeffrey B. Wang, Vinod Yegneswaran, Linda Briesemeister, Steven W. T. Cheung, Frank Wang, and Dan Boneh. 2012. StegoTorus: a camouflage proxy for the Tor anonymity system. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. Raleigh, NC, USA, 109–120.
[84] Philipp Winter. 2013. Towards a Censorship Analyser for Tor. In *Proceedings of the 3rd USENIX Workshop on Free and Open Communications on the Internet*. Washington, D.C., USA.
[85] Philipp Winter and Stefan Lindskog. 2012. How the Great Firewall of China is Blocking Tor. In *Proceedings of the 2nd USENIX Workshop on Free and Open Communications on the Internet*. Bellevue, WA, USA.
[86] Philipp Winter, Tobias Pulls, and Jürgen Fuß. 2013. ScrambleSuit: a polymorphic network protocol to circumvent censorship. In *Proceedings of the 12th ACM Workshop on Privacy in the Electronic Society*. Berlin, Germany, 213–224.
[87] Sebastian Wolfgarten. 2006. *Investigating large-scale Internet content filtering*. Technical Report. Dublin City University. https://censorbib.nymity.ch/pdf/Wolfgarten2006a.pdf
[88] Xueyang Xu, Z. Morley Mao, and J. Alex Halderman. 2011. Internet Censorship in China: Where Does the Filtering Occur?. In *Proceedings of the 12th International Conference on Passive and Active Measurement*. Atlanta, GA, USA, 133–142.
[89] Diwen Xue, Reethika Ramesh, Valdik S S, Leonid Evdokimov, Andrey Viktorov, Arham Jain, Eric Wustrow, Simone Basso, and Roya Ensafi. 2021. Throttling Twitter: an emerging censorship technique in Russia. In *Proceedings of the 21st ACM Internet Measurement Conference*. Virtual Event, 435–443.
[90] Wenxuan Zhou, Amir Houmansadr, Matthew Caesar, and Nikita Borisov. 2013. SWEET: Serving the Web by Exploiting Email Tunnels. In *6th Workshop on Hot Topics in Privacy Enhancing Technologies*. Bloomington, IN, USA.

A Case Study: Belarus in 2020–2021

We explore the utility of bridge statistics through the case study of Belarus in 2020–2021 as documented by Bocovich [8].

Starting around August 8, 2020, direct connections to Tor from Belarus significantly fell (presumably indicative of a censorship event), as shown in Figure 3, while bridge use soared, as shown in Figure 4. By August 15, 2020, both counts had returned to their normal ranges. Censorship of Tor resumed in October.

In February 2021, it appears that Belarus began enumerating and blocking Tor bridges. At that time, Tor bridges were distributed via three distribution channels: HTTPS, email, and Moat (domain fronting). The set of bridges was partitioned such that each bridge was distributed through at most one of these three channels. (We say "at most" because some bridges were unallocated and were not distributed through any of these channels.) Based on a sample of bridges distributed through each channel (including both bridges

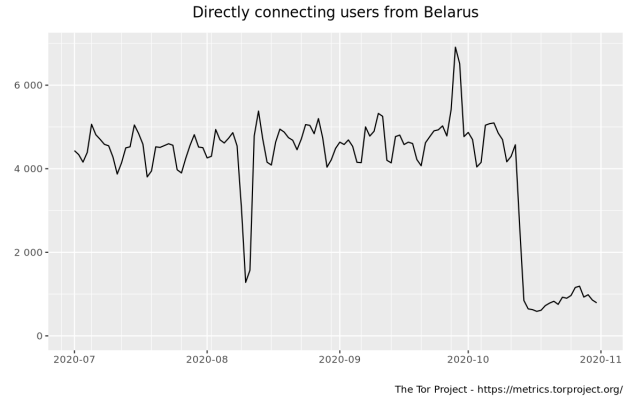

Figure 3: Average number of concurrent users from Belarus who connected directly to Tor from July through October 2020. Graph from the Tor Project [72].

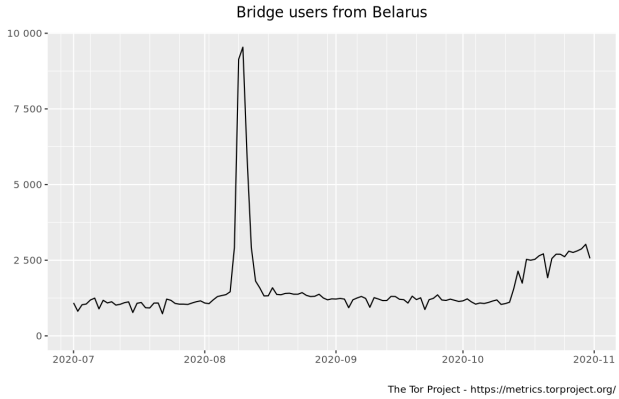

Figure 4: Average number of concurrent users from Belarus who connected to Tor using bridges from July through October 2020. Graph from the Tor Project [68].

that used vanilla, i.e., unobfuscated, Tor and bridges that used obfs4 to resist deep-packet inspection), Bocovich reports that only obfs4 bridges that were first distributed by email on or before February 22 were blocked, and all tested obfs4 bridges distributed by email before February 22 were blocked. Two obfs4 bridges first distributed on February 22 were tested; one was blocked, while the other was not.

This case study has two features that make it potentially useful for validating a censorship detection algorithm. First, we have real-world data and a set of bridges we believe were blocked. If the algorithm detects that bridges in that set were blocked in late February 2021 (when we believe they were actually blocked), while avoiding detecting other bridges as blocked, then that supports the algorithm's accuracy. Second, the rise in bridge use in August 2020, followed by a decline in bridge use due to access to vanilla Tor being restored, not due to increasing censorship, is an instance in which an algorithm might report false positives (detecting a drop in bridge use as an indicator of censorship). Through this instance, we can evaluate whether an algorithm can successfully detect censorship while avoiding false positives.

A.1 Methodology

To identify the relevant bridges, we downloaded the February 2021 `bridge-pool-assignments` archive [65]. The files in this archive indicate which bridges were distributed via which channels. We parsed the records from February 1–21 to find all the bridges that supported obfs4 and were distributed via email. Out of 1,890 total bridges represented in these records, we identified only 93 email-distributed obfs4 bridges. (530 bridges were distributed by HTTPS, 1135 by Moat, and 102 were unallocated. 123 bridges were distributed by email; of these, 93 reported supporting obfs4.)

We started with the algorithm used in Loesing's work [39], provided as Algorithm 1. In this algorithm, a bridge is considered blocked if it receives 32 or fewer connections on a given day. Loesing's work only considered bridges as blocked if they received at least 100 connections (that is, at least 97, reported as at least 104 due to rounding) on some day during the period; we replicate this constraint.

Algorithm 1 Bridge blockage detection with an absolute threshold of 32 and an absolute connection count minimum of 100, as used in Loesing's study.

```
function IsBlocked(bridge_ips, date)
    u ← max_i(bridge_ips_i)
    return bridge_ips_date ≤ 32 ∧ u ≥ 100
end function
```

Algorithm 2 Bridge blockage detection with an absolute threshold of t and an absolute connection count minimum of m.

```
function IsBlocked(t, m, bridge_ips, date)
    u ← max_{i<date}(bridge_ips_i)
    return bridge_ips_date ≤ t ∧ u ≥ m
end function
```

Algorithm 3 Bridge blockage detection with a relative connection count difference d.

```
function IsBlocked(d, bridge_ips, date)
    return bridge_ips_date ≤ bridge_ips_{date−1} − d
end function
```

However, we also make an attempt to detect censorship of bridges with fewer regular connections. Algorithm 2 is modeled after the algorithm by Loesing but varies both the threshold t for considering a bridge blocked (32 in Loesing's case, varied from 8 to 32 in ours) and the requisite number of connections m that must have been observed (104 in Loesing's case, varied from 16 to 104 in ours, with the minimum being 8 higher than the threshold for considering a bridge blocked). Unlike Loesing, we only consider prior days' connection counts when determining whether m has been achieved. Thus, a new bridge that receives low connection counts for a few days, then rises to 104 connections should not be considered to have been blocked in the first few days.

We also consider detecting censorship due to sudden drops in day-to-day connection counts, rather than an absolutely low number. Algorithm 3 considers a bridge blocked on a given day if that day's connection count is d less than the previous day's count.

We downloaded the `extra-info` records for all bridges from July 2020 through April 2021 from CollecTor and parsed the number of users connecting from Belarus for each day from the `bridge-ips` field for each bridge. Notably, if a bridge does not receive any connections from a given country on a day, it omits that country code from its `bridge-ips` field. We treated the absence of "by" (the country code for Belarus) as 0 connections if and only if the bridge had previously reported receiving a connection from a Belarusian IP. (In other words, we did not start considering that a bridge might be blocked until we had evidence that the bridge had ever been distributed to someone in Belarus.) Similarly, if a bridge did not publish an `extra-info` record for a day (which might have happened, for example, if the bridge was offline during that day), we treated this as missing data and excluded that day from the analysis of the bridge, rather than treating it as a record of 0 connections.

We used Algorithms 1, 2, and 3 to infer which bridges were blocked and when, then compared the list of bridges inferred as blocked to the list of bridges that were actually identified as blocked in February 2021. For Algorithm 2, we varied t from 8 to 32 and m from $t + 8$ to 104. For Algorithm 3, we varied d from 8 to 104.

Estimating that censorship began no earlier than February 20, we calculated a true positive (TP) as an email-distributed obfs4 bridge being marked as blocked on or after February 20[6] and a false negative (FN) as an email-distributed obfs4 bridge that was not marked as blocked. Likewise, we calculated a true negative (TN) as any bridge that was not marked as blocked and either was not distributed by email or did not use obfs4. There were two cases that we considered false positives (FPs). Any bridge that that was not among the 93 email-distributed obfs4 bridges but was determined to be blocked was a false positive. We also treated an email-distributed obfs4 bridge as a false positive if we determined that it was blocked before the censorship actually began.

A.2 Results

All three algorithms were abysmal classifiers for this case study. Only Algorithm 3 resulted in any true positives at all. It only did so in the $d = 8$ case (in which a bridge was detected as blocked if its daily connection count ever decreased at all), and in this case it resulted in 5 true positives and 995 false positives (along with 848 true negatives and 42 false negatives).

Upon closer inspection, we discovered that the email-distributed obfs4 bridges in question were simply not very popular in Belarus during the period from July 2020 through February 20, 2021. Of the 93 email-distributed obfs4 bridges we considered, only 43 had received any connections at all during this period. Only 13 had received more than 8 connections from Belarus during any day in this period. Of those, only 2 bridges received more than 16 connections, and no bridge received more than 24 connections. We also found that 0 connections was a common value during the period before the active censorship. In fact, only 5 of the 93 bridges had a mean connection count more than one standard deviation above 0. Of these, the greatest distance between 0 and the mean (measured in standard deviations) was only about 1.6 standard deviations.

Of the 30 bridges that were distributed via email but did not report supporting obfs4, only 2 received any connections at all. Neither of these two bridges reported more than 8 connections on any day. By contrast to email-distributed bridges, of the 1,767 bridges that were not distributed by email, 662 reported receiving connections from Belarus on some day during the period, with 492 reporting more than 8 connections, 420 reporting more than 16, and 370 reporting more than 24. 176 non-email bridges reported at least 104 connections on some day during the period. From these counts we can see that there *were* bridges that were popular in Belarus. It was not the case, for example, that bridge use in Belarus was simply very low or uniformly distributed across bridges, such that all bridges had low rates of use. However, the bridges that were popular were not distributed via email.

Our full results from running Algorithms 1, 2, and 3 are provided in Table 4. Algorithms 1 and 2 never produced any true positives at all, and Algorithm 3 only in the $d = 8$ configuration (where it produced 5 true positives and 995 false positives), showing that none of these algorithms are useful classifiers. The issue is that the bridges that were known to be blocked had too low usage before the censorship event to be successfully detected as blocked by these algorithms.

An observation that we expected might help us overcome the limitation of low usage of individual bridges is that bridges are not distributed individually; when a user requests bridges via email, they receive a *pair* of bridge lines. By associating pairs of bridges together, it may be possible to observe a stronger signal. However, it is not clear which bridges were distributed together. The `bridge-pool-assignments` records discussed in Section A.1 include a distribution `ring` value for some records; this value relates to the question of which bridges are distributed together. However, the records for email-distributed bridges do not include a `ring` value, so we could not determine which bridges were actually distributed together. Instead, we considered all email-distributed obfs4 bridges to be potentially distributed together and examined all pairs.

Considering all 4,278 pairs of the 93 bridges, we found that connection counts were still low overall. With single bridges, we had considered data starting on the first date a non-zero value was observed; for each pair of bridges, we considered data for a day only if we had evidence that each bridge in the pair had been distributed to a user in Belarus (i.e., each bridge had received a non-zero connection count on that day or some prior day). While 557 bridge pairs received combined counts of more than 8 on at least one day, only 83 pairs received combined counts greater than 16 on a single day, and only 6 bridge pairs received combined counts greater than 24. No bridge pair ever received more than 32 total connections on a single day. This is particularly noteworthy due to the fact that connection counts are rounded up for each bridge, meaning a value of 40 requires only 26 total connections; e.g., 9 (rounded up to 16) and 17 (rounded up to 24) would make 40. Of the 4,278 bridge pairs, 264 had a mean connection count more than one standard deviation above 0; the greatest distance was still

[6]The specific choice of date in February had very little impact on our results. The results were identical for all dates in the range February 1–27. Algorithm 3 returned one fewer TP (and a corresponding additional FP) when we started on February 28 with $d = 8$.

Table 4: Results from running Algorithm 1 (using an absolute threshold of 32 and an absolute connection count minimum of 100), Algorithm 2 (using an absolute threshold t and an absolute connection count minimum m), and Algorithm 3 (using a relative connection count difference d).

Algorithm 1

TP	TN	FP	FN	Precision	Recall
0	1596	201	93	0	0

Algorithm 2

t	m	TP	TN	FP	FN	Precision	Recall
8	16	0	1145	665	80	0	0
8	24	0	1331	468	91	0	0
8	32	0	1416	381	93	0	0
8	40	0	1480	317	93	0	0
8	48	0	1527	270	93	0	0
8	56	0	1563	234	93	0	0
8	64	0	1586	211	93	0	0
8	72	0	1608	189	93	0	0
8	80	0	1626	171	93	0	0
8	88	0	1640	157	93	0	0
8	96	0	1656	141	93	0	0
8	104	0	1672	125	93	0	0
16	24	0	1319	480	91	0	0
16	32	0	1397	400	93	0	0
16	40	0	1456	341	93	0	0
16	48	0	1499	298	93	0	0
16	56	0	1530	267	93	0	0
16	64	0	1548	249	93	0	0
16	72	0	1572	225	93	0	0
16	80	0	1593	204	93	0	0
16	88	0	1608	189	93	0	0
16	96	0	1626	171	93	0	0
16	104	0	1642	155	93	0	0
24	32	0	1389	408	93	0	0
24	40	0	1443	354	93	0	0
24	48	0	1480	317	93	0	0
24	56	0	1510	287	93	0	0

Algorithm 2 (continued)

t	m	TP	TN	FP	FN	Precision	Recall
24	64	0	1529	268	93	0	0
24	72	0	1553	244	93	0	0
24	80	0	1574	223	93	0	0
24	88	0	1589	208	93	0	0
24	96	0	1609	188	93	0	0
24	104	0	1626	171	93	0	0
32	40	0	1434	363	93	0	0
32	48	0	1471	326	93	0	0
32	56	0	1501	296	93	0	0
32	64	0	1519	278	93	0	0
32	72	0	1543	254	93	0	0
32	80	0	1562	235	93	0	0
32	88	0	1575	222	93	0	0
32	96	0	1592	205	93	0	0
32	104	0	1610	187	93	0	0

Algorithm 3

d	TP	TN	FP	FN	Precision	Recall
8	5	848	995	42	0.005	0.1
16	0	1333	466	91	0	0
24	0	1422	375	93	0	0
32	0	1476	321	93	0	0
40	0	1506	291	93	0	0
48	0	1536	261	93	0	0
56	0	1561	236	93	0	0
64	0	1593	204	93	0	0
72	0	1626	171	93	0	0
80	0	1659	138	93	0	0
88	0	1687	110	93	0	0
96	0	1713	84	93	0	0
104	0	1732	65	93	0	0

only about 1.7 standard deviations from 0 to the mean. Even when considering bridges in pairs to increase the observed connection counts, normal connection rates for email-distributed obfs4 bridges were still too low to reasonably detect censorship based on a drop (even if the connection count fell to 0).

The noteworthy finding of this case study is that bridges distributed via email had very low usage in Belarus. It appears that this trend was global; Figure 5 shows that the number of requests for bridges via email was minuscule compared to the number of requests via HTTPS or Moat during the period we evaluated.

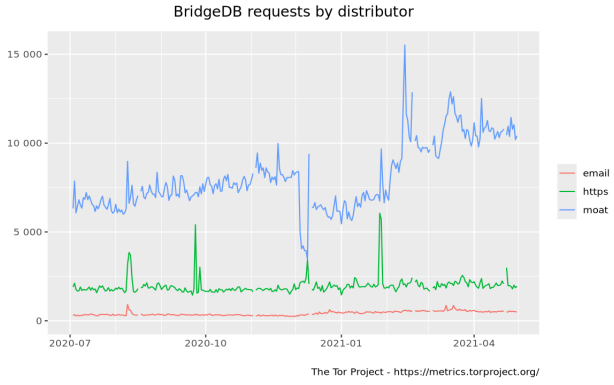

Figure 5: Approximate count of daily requests (by users in any country) for bridges from each distributor during the period July 2020 through April 2021. Graph from the Tor Project [70].

B Extension for Resetting f

The negative reporting scheme discussed in Section 5 only allows the false report count f to increase over time; however, this may cause honest users to lose the ability to submit reports. We describe here an extension to our reporting scheme that allows such users to eventually reset their credentials' f values, but still limits the ability of malicious users to submit many false reports. Intuitively,

Table 5: Performance statistics for the version of the Lox code from the original Lox paper. This table corresponds to Table 4 from the Lox paper [75]. We use 3,600 bridges (1,800 untrusted open-entry one-bridge buckets and 600 hot spare three-bridge buckets) over 100 runs. All times are reported in milliseconds (ms), and sizes are reported in bytes. Results for the check blockage protocol are reported for 5, 50, and 100% of trusted (invitation-only) bridges being blocked as the performance changes accordingly.

Protocol	Request Size	Request Time	σ	Response Size	Response Time	σ	Response Handling Time	σ
Open Invitation	332	0.33	0.02	740	1.38	0.07	0.93	0.05
Trust Promotion(0→1)	2216	4.3	0.2	252104	108	6	0.9	0.1
Trust Migration (0→1)	936	1.9	0.1	584	2.7	0.1	1.15	0.06
Level Up (1→4)	3368	6.5	0.2	712	6.2	0.2	1.51	0.05
Issue Invitation	1672	3.4	0.2	1480	6.4	0.3	3.0	0.2
Redeem Invitation	1576	3.0	0.2	680	3.8	0.2	1.32	0.07
Check Blockage 5%	744	1.407	0.003	4304	4.026	0.006	0.1006	0.0009
Check Blockage 50%	744	1.404	0.003	42104	19.437	0.009	0.202	0.003
Check Blockage 100%	744	1.406	0.007	84104	36.6	0.2	0.316	0.009
Blockage Migration	1224	2.503	0.002	840	3.904	0.004	1.668	0.006

at the moment that a user resolves a reporting session with too high an f value to start a new reporting session, they receive a credential that allows them to reset f to 0, but only after waiting for a certain period of time.

First, we add an attribute to users' primary credentials: f_{since}. When the user's credential is first created, the value of this attribute is set to the creation date, and it is subsequently updated to the current date whenever the user performs a Report Submit or Report Resolve action (as these are the protocols that can change f).

Next, we add a new credential type: the *reset credential*, which has the following attributes:

- Φ: a unique ID (not shared with any other credential)
- f: a false report count, the same as in the user's primary credential

A reset credential is always issued when the user performs Report Resolve with $b = 0$ (resolving their pending status), and its f value is set to the f value of the user's newly issued primary credential; however, the reset credential is only valid if its f value exceeds the allowable false report limit m. This ensures that the bridge distributor does not learn whether or not the user's new f value exceeds m, but the user can only receive a valid reset credential once they lose the ability to submit new reports. This also prevents the user from obtaining multiple valid reset credentials. They can only obtain one by ending a reporting session with a sufficiently high f that they are unable to start new reporting sessions.

Finally, we add a new protocol: Report Reset. In this protocol, the user proves in zero knowledge that

(1) their reset credential's f value exceeds m
(2) their reset credential's f value equals their primary credential's f value
(3) their primary credential's f_{since} value was sufficiently long ago

The only revealed attributes in this protocol are Φ from the reset credential and the ID and p from the user's primary credential (where p must be 0 for this protocol to be allowed). The bridge distributor ensures that the reset credential's ID Φ has never been seen before; if this is the case and the bridge distributor can verify the user's proof as described above, the protocol succeeds, and the reset credential's Φ is added to a deny-list to prevent reuse of the reset credential. (As per usual, the primary credential's ID is also added to a deny-list, and the user's new primary credential contains a new ID that is not known to the bridge distributor.) The user is issued a new credential with a new ID, f set to 0, f_{since} equal to the current date, and all other attributes unchanged.

Reset credentials are not restricted to specific primary credentials (Alice could give her reset credential to Bob to use, provided Bob's f value is the same as the one in Alice's reset credential), but each primary credential can only be used to obtain a single reset credential before resetting its f value. A malicious user may attempt to submit many false reports, but they will soon be restricted from submitting new reports. They will receive a single reset credential and must wait until f_{since} is sufficiently old before using it. While waiting, they may gain the ability to invite friends and use this ability to issue additional credentials for themself; however, those credentials would inherit the f value from the original credential (restricting them from submitting additional reports), and the user would not receive additional reset credentials. They would only be able to reset the f value for one of their credentials.

C Performance Evaluation of Original Lox Code

In Section 6, we provided an evaluation of the performance impact of our Troll Patrol prototype, compared to the Lox development branch from which we forked. Table 5 provides the equivalent performance metrics, collected by running the version of the Lox code from the original Lox paper[7] on the same CPU in the same configuration that was used for our performance evaluations from Section 6.

Interestingly, computing the bridge distributor's response for the Trust Promotion and Check Blockage protocols took an order of magnitude longer for the Lox development branch than for the older version of Lox, and our code inherits this performance penalty. This is likely related to changes to the format of the Lox bridge table that were made in 2023 [74]; the protocols in question require re-encrypting every bucket in the bridge table.

[7] https://git-crysp.uwaterloo.ca/iang/lox