Lox: Protecting the Social Graph in Bridge Distribution

Lindsey Tulloch
University of Waterloo
Waterloo, Canada
ltulloch@uwaterloo.ca

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

ABSTRACT

In regions of the world where censorship of the Internet is used to limit access to information, monitor the activity of Internet users, and quash dissent, anti-censorship proxies, or *bridges*, can offer a connection to the open Internet beyond a censor's area of influence. Bridge distribution systems, built to publicly distribute large pools of bridges to users in censored regions, face the inherent conflict of providing bridges to unknown users when some of them may be malicious. If not designed with care, bridge distribution systems can be quickly overwhelmed by attacks from censors, undermining the integrity of the system and the safety of users. It is therefore crucial to prioritize protecting users when developing such systems.

In this paper, we present a new bridge distribution system, Lox. Lox prioritizes protecting the privacy of users and their social graphs and incorporates enumeration resistance mechanisms to improve access to bridges and limit the malicious behaviour of censors. We use an updated unlinkable multi-show anonymous credential scheme, suitable for a single credential issuer and verifier, to protect Lox bridge users and their social networks from being identified by malicious actors. We formalize a trust level scheme that is compatible with anonymous credentials and effectively limits malicious behaviour while maintaining user anonymity. Our work includes an open-sourced, Rust implementation of our Lox protocols as well as an evaluation of their performance. With reasonable performance and latency for the expected user base of our system, we demonstrate Lox as a practical, social graph protective bridge distribution system.

KEYWORDS

bridge distribution, censorship resistance, anonymous credentials

1 INTRODUCTION

Internet censorship takes many forms to fit the capabilities and motivations of the censor, from repressing self-expression, to controlling the political landscape and information available to users within the regions they control. Recent work demonstrates the variability of Internet censorship employed around the world. In a collaborative report by OutRight Action International, the University of Toronto's Citizen Lab, and the Open Observatory of Network Interference (OONI), Dalek et al. [8] analyze the methods employed by censors across six regions (Indonesia, Malaysia, Russia, Iran, Saudi Arabia and the United Arab Emirates) to censor LGBTIQ websites and related content. In other work, Padmanabhan et al. [32] examine the evolving face of Internet shutdowns and censorship in Burma (Myanmar) in the days preceding, during, and after the military coup in February 2021. A report by OONI details the anomalies they detected in Tor network connections in Russia [40] beginning in December 2021, which indicated a sudden shift to widespread blocking of the Tor anonymity network by many Russian ISPs in the lead up to Russia's invasion of Ukraine.

Anti-censorship tools aim to provide journalists, activists, and individuals from marginalized groups with access to the free and open Internet. The Tor [11] anonymity network allows users to anonymously browse the Internet and has proven to be an important tool to evade Internet censorship and surveillance. Unfortunately, Tor's widespread popularity as a censorship circumvention tool makes it an obvious target of blocking by censors [18, 20, 24, 40]. Where access to Tor and the wider Internet is heavily censored, *bridges*, or anti-censorship proxies, must be used to connect. The Tor Project's existing bridge distribution mechanism, BridgeDB [34], provides adequate support for some users; for example, use of Tor bridges in Russia rose sharply at the above December 2021 blockage, and rose further in late February 2022 [35]. However, passive and active detection techniques such as traffic flow analysis [2], Deep Packet Inspection (DPI) [38], website fingerprinting [5], and active probing, have been demonstrated in prior work to reveal Tor bridges [14, 18, 38], making Tor inaccessible for the vast majority of users in some regions [14, 24].

While there has been significant innovation in the development of probe-resistant bridges [3, 4, 31, 36] and transports [10, 16, 19, 21, 39] that are effective at circumventing censors' detection techniques, the problem remains of how to distribute these bridges to untrusted users that will not make them vulnerable to widespread blocking by censors. Prior work on bridge distribution has addressed this inherent conflict from several different angles.

Douglas et al. [13] devise the Salmon scheme, which uses a trust level hierarchy to extend and limit privileges based on user behaviour within the system. Trust levels can be inherited through being invited to the system by a highly trusted user as well as accrued over time while bridges that a user knows of remain unblocked. While Salmon's trust levels are intriguing, the Salmon scheme's use of Facebook to verify users and their tracking of the recommendation graph fails to provide robust privacy guarantees to users and makes the distribution server a prime target for an adversary who desires information about bridge users and their social network. In Salmon's scheme, the use of a third party to authenticate and verify users is critical to limiting the censor's ability to create multiple accounts which can be used to flood the system with malicious new users, ensuring that genuine users are never able to gain trust.

Conversely, Wang et al.'s rBridge [37] and Lovecruft and de Valence's Hyphae [27] both focus on the importance of privacy

Proceedings on Privacy Enhancing Technologies YYYY(X), 1–16

https://doi.org/XXXXXXX.XXXXXXX

protection in bridge distribution and each present a system, differing in their anonymous credential scheme, that protects user privacy and anonymity while disseminating bridges through a network of trusted users with records of good behaviour. Hyphae points out that because the bridge authority can act as both the issuer and verifier of credentials, a much simpler and more efficient anonymous credential scheme, Chase et al.'s keyed algebraic MAC credentials [7], can be used in place of rBridge's k-TAA [1] credentials that require elliptic curve pairings. To limit censors' sock-puppet accounts, these systems both imagine invite-only systems that are only available to users invited by trusted friends, limiting accessibility to new, genuine users.

In this paper, we present Lox, a new bridge distribution system that provides a framework for combining Salmon's trust levels with a formalized privacy protective reputation scheme, emphasizing privacy of a user's social graph. We follow the lead of Hyphae to design our bridge distribution system with the use of Chase et al.'s keyed algebraic MAC anonymous credentials. By addressing known threats to existing systems and combining insights from prior work, we create a system that limits malicious behaviour, provides privacy protection to users and their social graphs, and is open to all users. Our contributions can be summarized as follows:

- We design **Lox**, a new **bridge distribution system** that:
  - resists bridge enumeration by formalizing a **trust level scheme** that is **compatible with anonymous credentials**
  - provides **privacy protection for users and their social graphs** through the use of an unlinkable multi-show anonymous credential scheme
  - makes **bridges available** to untrusted users while **leveraging users' trust networks** to provide trusted access to users invited by a trusted user
- We **evaluate the performance of Lox's protocols** with a Rust implementation, specifically measuring the cost of Lox's privacy protection and demonstrating that Lox achieves reasonable performance for small bridgepools (~3600 bridges, which is comparable to Tor's current bridgepool)

2 BACKGROUND

2.1 Bridge Distribution Problem

The bridge distribution problem, in its simplest construction, focuses on distributing viable network bridges to honest users in the presence of a censor that aspires to prevent bridge use. As explored in previous work [13, 27, 30, 37], designing and implementing such a system for practical use requires an understanding of how censors operate in different regions. This includes the resources at a censor's disposal, the tools they are known to deploy and the known dangers to individuals who are discovered using censorship circumvention tools. In designing Lox, we prioritize safety and accessibility for users under even the most severe conditions. In this section we discuss features from prior work that we have incorporated into Lox.

2.2 Bridge Availability

We consider bridge availability to refer to the ability of a given user to obtain and use a bridge. In Lox, our design includes the following considerations to provide bridge availability comparable to existing systems.

2.2.1 Open While Defending Against Sock-Puppets. Sock-puppet accounts, used to impersonate users in order to enumerate and block bridges, are an ever-present problem in the bridge distribution space. Providing anonymity to users complicates many existing approaches to curb this behaviour. Relying on third-party email providers to verify users was demonstrated by Ling et al. [25] to leave Tor's BridgeDB vulnerable to censors able to make multiple accounts. While there are no perfect solutions to preventing sock-puppets in bridge distribution systems, prior work offers some suggestions to dissuade or limit the number of sock-puppet accounts that are able to access their systems.

rBridge and Hyphae are only open to trusted users and their friends, making these systems inherently more robust against censors, at the cost of utility for users not well socially connected. Sock-puppet accounts can still cause considerable damage if they find paths into these systems. However, their reputation systems limit the amount of damage a censor can do, forcing censors to make a tradeoff between disrupting a relatively small number of users immediately and keeping bridges open longer to enumerate more bridges or invite more malicious users.

Taking a different approach, Salmon recommends only allowing the registration of users with valid Facebook accounts, created some number of years ago and with profile pictures showing the user's face. This makes account duplication more difficult than in BridgeDB's approach. Since Salmon relies on the ability to ban users permanently, being able to trust some external body to verify that a user is not a sock-puppet is critical to the functionality of the Salmon design. Once granted entry, users are tracked as they move through Salmon's deterministic levels of trust to unlock different privileges. Leaving aside privacy concerns about Facebook [22] and the ease with which a state-level censor could create multiple fake accounts, relying on *any* centralized body to verify a user's identity, whether social media, government agency or otherwise, introduces challenges that can cause disproportionate harm to some users over others and sow distrust among the most vulnerable users.

In Lox, we prioritize protection of the user's social graph, opting for a more private and less restrictive means for allowing new, untrusted users to join the system. To protect against sock-puppet attacks, we incorporate features from prior work such as reputation systems and bridge distribution by trusted friends, which we expect would improve over BridgeDB. We also note that while Lox does not depend on account validation, Lox's design does not preclude a valid account check prior to distributing open-entry tokens that, unlike Salmon, could remain separate and unlinkable from future interactions with Lox. Additionally, trusted users, joining by invitation, can gain access to Lox without any need for proving access to a valid third-party account.

2.2.2 Leveraging Trust Networks for Bridge Distribution. Distributing bridges by leveraging a user's trust network has been proposed as a means of gaining an advantage over a sophisticated censor that has more resources than any genuine user, but has few honest friends [27, 29, 37]. Invitations from trusted users give some means of distinguishing between probable honest users and censors, making the system more resistant to enumeration. In a system that is

open to *all* users, the ability to make this distinction is even more valuable. Bootstrapping the system with some number of trusted users that are able to invite friends can allow for the number of trusted users of the system to grow more quickly than would be the case if trust levels alone were used. Additionally, ensuring that the bridges distributed to trusted users are not distributed *except* by trusted invitation reduces the likelihood that those bridges will be impacted by censors.

One of the first schemes that looked at trust networks as a bridge distribution channel was Proximax, introduced by McCoy et al. [29]. Proximax optimizes for system uptime by monitoring the user-hours of highly trusted (registered) users and their invitees who are connected in a tree-like structure. If malicious behaviour is detected anywhere along the branch, the whole branch is considered suspicious, damaging the reputation of all users in that branch and curtailing their ability to distribute invitations.

Wang et al. [37] built on the Proximax scheme with rBridge, which similarly leverages a user's social network for bridge distribution and as the primary contribution, adds privacy protection for users and their social graph through the use of anonymous credentials. rBridge makes several additional design changes to Proximax's trust distribution scheme to limit malicious behaviour and increase the amount of time a bridge can be used before it is blocked. Users join the rBridge system through invitations and accrue credit by keeping their bridges unblocked. When presenting an invitation to the bridge authority (that acts as the issuer and verifier of credentials), users first receive their bridges through oblivious transfer with the authority so that the authority does not learn which bridges are being requested. The user then requests their user credential, and can begin accruing credit. When one of the user's bridges is blocked, they can report it and accrue unclaimed credit up until the blockage. If they have enough credit, they are able to exchange the credit for a new bridge. This allows users to build trust with good behaviour and avoid being locked out of the system permanently. Highly trusted users can request invitation tokens to invite their friends to the system but rBridge's bridge authority will only grant the request with a certain probability in order to encourage users to hand out invitations sparingly, to only their most trusted friends.

Lovecruft and de Valence's Hyphae [27] leaves much of the rBridge protocol intact while replacing the anonymous credential scheme with keyed-verified algebraic MACs [7]. However, they omit the initial oblivious transfer, having the authority issue bridge tokens that users can immediately use to exchange for new bridges anonymously. They also propose reducing the amount of time required for a user to begin inviting friends to the system and allowing users to use their accrued trust credits to exchange for invitations much like they can exchange for new bridges if they are blocked.

Both rBridge and Hyphae operate under the assumption of an invite-only environment where the vast majority of users are not malicious. This makes their systems much more robust against censors, with invitations less likely to be distributed to malicious users. However, the exclusivity of the system means that it can only benefit a relatively small, already socially connected group. A system that is open to all users must go further to limit malicious behaviour of censors since a greater percentage of users acting maliciously would quickly overwhelm these systems.

Douglas et al.'s Salmon [13] scheme also includes distributing bridges through invitations from trusted users. As a user increases their trust level, the bridges they know also gain in reliability and are removed from the pool of bridges distributed to untrusted users. When a user becomes eligible to invite friends, they are invited to the same bridges at an elevated trust level. This encourages users to be cautious when handing out invitations and limits a malicious user's ability to enumerate bridges through gaining trust and inviting themselves. As long as no malicious users were added to a bridge before it stopped being distributed to untrusted users, the bridge can become immune from the malicious behaviour of a censor. Salmon also tracks a user's *suspiciousness*, which is increased each time a bridge they know about is blocked. When a user's suspicion threshold has been reached, the user is banned from the system. While Salmon's use of trust levels and suspicion helps to limit malicious behaviour among a less trusted pool of users, it comes at the cost of limited privacy protections provided to users, who are monitored in order to adjust trust and suspicion assignments.

Trust networks provide an effective means of distributing bridges that inherently disadvantages censors. However, connecting users through their social graph may introduce a new target to censors: attempting to learn those very social graphs of bridge users.

2.3 Protecting the Social Graph

Douglas et al. [13] narrowly define an adversarial censor as one that simply blocks access to bridges rather than one that seeks to identify and persecute users and members of their social network. Using this threat model, Salmon tracks the invitees of trusted users and stores a subset of each user's social graph. The authors argue that a state-level censor would gain little information about a user's social network from Salmon server logs compared to what they would be able to learn from their access to state-connected social media sites and other surveillance tools at their disposal. While we concede that a sophisticated and determined censor may be able to undermine most privacy-preserving mitigations of a bridge distribution system, it does not follow that privacy protection for users and their social networks should be abandoned altogether. Since censorship strategies vary greatly across regions and can change over time, it is impossible to know how a censor might uniquely use server logs or a breach of the system that provides clear evidence of individuals working together to evade censorship.

In Lox, we consider a censor that may seek out and punish censorship circumvention system users and their networks if they can be identified. We thus strive to make it impractical, if not impossible, for a censor to identify users' friend groups even by compromising the bridge distribution server during operation or by confiscating information stored on it. As such, the bridge distribution server must not learn, store, or log identifiable or linkable information about the social graphs of users. Our threat model in Section 3.1 expands upon the protections of the social graph Lox provides, noting that social networks could be reconstructed outside of Lox by a censor that watches connections to particular bridges.

Anonymous credentials combined with other cryptographic primitives such as Pedersen commitments [33], ElGamal encryption [17] and zero-knowledge proofs [6], can provide annonymity to users of a bridge distribution system, connecting over a Tor circuit, and effectively protect their social graph with unlinkable multi-show credentials. Hyphae [27] makes use of multiple anonymous credentials and tokens, each able to support multiple attributes that can be revealed to or hidden from the issuer to maintain privacy and unlinkability while making requests. This makes the bridge authority a less interesting target to a censor hoping to learn about bridge users and their social networks.

Using Chase et al.'s anonymous credential scheme, in Lox, we integrate Salmon-like trust levels and suspicion as attributes of an anonymous credential scheme similar to Hyphae's. We use the trust level as a proxy for time spent as a user in the system where bridges remain unblocked, and increase a user's suspicion when a trusted user's bridges are blocked. A single Lox credential can contain all attributes needed to maintain a user's reputation and the attributes taken together make up a user's reputation and control their access to privileged operations.

2.4 Bridge Enumeration Resistance

The bridge distribution problem measures effectiveness against a censor in two ways. The first is through maximizing the number of days a censor would need to keep a bridge they learn about unblocked in order to gain additional advantages within the system. The second is the degree to which a design choice benefits users and disadvantages censors. Prior work has identified and employed different techniques to limit malicious behaviour and maximize effectiveness against censors.

Nasr et al. [30] take a game-theoretic approach to the bridge distribution problem, finding that a bootstrapping period, during which only trusted users are invited to use the system, is essential to counter the optimal censor strategy. In line with this finding, the systems we discussed above already involve restricting access to trusted groups of users or else verifying their accounts prior to registration [13, 27, 37]. In its most effective simulation, Salmon [13] began their simulation with a small group (20) of users who operated above the trust hierarchy and were able to subsequently invite new highly trusted users. In Lox, we follow a similar bootstrapping period to ensure that trusted users have time to establish groups of users that will retain access to bridges that will remain available regardless of the censor's strategy for acting maliciously toward bridges handed out to untrusted users.

2.4.1 Reputation Systems. As described in the previous sections, prior work on bridge distribution systems each employ some form of reputation system to limit access to functionality that has a higher risk to the system if used maliciously [13, 27, 37]. This forces the censor to make a tradeoff between gaining trust in order to cause greater damage (i.e., learning more trusted bridges to block, inviting more censors to bridges, etc., depending on the constraints of the specific reputation system), and keeping bridges unblocked for longer periods of time, which is necessary to build trust.

rBridge and Hyphae's protocols allow users to accrue credit by keeping their bridges unblocked. These credits can be used to unlock or purchase new bridges or invitations. Salmon tracks each user's *suspicion*, *trust level*, and *recommendation (social) graph*, which collectively help to ensure that trustworthy users and their friends are rewarded with reliable connections to proxy servers and censors are not elevated to trust levels where they can do significant damage to the system, but expose the social graph to the Salmon server. While no specific trust level scheme is fully evaluated in Salmon, extending privileges according to time spent in the system is compatible with rBridge and Hyphae's privacy-preserving credit schemes and can be used to enhance them.

In Lox, we propose a reputation system with trust levels that a user can ascend after (privately) proving a period of time has passed since gaining knowledge of bridges that remain unblocked. Each trust level unlocks greater privileges for users.

2.4.2 Inheritance. We can limit the censor's advantage further by having invited users *inherit* certain properties from their inviters, lest censors invite themselves in an attempt to learn new bridges or acquire new privileges. While no other system uses this term, Salmon [13] already incorporates the concept of inheritance as we have defined it. In Salmon, when a user with a trust level of L invites a friend to the system, the friend learns of the same bridges the inviter knows and enters the system at a trust level of $L - 1$, yielding inheritance of bridges and trust level. To limit a malicious actor from gaining an advantage we can also make suspiciousness inheritable. If a user has seen a blocking event, any user they invite can also be considered suspicious to prevent malicious users from inviting themselves with a clean slate after behaving maliciously.

In Lox, we group some number of bridges into *buckets*; the decryption key for a bucket is one of the inherited attributes in the Lox credential, as is the level of suspicion, indicating the number of blocked bridges a user has seen.

Summary: Lox incorporates each of the features described in this section into a novel privacy-preserving scheme. Lox allows users to build trust over time. Upon reaching a predetermined trust level, users are able to invite a number of friends. When a user is invited by a trusted user, they join Lox at a lower trust level and inherit their inviter's bridges and the number of blocking events they have witnessed. These features are encoded as attributes in a user's Lox credential, which allows the user to remain anonymous through their interactions with the system.

In Table 1, we summarize the contributions from prior work on bridge distribution systems in terms of the different features we have discussed in this section as well as those that we include in the design and implementation of our system Lox, which we will detail in the following sections.

3 LOX CONCEPT

Lox combines the above key features from previous work with novel enhancements in order to provide bridge distribution to socially connected and unconnected users while protecting the social graph even from the central *Lox Authority* server itself.

3.1 Threat Model

We envision our adversary to be a state-level censor with significant resources and two high-level motivations:

(1) to learn of as many bridges as possible in order to block them immediately or at some later time;

Table 1: Summary of Bridge Distribution System Features

Bridge Features	BridgeDB [34]	Proximax [29]	rBridge [37]	Hyphae [27]	Salmon [13]	Lox
Bridge Availability						
Openness	email/https	invitation	invitation	invitation	Facebook account	email/https, invitation
Distribution by Trust Network		✓	✓	✓	✓	✓
Social Graph Protection						
Unlinkable Credentials			k-times	✓		✓
Protects Social Graph	N/A		✓	✓		✓
Blinded Migrations						✓
Enumeration Resistance						
Friends Assigned to Same Bridges		✓			✓	✓
Reputation System		✓	✓	✓	✓	✓
Inheritance					✓	✓
Defined Trust Levels					✓	✓

(2) to learn the social graph of users.

Given that a state-level censor has vastly greater resources and compute power than an individual (or non-profit organization), we accept that it is impossible to completely stop a censor from infiltrating and impacting our system, but we can prevent such a censor from learning the social graphs of Lox users, while keeping users' actions anonymous and limiting the rate at which bridges can be enumerated. We assume that a censor may pose as many genuine users in order to cause disruptions and enumerate bridges in order to block them. We also assume that a censor may attempt to uncover the social graph by gaining access to communication logs between the Lox Authority and Lox users. We acknowledge that other tactics such as traffic fingerprinting and network scanning to discover bridges [15, 28], registering a large number bridges to act as honeypots, and DoS attacks against bridges or the Lox Authority [23] are within the capabilities of many censors and have a significant impact on bridge distribution. However, we consider these problems to be out of scope for the Lox system we present. Instead we aim to make Lox itself resistant to bridge enumeration and ensure that users' social graphs are not revealed by the Lox system itself.

3.1.1 Lox Authority. A fully compromised Lox Authority (LA), which verifies and distributes Lox credentials, could trivially block all bridges in the system, meeting its first motivation. For the purpose of protecting the list of bridges, we therefore assume that the LA is not fully compromised by an adversary. We further assume that the LA has committed to a set of keys for issuing and verifying anonymous credentials and made this commitment publicly accessible. Davidson et al. [9] stress the importance of having the published commitments to keys in a public place, such as the Tor consensus, that is visible to all users and can be used to independently verify the correctness of all tokens. Without such a commitment, the LA could use a different key for each new Lox user to track users' usage of Lox. With this assumption we can assure privacy protection to users (beyond the initial open-entry interaction with the LA) and their social graph against the adversary's second motivation, even if the LA is fully compromised. To successfully distribute bridges, we assume that the LA performs the Lox protocols correctly; however, even if it does not, it will be unable to learn a user's social graph.

Having described our threat model, we describe the elements of our system that work together to preserve the privacy of and ensure unlinkability between users while limiting the ability of a malicious actor to enumerate and block bridges.

3.2 System Goals

We designed Lox with the overall goals of:

(1) **Openness:** where anyone with access to the Internet should be able to access an *open-entry invitation token* for Lox or else receive an invitation from a trusted user.
(2) **Leveraging Trust Networks:** to distribute bridges and allow highly trusted users to invite some number of friends for each trust level increase.
(3) **Privacy and Unlinkability of Users:** by using anonymous credentials [7] to maintain privacy of users and their social graph as they use Lox.
(4) **Enumeration Resistance:** by protecting the bridges in our bridgepool from discovery by a censor.

4 THE LOX SYSTEM

In the design and implementation of our Lox system, we aim to protect the usage patterns of users as well as their social graph through unlinkable, one-show transactions with the Lox Authority. With these considerations in mind, we present our design and implementation details of Lox, using the keyed-verification anonymous credential scheme proposed by Chase et al. [7] incorporating many insights from Lovecruft and de Valence [27], and adding new features of our own. Chase et al.'s anonymous credentials allow for a credential holder to verify attributes of their credentials with an authority arbitrarily many times without revealing their identity or any other unnecessary attributes. Since the LA acts as both the issuer and verifier of anonymous credentials, this is an appropriate scheme to achieve our goals.

4.1 Lox Authority

Lox relies on a single central Lox Authority (LA) that acts as both the issuer and verifier of anonymous credentials and tokens for all Lox protocols. We assume that the LA holds the bridge database, containing hundreds to thousands of individual bridge records (i.e., IP addresses and the secrets required to connect to each of them) that can be leveraged by individuals to connect to the open Internet. We further assume that the LA can verify with bridge operators whether or not a bridge is blocked at any given time. A **Lox open-entry invitation token distributor** is also required for our system but could operate much like Tor's existing BridgeDB [34] to distribute invitation tokens by email, through an https website, or embedded in a website or chat client. We note that the Lox invitation token distributor and Lox Authority could be the same entity.

The LA divides bridges into open-entry and invite-only buckets. We consider that open-entry buckets are much more likely to be distributed to malicious users whereas trusted buckets are less likely to be held by malicious users. To limit a censor's ability to enumerate bridges and increase the probability that any given user's buckets will remain unblocked, open-entry buckets contain a single bridge and invite-only buckets contain three bridges. The LA assigns an ID i to each bucket and maintains a global list of buckets. Each bucket has an encryption key K_i that the LA derives from a hash function (i.e., $K_i = H(LA_{SK}, i)$ where LA_{SK} is a single secret stored by the LA). For each bucket i the LA encrypts the bucket with the key K_i and posts the full list of concatenated encrypted buckets to a public website accessible to users. Since this website is itself susceptible to blocking by a censor, the LA hands out a bridge line, containing the information necessary to connect to a bridge, to users presenting valid invitation requests (along with the initial Lox credential, discussed below). Users are given the bridge key (i, K_i) as an attribute of their Lox credential. This allows them to decrypt the corresponding encrypted bucket to access their bridges. Open-entry buckets can upgrade to invite-only buckets over time. To handle this, the LA pre-groups three open-entry buckets into an invite-only superset bucket with its own unique key. A group of new users will each be assigned to 1 of 3 open-entry bridges in a superset bucket on approximately the same day. When these users become trusted, because enough time has passed without the bridges being blocked, they will all migrate to the superset bucket. This also assumes that over time, new bridges will arrive into the system to serve as open-entry bridges. Additionally, if a bridge in a trusted bucket goes *down*, but is not *blocked*, it can be replaced in the same bucket by the LA to be retrieved by the user, as we explain in Section 4.2.5.

We note that an important aspect of open-entry bridge distribution is the mechanism, like BridgeDB's https/email distribution mechanisms, with which the LA distributes particular open-entry buckets to users requesting them. The LA should be implemented to do this in a way that maximizes the chances of genuine users gaining access to buckets and minimizes the likelihood of censors being given the same buckets as genuine users. As this open-entry distribution mechanism may look different for different systems, we consider this to be orthogonal to Lox and leave this important aspect of open-entry bridge distribution to future work; we discuss it further in Section 6.

4.2 Lox Credentials and Tokens

Anonymous credentials are a versatile cryptographic construction with practical uses in the areas of communication, payment and credential transaction systems. Chase et al.'s anonymous credential scheme [7] includes MAC_{GGM}, a generalization of a MAC presented by Dodis et al. [12], that is proved to satisfy the standard notion of MAC unforgeability (uf-cmva security) in the generic group model (GGM). As in Hyphae, Lox uses the MAC_{GGM}-based anonymous credentials to authenticate collections of attributes.

The LA can verify that a shown credential was authentically issued, but is unable to link it to a particular credential it issued since showings are blinded. Users can utilize the issuer's public key to verify that credentials issued to them are correctly authenticated. We use several types of spend-once credentials that are issued and verified by the LA.

4.2.1 Lox Credential. Lox users hold a Lox credential (Ψ_{Lox}) as a non-rerandomizable (one-show) anonymous credential that is presented for re-issue at every interaction between the user and the LA. It contains the following attributes:

- ***ID*** Φ**:** A random ID (nonce) is jointly created by the user and the LA at Lox Credential issue time such that the LA does not learn the ID of the issued credential, and the user cannot unilaterally select it. When the user presents this credential to the LA at the time of the next interaction, Φ is revealed so the LA can add it to its database of spent IDs. It is then discarded.
- ***Time*** t**:** Lox credentials have a time attribute that marks the time the user last updated their trust level. If the user has just joined the system, the time will be marked with their join date.
- ***Trust level*** L**:** Users receive a trust level attribute that is assigned by the LA when their credential is created. All open-entry users enter Lox with $L = 0$. All invited users enter Lox with $L = 1$. Users that can prove to the LA that a sufficient time period has passed without the bridges in their bucket becoming blocked, can upgrade their trust level through the LA. If a user continues to hold a credential for an unblocked bucket, they can continue to level up to a maximum of $L = 4$ with intermittent level up requests to the LA. Table 2 shows the days required to increase L at each level and what each level of L allows users to do.
  Compared to Salmon [13], users are punished more severely in Lox due to its openness to untrusted and unauthenticated users. If we assume that it is not impossible or particularly challenging for a user to get a new credential, any credential below our migration-eligible threshold would be discarded in favour of a new one.
- ***Bridge bucket*** β**:** Users receive (i, K_i) as an attribute in their credential where i is a bucket ID assigned by the LA and K_i is the encryption key used to encrypt the bucket. Open-entry users receive one bridge in their bucket whereas invited users receive three bridges.

Table 2: User capabilities and invitations for different L values as well as the days required to level up. Users migrating to unblocked buckets are given $L-2$. A user's trust level does not increase beyond $L = 4$. However, users moving up to $L = 4$ receive $a = 6$ invitations. Those that have had $L = 4$ for 84 days or more can receive $a = 8$.

L	Status	Invites a	Upgrade t
0	Untrusted, 1-bridge bucket	0	30
1	Trusted, 3-bridge bucket	0	14
2	Trusted, Invitations	2	28
3	Trusted, Invitations, Migration Eligible	4	56
4	Trusted, Invitations, Migration with Invitations Eligible	6, 8	84

Available invitations a: The invitation countdown counter is an attribute that allows users to issue invitations once they have advanced to $L \geq 2$. The a value is set to the correct value (see Table 2) when the user upgrades their trust level. The value of a is decremented for each invitation issued by the user.

Blockages d: The blockages attribute records the number of times the user has migrated to a new trusted bucket. Open-entry users will always begin with $d = 0$. Users who are invited will inherit the d value of their inviter. Experiencing blockages limits the trust level a user can achieve; 3 and 4 blockages limit the user to trust levels 3 and 2 respectively, after which they will be ineligible to migrate.

We note that in our design of Lox, we have indicated particular thresholds for the L, t, a, and d attributes. The selection of these values started from L and were otherwise selected with the intention of demonstrating how a system can use trust levels to increase user privileges as they use the system. We expect the particular values chosen for t, a, and d will require further optimization and consideration depending on the specifics of the system and threats faced by Lox implementers. We discuss optimization of Lox parameters further in Section 6.

4.2.2 Invitation Token. An open-entry invitation token ($\Omega_{\mathfrak{I}}$) is issued to anyone that is able to request them through one of the methods available to a new Lox user. We assume that Lox users will have the same methods available as users of Tor's BridgeDB [34]: an https website, through email to a specific email address, or to be embedded in a browser such as Tor. However, we note that other distribution methods could be appropriate depending on the deployment. Invitation tokens have their own ID attribute Φ, a time attribute so that they can expire after some period, and an optional attribute that can be used by the LA to select an open-entry bucket from a specific pool of buckets (e.g., if buckets were further divided into email, https, or browser bucket).

Lox incentivizes any new user with friends already using the system with an elevated trust level, to request bridges from these friends. Entering Lox with a trusted Invitation Credential automatically gives the new user more privileges. However, open-entry invitation tokens can fill the gap for users who are unconnected to trusted friends or whose friends have not yet reached a high enough trust level to issue invitations. Invitation tokens aim to ensure anyone that needs a bridge is able to get one and can start building their own trust level immediately even without friends already using the system.

4.2.3 Invitation Credential. Invitation credentials ($\Psi_{\mathfrak{I}}$) can be generated only by trusted users when they achieve a trust level $L \geq 2$ and have invitations remaining (i.e., $a > 0$). An invitation credential contains a hidden invitation ID (Φ), the day (t) that it was generated, and the inviter's bridge bucket key (β) and blockages (d). Proposed invitation credentials are presented by the inviter to the LA and if verified, are signed and issued back to the inviter to be given to a friend. The invite ID ensures that invitation credentials can not be reused once they are redeemed and the day attribute enforces that they expire after 15 days.

4.2.4 Migration Key Credential & Migration Token. Lox allows users to migrate to a new bucket β under some circumstances. In order to maintain the user's privacy and ensure that the LA does not learn either the old or the new β value of the user requesting the migration, all migrations are performed as two-step protocols. There are two types of migration protocols in the Lox system:

(1) **Trust Promotion:** When an open-entry user requests a trust promotion from $L = 0$ to $L = 1$ because their t attribute is set to a date more than 30 days before the current date as shown in Table 2, and the (single) bridge that they know has remained reachable over that time. This allows them to learn of the superbucket that contains their current bridge and two others.
(2) **Blockage Migration:** When a user with a trust level $L \geq 3$ and blockages $d < 4$ requests to be unblocked and the LA can confirm that the bridges in their bucket are blocked. This allows them to learn of a new bucket and re-enter Lox with a trust level $L - 2$ and blockages $d + 1$.

As discussed below, the LA maintains a list of buckets that are eligible for migration. This list is updated daily to include newly blocked buckets and bridges that have been newly added to the database. When a request to migrate is initiated, the LA verifies that the user meets the criteria to migrate for the indicated migration type and if this succeeds, the LA blindly issues a migration key credential ($\Psi_{\mathfrak{M}}$) to the user. The attributes of a migration key credential are a Lox credential ID (Φ) and the bucket the Lox credential is migrating from (β_{FROM}). The LA returns this blindly issued credential (so that the LA does not learn any of the attributes, save that they match those in the blinded Lox credential whose attributes were checked to be eligible for migration as described above), along with an encrypted migration table, freshly encrypted for each request (see Section 4.7 for details).

The user then uses a hash of Φ, β_{FROM}, and the unblinded MAC from the migration key credential to decrypt an entry from the migration table to yield a *migration token* ($\Omega_{\mathfrak{M}}$). Migration tokens have attributes of Φ, the buckets β_{FROM} and β_{TO} the user is migrating from and to, and a flag indicating whether this is a trust promotion or a blockage migration.

In the second phase, the user blindly presents their Lox credential and their migration token (with matching Φ and $\beta = \beta_{\text{FROM}}$), and the LA blindly issues in return a new Lox credential with β set to β_{TO}, and L and d modified appropriately.

Table 3: Lox Protocol Requests: Lox credential attribute options in executions of each protocol by a Lox user, along with what extra credentials or tokens are presented as Hidden, and what new credentials or tokens are generated as the output of the protocol. Note that a new Lox credential is output after every protocol except trust promotion and check blockage, which issue a migration key credential ($\Psi_{\mathfrak{M}}$) that the user can use to decrypt their migration token ($\Omega_{\mathfrak{M}}$). The attribute states for all other credentials can be found in Appendix A.

Protocol	ID Φ	Time t	Trust L	Bucket β	Invites Left a	Blockages d	Extra $\mathcal{H}$	New Output
Open-entry Invitation	-	-	-	-	-	-	$\Omega_{\mathfrak{J}}$ *	$\Psi_{\mathfrak{L}}$
Trust Promotion	$\mathcal{R}$	$\mathcal{H}$	$\mathcal{R}$	$\mathcal{H}$	$\mathcal{R}$	$\mathcal{R}$	-	$\Psi_{\mathfrak{M}} \rightarrow \Omega_{\mathfrak{M}}$
Trust Migration	$\mathcal{R}$	$\mathcal{H}$	$\mathcal{R}$	$\mathcal{H}$	$\mathcal{R}$	$\mathcal{R}$	$\Omega_{\mathfrak{M}}$	$\Psi_{\mathfrak{L}}$
Level Up	$\mathcal{R}$	$\mathcal{H}$	$\mathcal{R}$	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$	$\Psi_{\mathfrak{R}}$	$\Psi_{\mathfrak{L}}$
Issue Invite	$\mathcal{R}$	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$	$\Psi_{\mathfrak{R}}$	$\Psi_{\mathfrak{L}}$, $\Psi_{\mathfrak{I}}$
Redeem Invite	$\mathcal{R}$	$\mathcal{H}$	-	$\mathcal{H}$	-	$\mathcal{H}$	$\Psi_{\mathfrak{I}}$	$\Psi_{\mathfrak{L}}$
Check Blockage	$\mathcal{R}$	$\mathcal{H}$	$\mathcal{R}$	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$	-	$\Psi_{\mathfrak{M}} \rightarrow \Omega_{\mathfrak{M}}$
Blockage Migration	$\mathcal{R}$	$\mathcal{H}$	$\mathcal{R}$	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$	$\Omega_{\mathfrak{M}}$	$\Psi_{\mathfrak{L}}$

* $\Omega_{\mathfrak{J}}$ is revealed, not hidden; the presented Φ for the to-be-issued $\Psi_{\mathfrak{L}}$ is encrypted to make subsequent credentials unlinkable.

4.2.5 Bucket Reachability Credentials. Bucket reachability credentials ($\Psi_{\mathfrak{R}}$) are created by the LA (see Section 4.6 below) and act as a certificate of whether or not the bridges in a particular bucket are accessible to users. Bucket reachability credentials have only two attributes, the current day (t), and the bridge bucket (β) and must be presented to the LA for protocols that require proof of an unblocked bucket (see Table 3). Each bucket with reachable bridges will have its bucket reachability credential placed in the bucket itself (in the encrypted bucket list) each day. Bridge users would download their encrypted bucket (over Tor) every day to obtain their current bucket reachability credential. At the same time, they would learn replacements for any bridges in their bucket that had gone offline (but not blocked).

4.3 Attribute Options

Issuing Time: In prior work [7, 27], credential attributes were either *revealed* ($\mathcal{R}$) to or *hidden* ($\mathcal{H}$) from the issuer by the user at credential issuing time. We add two additional options: the *server-selected* ($\mathcal{S}$) option is used for attributes that are selected solely by the LA such as the bucket attribute β; the *joint* ($\mathcal{J}$) option is used for credential ids Φ. Each side contributes a random component to the attribute, so that the user cannot select the resulting Φ (as in a server-selected attribute), and the LA cannot learn the resulting Φ (as in a hidden attribute), but is assured that it will be random.

Presentation Time: All attributes, no matter how they are issued will either be revealed ($\mathcal{R}$) or hidden ($\mathcal{H}$) from the LA when a credential or token is presented by the user.

4.4 Credential Instantiation

We outline the MAC_{GGM}-based construction of Chase et al. [7] applied to the context of Lox with the LA acting as both the credential issuer and verifier.

Let A, B be generators of a fixed additive group $\mathbb{G}$ of prime order ℓ such that $\log_B(A)$ is unknown. For each credential with n attributes, the LA's secret key is the vector $(\tilde{x}_0, x_0, x_1, \ldots, x_n) \xleftarrow{\$} (\mathbb{Z}/\ell\mathbb{Z})^{n+2}$ and the public key is $(X_0, X_1, \ldots, X_n) = (\tilde{x}_0 A + x_0 B, x_1 A, \ldots, x_n A)$.

To create the tag (P, Q) for a vector of attributes $(m_1, \ldots, m_n) \in (\mathbb{Z}/\ell\mathbb{Z})^n$, the issuer selects $b \xleftarrow{\$} (\mathbb{Z}/\ell\mathbb{Z})^*$ and computes $P \leftarrow bB$, $Q \leftarrow \left(x_0 + \sum_{i=1}^{n} x_i m_i\right) P$. To issue a credential with hidden attributes, the user picks an ElGamal public key D, encrypts the hidden attributes to that key, and creates a zero-knowledge proof of the desired properties of the hidden attributes. The LA can then homomorphically compute the encryption of Q to the key D, even though it cannot learn Q itself. The user decrypts to learn Q. The user rerandomizes the MAC by picking $t \xleftarrow{\$} (\mathbb{Z}/\ell\mathbb{Z})^*$ and setting $(P', Q') = (tP, tQ)$.

To show a credential with hidden attributes, the user presents Pedersen commitments of the hidden attributes, a zero-knowledge proof that the hidden attributes have the desired properties, and a designated-verifier proof (that requires the secret key to check) that the (hidden) MAC is correct.

4.5 Lox Authority Protocols

Having described the components of Lox, we now give a high-level overview of the various interactions a user of our system may have with the LA through each of the Lox protocols.

Open-entry Invitation. Any user can request a Lox credential from the LA by presenting an open-entry invitation with $\Omega_{\mathfrak{J}}.\Phi$ revealed. When a valid open-entry request is presented to the LA and the invitation token $\Omega_{\mathfrak{J}}$ is verified by the LA in zero knowledge, the initial Lox credential $\Psi_{\mathfrak{L}}$ is blindly issued. The attributes presented during an open-entry request are summarized in Table 3.

Attributes in the initial Lox credential are set and issued unilaterally by the LA, excluding the jointly chosen credential ID Φ that ensures subsequent transactions can not be linked to the user.

Trust Promotion and Trust Migration. As discussed in Section 4.2.4, trust promotion from $L = 0$ to $L = 1$ is a two-step protocol, marking the upgrading user's transition from an untrusted user to a trusted user where they gain access to a new trusted superbucket credential with three bridges. When the LA identifies that an open-entry bucket has remained reachable for 30 days, it stops giving that bridge's bucket credential to open-entry users. This will occur just prior to the time when the first untrusted users that were given access to the bucket become eligible to upgrade to $L = 1$. Once this occurs, the only way new users will learn the newly trusted bucket credential is by invitation from someone already using that bucket.

To initiate the Trust Promotion interaction, the user must present a valid Lox credential. We note that the protocol does not require a

bridge reachability credential since if an untrusted bridge becomes blocked, the LA will not place it in the table of eligible trust promotion migrations. The LA verifies the presented Lox credential and all zero knowledge proofs, then checks that Φ was not used before to request a trust promotion, and adds it to the trust promotion specific list of spent tokens but does *not* add it to the used Lox credential spent token list. The LA then returns a migration key credential to the user (described in Section 4.2.4). The user uses their migration key credential to decrypt their migration token ($\Psi_{\mathfrak{M}}$). The user is then ready to make a trust migration request by presenting their migration token along with their current Lox credential for verification and their updated Lox credential with the hidden new bucket attribute matching the migration token's $\Psi_{\mathfrak{M}}.\beta_{\text{TO}}$ for issuing.

If the LA successfully verifies these credentials in zero knowledge, the LA blindly issues the new Lox credential to the user, without having learned which buckets the user had migrated from or to.

Level Up. Lox requires that users actively engage with the LA to upgrade trust levels after a pre-determined interval has passed, as described in Table 2. For example, even if a user has known of their buckets long enough to have achieved the highest trust level, if they have not upgraded past $L = 1$, they will only be eligible to upgrade to $L = 2$, thereby rewarding continued engagement with Lox. A user with $L \geq 1$ must prove to the LA that their Lox credential attribute t is older than some pre-determined number of days (see Table 2) and that the bucket in their Lox credential is reachable at the time of the upgrade by blindly presenting a valid bucket reachability credential for their bucket β. If the credentials are successfully verified by the LA in zero knowledge, the LA issues the new Lox credential with today's date, $L \leftarrow L + 1$ and a set to the appropriate value for the new level in Table 2. That table also lists the advantages that accrue to users at higher trust levels.

Issue Invite. Users with trust levels of $L = 2$ become eligible to issue invitations. Invitations allow untrusted users to enter into the system at an elevated trust level ($L = 1$) and gain knowledge of the same bucket (β) and blockages (d) held by their inviter. Grouping friends in the same bucket encourages the inviter to be cautious in distributing invitation tokens and prevents a malicious user who is able to gain access at a high trust level from enumerating more bridges. To issue an invitation, the user blindly presents their Lox credential with $a > 0$ and a bridge reachability certificate to the LA. If the LA successfully verifies these credentials in zero knowledge, the LA will sign the new Lox credential (with a decremented a) as well as the new invitation credential, and issue them to the user.

Lox's trusted invitations give very little advantage to a censor that keeps bridges open to gain trust since gaining and distributing more invitations does not allow them to enumerate more bridges. Since blockages are inherited, they are also unable to issue an invitation to themselves to hide past malicious behaviour. Thus, if a censor gains access to an invite-only bucket, their most likely action would be to immediately block the bridges. Still, whether they choose to block immediately or during a certain event or crisis, this behaviour only impacts a fraction of the overall system. Genuine users, conversely, may be inclined to share bridge addresses they know with friends outside of Lox, so grouping users in the same bucket matches expected user behaviour patterns and the possibility of gaining access to new bridges in the case of a blockage provides a significant advantage to users over just sharing bridges directly, incentivizing the use of Lox.

Redeem Invite. A new, invited user can create a Lox credential with the bucket β given to them in their invitation credential by blindly presenting their invitation credential to the LA. If the LA successfully verifies their invitation token and proposed new Lox credential in zero knowledge, the LA blindly issues the new credential back to the user.

Check Blockage and Blockage Migration. If a trusted bucket becomes blocked (i.e., two or more bridges in a bucket are blocked), there are limited avenues for trusted users to re-enter the system at an elevated trust level. We assume that the majority of Lox users will need to re-enter the system as untrusted users either through open-entry invitations or else through invitations from friends with knowledge of different buckets. However, users who have achieved $L \geq 3$ are able to migrate to a new trusted bucket. When trusted users migrate, their d attribute increases by 1. When a user reaches $d = 4$ they are no longer able to upgrade their trust level high enough to be eligible to migrate, as described in Section 4.2.1.

Migrating after a blockage is a two-step protocol similar to the Trust Upgrade protocol from $L = 0$ to $L = 1$. A user can initiate migration to another trusted bucket when their bucket becomes blocked by first requesting the check blockage protocol, blindly presenting their Lox credential to the LA. The LA checks that Φ was not used before to request a blockage migration, but does *not* add it to the used token list since users may need to make the request multiple times if the migration table has not yet been updated or if the bridge is just temporarily unavailable.

Upon verifying the credential in zero-knowledge, the LA returns a migration key credential to the user (described in Section 4.2.4). The user uses their migration key credential to decrypt their migration token $\Omega_{\mathfrak{M}}$ and blindly presents this, along with their current and proposed new Lox credential, to the LA with the blockage migration protocol. If the LA successfully verifies these credentials in zero-knowledge, it will sign the new Lox credential and issue it to the user, allowing them to access their new trusted bridges.

All users in a particular bucket that are eligible to migrate will migrate to the same unblocked trusted bucket in order to keep friend groups together. Since Lox does not track which users are connected to one another through invitations, partitioning users of blocked buckets by friend group into new unblocked buckets, as is done in Salmon [13], is not possible.

We provide the details of the attribute options for each of our Lox credentials as they are used across protocols in Appendix A.

4.6 Ancillary Tasks Performed by the Lox Authority

Aside from issuing and verifying credentials, the LA performs a number of tasks to set up the system and provide information about the system.

Bridge Reachability Check. We assume that a reliable checking mechanism exists for bridge operators to determine whether or not their bridges are blocked (and where), and that any blocked bridges that are detected are reported to the LA when a bridge is blocked. The LA collects this data from bridge operators once daily and

generates a bucket reachability credential for each bucket known to it. If two or more of the three bridges in a bucket are unreachable, the bucket is considered to be blocked for the purpose of the $\Psi_{\mathfrak{R}}$ and the date attribute of the token is not updated to the current date. After checking the reachability of each bridge in a bucket, the LA places each $\Psi_{\mathfrak{R}}$ in its corresponding encrypted bucket to be posted with the full list of encrypted buckets described in Section 4.1. Users with credentials for the bucket are able to anonymously retrieve the $\Psi_{\mathfrak{R}}$ for their bucket through a Tor circuit each day.

Maintain a Table of Eligible Migrations. For both trust promotion from trust levels $L = 0$ to $L = 1$ and bucket blockage migration operations after a trusted bucket's bridges are blocked, a user can request to be migrated to a new bucket. This requires that the LA maintain a table of eligible bucket migrations for each of these two operations that can be encrypted and sent to a user when a migration is requested and the presented Lox credential is verified. The trust promotion migration table holds the list of open-entry buckets that have remained unblocked since they appeared in an open-entry bucket, and their corresponding trusted superset buckets. The blockage migration table holds the list of trusted buckets that have become blocked and their corresponding trusted unblocked, replacement buckets that have been held in a reserve of hot spare buckets that have not yet been handed out to users.

Maintain a List of Used Token Nonces. As in Hyphae [27], our system makes use of several spend-once tokens that the LA must maintain in a database of spent token ids for each type of credential and token. At issue time, users will prepare ids (random nonces) that are hidden from the LA in order to be included as attributes of a new credential. The LA adds its own nonce homomorphically to the hidden id attribute, signs the credential and issues it back to the user. At presentation time, the ID is revealed to the LA, that then adds the ID to its database of spent id tokens for the given credential, allowing for unlinkable spend-once tokens.

4.7 Encryption of Migration Tables

Unlike previous bridge distribution proposals, Lox has the unique ability to allow trusted users to learn new bridges after a trust upgrade or blockage event without revealing to the authority which bridges the user previously, or subsequently, knows about. As described in Section 4.6, the LA maintains a table $\langle \beta_{\text{FROM}_i}, \beta_{\text{TO}_i} \rangle_{i=1}^{r}$ of eligible migrations. (In fact, there is one table for trust promotion migrations and one for blockage migrations, but we will just consider one, as they work in the same way.) For the first step in the two-part protocols for trust migration and blockage migration, the user requests a blinded migration key credential ($\Psi_{\mathfrak{M}}$) and an encrypted migration table of all eligible migrations. Recall from Section 4.4 that the issuing of the blinded migration key credential will give to the user a pair $(Pk, Enc_D(Qk))$ for an ElGamal key D chosen by the user, but that Qk will be unknown to the LA. The LA will create a fresh encrypted migration table for each request, as the encryptions will depend on Pk. (This is in fact the most expensive step of the Lox protocols, as we will see later in Table 4.) The user must learn only the β_{TO_i} for the i for which the user's current $\beta = \beta_{\text{FROM}_i}$, and also receive a signed migration token ($\Omega_{\mathfrak{M}}$) containing their current Φ, and that single $(\beta_{\text{FROM}_i}, \beta_{\text{TO}_i})$ pair.

In order to do this, the LA generates a hashtable such that for each i, there will be an entry in the hashtable with key $H_1(\Phi, \beta_{\text{FROM}_i}, Qk_i)$ and value $E_{H_2(\Phi,\beta_{\text{FROM}_i},Qk_i)}(\beta_{\text{TO}_i}, P_i, Q_i)$. Here H_1 and H_2 are hash functions, (P_i, Q_i) is a MAC on the migration token that has attributes $\Phi, \beta_{\text{FROM}_i}$, and β_{TO_i}, and (Pk, Qk_i) is a MAC on the migration key credential that has attributes Φ and β_{FROM_i}. When the user decrypts their Qk, they can compute $H_1(\Phi, \beta_{\text{FROM}}, Qk)$ and use that as the index to the hashtable to find the appropriate encrypted entry. They then compute $H_2(\Phi, \beta_{\text{FROM}}, Qk)$ to use as the decryption key for that entry, to yield the attributes and MAC for the migration token. This satisfies the requirements that only the entry corresponding to the current bucket β_{FROM} of the user with Lox credential ID Φ can be decrypted, even if the migration key credentials are shared with other users, and that the migration token can only be used once. A valid, decrypted migration token with attributes $\Phi, \beta_{\text{FROM}}, \beta_{\text{TO}}$ is required in the second step of the migration protocol when the migration to a new bucket actually occurs. The user migrating either from $L = 0$ to $L = 1$ or else from a now blocked bucket in which they had $L \geq 3$ and $d < 3$ must blindly present their old Lox credential with hidden values Φ and β that is proven in zero knowledge to match the migration token's Φ and β_{FROM}, and an updated Lox credential with a β that matches the Migration Token's β_{TO}.

5 EVALUATION

We developed an implementation of the Lox system to evaluate the practicality of our social graph protective protocols. In this section we present the performance results of running each protocol as well as an evaluation of the performance our system achieved. We consider our system in relation to the existing Tor bridge distribution network, the number of users that are typically served by the existing infrastructure and how the addition of the Lox system would impact this.

5.1 Implementation and Experiment Setup

We implemented our Lox protocols in Rust, available at https://git-crysp.uwaterloo.ca/iang/lox. Each of our experiments was run on a single core of an Intel Xeon E7-8870 at 2.40 GHz running Ubuntu 16.04. Noting that Tor metrics puts the number of active bridges just above 3500 [26], we test the performance of our protocols across bridge pools of various sizes from 900 to 9000 in increments of 900. For each test, the total number of bridges in the bridge pool are divided in half such that half of the bridges are sorted into open invitation buckets and the other half are sorted as hot spare bridges. None of the initial bridges are sorted into trusted-user buckets until users request trust promotions. The request size and time, response size and time, and response handling time for each protocol were averaged over 10 000 runs.

5.2 Results and Evaluation

The results of running each protocol with 3600 bridges total, 1800 open-entry buckets with 1 bridge each and 600 hot spare buckets with 3 bridges each, are displayed in Table 4. We note that for most of the reported protocols, the sizes and times are the same for all

Table 4: Performance statistics per run for Lox protocols initialized with 3600 bridges (1800 untrusted open-entry one-bridge buckets and 600 hot spare three-bridge buckets) over 10 000 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 bridges being blocked as the performance changes accordingly. All times and sizes reported in the table are the same for all bridge pool sizes for all protocols except trust promotion and check blockage, which change linearly with a growing bridge pool.

Protocol	Request Size	Request Time	σ	Response Size	Response Time	σ	Response Handling Time	σ
Open Invitation	332	0.75	0.02	740	3.23	0.03	2.26	0.03
Trust Promotion (0→1)	2 216	9.94	0.04	378 104	364.2	0.3	2.3	0.2
Trust Migration (0→1)	936	4.29	0.03	584	6.50	0.04	2.80	0.03
Level Up (1→ ⋯ →4)	3 368	15.16	0.05	712	15.28	0.05	3.70	0.03
Issue Invitation	1 672	8.00	0.04	1 480	15.04	0.06	7.29	0.05
Redeem Invitation	1 576	7.00	0.04	680	9.09	0.05	3.24	0.03
Check Blockage 5%	744	3.27	0.03	6 404	11.22	0.02	0.25	0.01
Check Blockage 50%	744	3.27	0.03	63 104	64.31	0.09	0.49	0.01
Check Blockage 100%	744	3.26	0.03	126 104	122.5	0.1	0.75	0.02
Blockage Migration	1 224	5.84	0.03	840	9.39	0.04	4.12	0.03

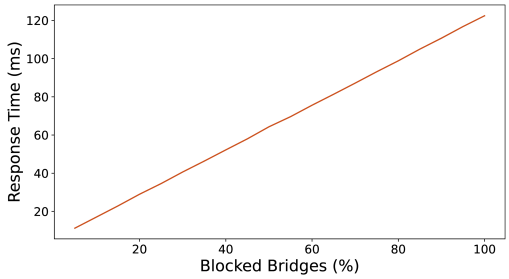

Figure 1: The response time of the check blockage protocol grows linearly with the number of blocked bridges. The std-devs are shaded, but may be too small to see.

problem sets (i.e., bridge pools from 900 to 9000). The exceptions are the trust promotion and check blockage protocols, since both require the Lox authority to compute and send an encrypted hash table of the bridges eligible for migration. For these, the size and time required to receive and handle the response grows linearly with the number of open-entry buckets and the number of blocked bridges, respectively. For example, the server response time for the check blockage protocol is displayed in Figure 1.

We can use our results to analyze the practicality of Lox in terms of the added cost to existing bridge users and to provide estimated system requirements, specifically the number of cores needed for deployment.

The latency considerations for each Lox protocol can be calculated using our performance statistics from Table 4. Each would contribute additional latency and overhead to an existing bridge distribution system such as Tor's BridgeDB on both client and server sides. Lox was designed to impose a minimal burden on bridge operators and users and could be implemented to use existing channels for bridge distribution by replacing directly distributed bridges with open invitations that users can then use to create their Lox credential. This would cause minimal disruption to the usage patterns of current users and bridge operators of a system like Tor's BridgeDB.

Client-side Latency. We find that the latency for individual users is reasonable for all operations. For trust promotion and check blockage protocols, server side response sizes and times are hindered by the hashtable of bridges that must be sent in the migration credential and thus have the highest latency. Trust promotion consistently has the worst response time (364.2 ms) and size (378 kB) but since every Lox user will only make a single trust promotion request and they will have access to an open-entry bridge to receive the response (i.e., a sufficiently high-bandwidth connection), this should not be a significant burden. Below we discuss the number of cores Lox needs to support user requests in greater detail.

The check blockage protocol, in the worst case where 100% of bridges are blocked, has somewhat better performance than trust promotion (response time of 122.5 ms and size 126 kB). However, if the user has lost access to all three of their bridges, they must first acquire a new open-entry bridge in order to run the check blockage protocol without exposing their IP address to the LA, adding to the overall burden of running this protocol. Much more than the trust promotion protocol, the check blockage protocol is likely to see spikes in requests frequency that coincide with bridges being blocked, which could further slow the response time for individual users. A strategic and patient censor could exploit this bottleneck in Lox to slow down or even deny successful blockage migrations to coincide with a particular event, such as an election or planned coup, as described in rBridge [37]. A targeted, long-term strategy like this by the censor would not only limit access to bridges at a critical moment, but could create a significant spike in check blockage and blockage migration requests while users migrate to unblocked bridges. If the censor subsequently blocks the newly-migrated-to bridges, trusted users will be effectively locked out of accessing trusted bridges, which could have a spillover effect on increased open invitation requests as well. Although the scenario described is possible for a determined censor, we note that it would require a censor to gain access to the system and leave bridges unblocked for genuine users for a minimum of 72 days. From Table 4 we can infer that the check blockage protocol is particularly vulnerable to bursts in requests, intensifying with the percentage of blocked bridges. The open invitation protocol is much more robust against bursts in

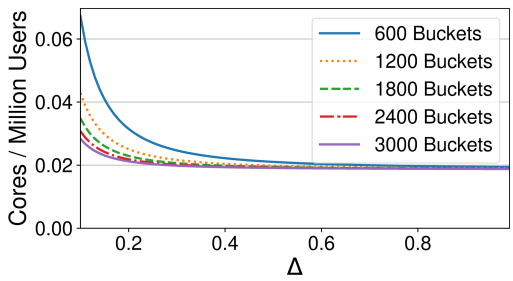

Figure 2: The total number of cores required per million users for varied bridge pool sizes and one bucket being blocked on average every Δ days. Frequent bridge blocking increases the load on the system, but this is mitigated by a larger bridge pool for a constant user base.

usage and thus may be a better stopgap solution for regaining access to bridges in a large-scale attack against Lox's trusted bridges.

In addition to running the Lox protocols themselves, users must periodically download the posted table of encrypted buckets and their reachability credentials. Each encrypted entry in the table is 756 bytes, making a total of 1.4 MB for our evaluated bridge pool.

Overall, we find that the latency added by Lox is reasonable and would not overburden users or bridge operators. The ability for trusted users to migrate to new bridges when their bridges are blocked provides a strong incentive for users to compromise on some latency and overhead increases.

Server-side Load. Lox requires periodic communication between bridge users and the Lox authority. Though a single user may communicate with the LA infrequently (twice in the same day for migration protocols, otherwise with larger gaps between protocols in most cases), we must consider how frequently each protocol may be called across the entire userbase to get a sense of the worst-case load on our system. Among our Lox protocols, the majority will be called at most one time by every new Lox user. However, users that have a trust level of $L = 4$ will be able to level up every 84 days and issue eight invitations over the same period. Using our results from Table 4, we can calculate the CPU cost of the level up protocol and the issue invitation protocol as:

$$\text{Level Up: } 15.27\text{ ms} / 84\text{ days} = 0.18\text{ ms / day}$$
$$\text{Issue Invitation: } 15\text{ ms} \cdot\ 8 / 84\text{ days} = 1.43\text{ ms / day}$$

Check blockage and blockage migration protocols could be requested by trusted users as often as bridges become blocked, potentially creating a significant burden on the system.

Suppose there are b buckets, 1 bucket gets blocked every Δ days, and we remove blocked buckets from the eligible list of migrations after W days. Then there will be W/Δ blocked buckets in steady state, and any given bucket will get blocked every $b \cdot \Delta$ days. A user will run the check blockage and blockage migration protocols every $b \cdot \Delta$ days and it will cost $(5.31 + 0.065 \cdot\ W/\Delta + 9.4)$ ms.

Then, we can calculate the total for check blockage and blockage migration for $W = 365$:

$$(5.31 + 0.065 \cdot\ W/\Delta + 9.4)\text{ ms} / (b \cdot \Delta)\text{ days}$$
$$= (14.71/\Delta + 23.725/\Delta^2)/b\text{ ms / day}$$

Thus, the total cost in steady state is:

$$1.61 + (14.71/\Delta + 23.725/\Delta^2)/b\text{ ms / day}$$

Figure 2 shows the number of cores needed for each $n = 1\,000\,000$ users for varying bridge pools (the load is linear in n). We see that frequent bridge blocking increases the load on the system but this effect is mitigated by the size of the bridge pool as the same number of users are less impacted by blocked buckets if there are more buckets in the pool. Overall, we see that Lox can provide reasonable support for millions of users with a single core even in the worst case where all currently handed out bridges (but not the reserved hot spares) are blocked each day.

6 LIMITATIONS

6.1 Sock-puppet Attacks

The ability for censors to create sock-puppet accounts to enumerate and block bridges is a well-known problem noted in prior work [13, 27, 30, 37]. Unlike prior work that limits bridge distribution to trusted users [27, 37] or relies on a third party to limit duplicate accounts [13] at the expense of protecting the social graph, we provide an open-entry avenue to join Lox with fewer protections. This means that we must contend with a greater number of censor-controlled sock-puppet accounts attempting to gain access to bridges. In the previous sections, we have detailed several features of Lox's design that aim to contribute to enumeration resistance by censors but we have not provided a full evaluation of these features. To summarize, in Lox we propose:

(1) Limiting the number of bridges distributed to new users to a single bridge until the user is eligible to advance to the next trust level.
(2) Limiting a user's privileges until they have spent some configurable amount of time with knowledge of unblocked bridges.
(3) Allowing trusted users to invite friends incrementally.
(4) Allowing trusted users to invite friends only to their bucket.
(5) Using the concept of inheritance to prevent users who have migrated to a new bucket after a blocking from re-joining with an elevated trust level and no record of the blocking event.

While these interventions together limit the effectiveness of sock-puppet attacks, we note that they are still imperfect and do not stop sock-puppets at the entry point to Lox. Using the entry pathways that the Tor project currently uses to distribute Lox open-entry invitations may lead to bridges being almost completely inaccessible in some regions.

Salmon [13] relies on the use of a third party with rigorous sock-puppet protection for its own purposes, to limit sock-puppet accounts at the entry point to their system. With the assumption that most accounts are not censors and that censor accounts can only do limited damage, Salmon is able to offer assurances about the viability of their trust level scheme. However, bridges are most needed in regions where censorship methods are most sophisticated

and restrictive. It is difficult to conceive of a third party in such regions that is unblocked, readily available, and likely to cooperate with a bridge distribution system to limit sock-puppet accounts. Furthermore, limiting sock puppets may be impossible against a third party operating in a censored region where the censor/state can issue any documents or resources required to bypass the check for sock-puppets. Genuine users would therefore either need a bridge to set up an account with a friendly third party or else bridge distribution would be inaccessible, or in the best case, no less accessible than Lox.

Lox takes the position that protection of the social graph should not be sacrificed for the ability to monitor and punish untrusted malicious users and their friend groups since this could have consequences extending beyond the Lox system in the event of a compromise by the censor. In practice, this may be an idealistic position that results in genuine users being unable to access Lox through an open-entry pathway. Still, bootstrapping for trusted users and a system that protects their usage and social networks while rewarding their good behaviour, does not exist in currently deployed bridge distribution systems and we anticipate that this could benefit many users. Lox is intended as a proof of concept system that provides privacy protection and incorporates some of the benefits found in less privacy protective systems such as Salmon. We hope that our design can move the conversation on bridge distribution forward and act as a step towards an acceptable implementation that balances these inherently conflicting issues of protecting user's privacy and preventing sock puppets from enumerating the vast majority of bridges.

6.2 Unoptimized Parameters

Our design of Lox, as well as the implementation and evaluation, include a number of parameters such as the number of bridges that are distributed to untrusted vs. trusted users, the time it takes to upgrade between trust levels, etc. We note here that our parameter selection is meant to be demonstrative, not prescriptive, and is intended to be flexible to meet the needs of any particular deployed system. The stated parameters are an example to demonstrate a system that increases privileges based on a user's trust level. Intentionally, nothing in Lox prevents altering the parameters we have chosen or optimizing them for a specific use case, and changes to Lox parameters will have minimal impact on the performance of our Lox protocols though requiring more frequent requests to the LA will impact server-side load. Furthermore, nothing in Lox prevents incorporating an appropriate algorithm to determine how open-entry bridges are distributed, which we discuss further below.

Our hope with Lox was to show one piece of a deployable bridge distribution system: that protection of the social graph can be reasonably achieved while still incorporating enumeration resistance mechanisms from prior work. In the creating of this piece, it is crucial that we do not prohibit anyone from optimizing the particular details of a suitable deployment further. It therefore remains an open research question as to whether there is a parameter set that sufficiently satisfies all three of the bridge features we identified: bridge availability, social graph protection, and enumeration resistance. We offer our thoughts and insight into extensions that we expect would further improve Lox in Section 7.

6.3 Attacks on Small User Bases

If the set of Lox users is very small, the adversary may be able to infer which user is making a request such as trust promotion by timing when each recipient of an open-entry request becomes eligible for it. To mitigate this, it is important to ensure that a constant stream of users continue to join Lox during bootstrapping and afterward. To bootstrap Lox, open invitations with elevated trust levels should be distributed to some number of highly trusted users for a limited time period prior to public release of open invitations.

7 EXTENSIONS

7.1 Extensions for Bridge Operators

Bridge operators provide the proxy that users of the bridge distribution system can use to connect to the open Internet. Lox was intentionally designed to be compatible with existing bridge distribution systems in order to remove obstacles to adoption. However, some extensions involving significant upgrades to the current protocols run by bridge operators, may help to improve the Lox system further. Bridge tokens could be distributed by a bridge operator for users who are able to prove bridge usage over some period of time each day. The LA could then require some number of valid bridge tokens, along with the other valid Lox credentials for upgrading trust and generating invitations for issuing. This would serve to make it even more challenging for a censor to gain access to trusted functionality without sinking significant time and resources into keeping the bridges up and functioning.

7.2 Extensions for User Privacy

In our implementation and construction of Lox, the protocol that is being run is known to the LA, thereby leaking information about the time a specific protocol is requested and the frequency with which different protocols are run. We could eliminate even this leakage by blinding the protocol that is being requested with a zero knowledge array lookup. This change would require users to send exactly the same credentials and tokens for each request, even if they are no-op tokens. As an example, in our current scheme, a successful check blockage request from the user results in a valid migration token issued by the LA, allowing the user to migrate to a new bucket. To allow the same functionality without the LA learning which protocol was called, the LA would issue a migration token to the user for every request, regardless of whether or not it checks for blocked bridges. The vast majority of migration tokens issued would then be no-op tokens which would simply allow a user to migrate to the same bucket they are already in. If the check blockage protocol is called and bridges are found to be blocked, the migration table would be updated and the LA would issue a true migration token in zero knowledge. In both cases, credential issuance would continue in the same way for all protocols and the LA would not know which protocol was called. These changes would increase the size and computation time required to complete each protocol but this may be an acceptable trade off for the added privacy it would provide.

7.3 Location Based Distribution

Lox could provide location-based bridge distribution by having the LA sort pools of bridges based on their location and then create buckets from the bridges in these pools. When a user requests an open-entry invitation token, they could indicate their region or request bridges from a particular region, which could be encoded as an attribute of the invitation token. When the invitation is presented to the LA, a bucket within the requested region's bridge pool can be prioritized and a (hidden) location attribute can be added to the user's Lox credential for future migrations. The LA must also be able to verify when bridges worldwide are indeed blocked for particular regions, rather than experiencing a temporary outage for reasons other than censorship. This is complicated by locations of bridges and the differing capabilities and behaviours of censors around the world.

7.4 Open-Entry Bridge Distribution

Several parameters in Lox and adjacent to the Lox system are situation and system dependent. The amount of time required to move between trust levels in Lox is configurable and should be adjusted to complement each system's bridge pool, rate of bridge churn, the algorithm used to determine which bridges are distibuted to which users, and the number of users that know about a single bridge. Future work could look at creating an algorithm to distribute open-entry bridges in a way that optimizes for genuine users becoming trusted users and incorporates metrics such as the rate of bridge churn, the number of different channels of distribution, the number of users per bridge group, and the percentage of users that are censors. Research in this direction could help to improve Lox further by increasing the likelihood that new users and not censors tend to occupy new buckets.

7.5 System Rollback and Recovery

To make Lox robust against a mass-blockage event against bridges, the LA should replace blocked bridges in existing buckets with fresh ones, upon discovery of the blockage. Credentials of users who have not migrated will continue to work. For users who (fruitlessly) migrated, the LA could offer an "unmigrate" protocol that keeps users who show/prove their last migration was later than the mass-blockage time in their same bucket, but restores their level/suspicion, or allow reuse of the old credential used to migrate in order to refresh it without changing its non-ID attributes.

8 CONCLUSION

In this paper, we presented Lox, a bridge distribution system that leverages users' trust networks for bridge distribution while protecting the privacy of the users' social graphs, and limits the impact of malicious behaviour on the system.

Lox allows users to advance trust levels towards elevated privileges for proven good behaviour (i.e., bridges remain unblocked). Highly trusted users can invite friends, and migrate to new bridges after a blockage event. In order to protect the privacy of users and their social networks, Lox is implemented using anonymous, unlinkable credentials [7]. Lox also introduces several features to limit the impact of a censor's malicious behaviour. We implemented the Lox protocols in Rust, and measured their performance to show that Lox can provide reasonable protection of the social graph for millions of users with even a single core.

ACKNOWLEDGMENTS

This research was undertaken, in part, thanks to funding from the Canada Research Chairs program, the National Science and Engineering Research Council, an Ontario Graduate Scholarship, and Zonta International. This work benefited from the use of the CrySP RIPPLE Facility at the University of Waterloo.

REFERENCES

[1] Man Ho Au, Willy Susilo, and Yi Mu. 2006. Constant-Size Dynamic k-TAA. In *Security and Cryptography for Networks*. 111–125.
[2] Diogo Barradas, Nuno Santos, and Luís Rodrigues. 2018. Effective Detection of Multimedia Protocol Tunneling Using Machine Learning. In *USENIX Security Symposium*. 169–185. https://www.usenix.org/conference/usenixsecurity18/presentation/barradas
[3] 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 *ACM Conference on Computer and Communications Security (CCS '20)*. 35–48. https://doi.org/10.1145/3372297.3417874
[4] Cecylia Bocovich and Ian Goldberg. 2016. Slitheen: Perfectly Imitated Decoy Routing through Traffic Replacement. In *ACM Conference on Computer and Communications Security (CCS '16)*. 1702–1714. https://doi.org/10.1145/2976749.2978312
[5] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. 2012. Touching From a Distance: Website Fingerprinting Attacks and Defenses. In *ACM Conference on Computer and Communications Security (CCS '12)*. 605–616. https://doi.org/10.1145/2382196.2382260
[6] Jan Camenisch and Markus Stadler. 1997. *Proof Systems for General Statements about Discrete Logarithms*. Technical Report. ETH Zurich.
[7] Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. 2014. Algebraic MACs and Keyed-Verification Anonymous Credentials. In *ACM Conference on Computer and Communications Security (CCS '14)*. 1205–1216. https://doi.org/10.1145/2660267.2660328
[8] Jakub Dalek, Nica Dumlao, Miles Kenyon, Irene Poetranto, Adam Senft, Caroline Wesley, Arturo Filastó, Maria Xynou, and Amie Bishop. 2021. *No Access: LGBTIQ Website Censorship in Six Countries*. Technical Report Citizen Lab Research Report No. 142. University of Toronto. https://citizenlab.ca/2021/08/no-access-lgbtiq-website-censorship-in-six-countries/
[9] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. 2018. Privacy Pass: Bypassing Internet Challenges Anonymously. In *Proceedings on Privacy Enhancing Technologies*, Vol. 2018. 164–180. https://doi.org/10.1515/popets-2018-0026
[10] Roger Dingledine. 2012. Obfsproxy: The Next Step in the Censorship Arms Race. https://blog.torproject.org/obfsproxy-next-step-censorship-arms-race [Online; accessed February 21, 2022].
[11] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium*. 303–320.
[12] Yevgeniy Dodis, Eike Kiltz, Krzysztof Pietrzak, and Daniel Wichs. 2012. Message Authentication, Revisited. In *Advances in Cryptology (EUROCRYPT 2012)*. 355–374.
[13] Frederick Douglas, Rorshach, Weiyang Pan, and Matthew Caesar. 2016. Salmon: Robust Proxy Distribution for Censorship Circumvention. In *Proceedings on Privacy Enhancing Technologies*, Vol. 2016. 4–20. https://doi.org/10.1515/popets-2016-0026
[14] Arun Dunna, Ciarán O'Brien, and Phillipa Gill. 2018. Analyzing China's Blocking of Unpublished Tor Bridges. In *Workshop on Free and Open Communications on the Internet (FOCI'18)*. https://www.usenix.org/conference/foci18/presentation/dunna
[15] Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. 2013. ZMap: Fast Internet-Wide Scanning and Its Security Applications. In *USENIX Security Symposium*. 605–620.
[16] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. 2013. Protocol Misidentification Made Easy with Format-Transforming Encryption. In *ACM Conference on Computer and Communications Security (CCS '13)*. 61–72. https://doi.org/10.1145/2508859.2516657
[17] Taher Elgamal. 1985. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory* 31, 4 (1985), 469–472. https://doi.org/10.1109/TIT.1985.1057074
[18] Roya Ensafi, David Fifield, Philipp Winter, Nick Feamster, Nicholas Weaver, and Vern Paxson. 2015. Examining How the Great Firewall Discovers Hidden

Circumvention Servers. In *Internet Measurement Conference (IMC '15)*. 445–458. https://doi.org/10.1145/2815675.2815690
[19] David Fifield, Nate Hardison, Jonathan Ellithorpe, Emily Stark, Dan Boneh, Roger Dingledine, and Phil Porras. 2012. Evading Censorship with Browser-Based Proxies. In *Proceedings on Privacy Enhancing Technologies*. 239–258.
[20] David Fifield and Lynn Tsai. 2016. Censors' Delay in Blocking Circumvention Proxies. In *Workshop on Free and Open Communications on the Internet (FOCI '16)*.
[21] Sergey Frolov and Eric Wustrow. 2020. HTTPT: A Probe-Resistant Proxy. In *Workshop on Free and Open Communications on the Internet (FOCI '20)*. https://www.usenix.org/conference/foci20/presentation/frolov
[22] Jim Isaak and Mina J. Hanna. 2018. User Data Privacy: Facebook, Cambridge Analytica, and Privacy Protection. *Computer* 51, 8 (2018), 56–59. https://doi.org/10.1109/MC.2018.3191268
[23] Rob Jansen, Tavish Vaidya, and Michael E. Sherr. 2019. Point Break: A Study of Bandwidth Denial-of-Service Attacks against Tor. In *USENIX Security Symposium*. 1823–1840. https://www.usenix.org/conference/usenixsecurity19/presentation/jansen
[24] Andrew Lewman. 2010. China blocking Tor: Round Two | Tor Blog. [Online; accessed February 18, 2022] https://blog.torproject.org/china-blocking-tor-round-two.
[25] Zhen Ling, Junzhou Luo, Wei Yu, Ming Yang, and Xinwen Fu. 2015. Tor Bridge Discovery: Extensive Analysis and Large-scale Empirical Evaluation. *IEEE Transactions on Parallel and Distributed Systems* 26, 7 (2015), 1887–1899. https://doi.org/10.1109/TPDS.2013.249
[26] Karsten Loesing, Steven J. Murdoch, and Roger Dingledine. 2010. A Case Study on Measuring Statistical Data in the Tor Anonymity Network. In *Workshop on Ethics in Computer Security Research (WECSR 2010)*.
[27] Isis Agora Lovecruft and Henry de Valence. 2017. Hyphae: Social Secret Sharing. [Online; accessed February 21, 2022] https://patternsinthevoid.net/hyphae/.
[28] Srdjan Matic, C. Troncoso, and Juan Caballero. 2017. Dissecting Tor Bridges: A Security Evaluation of Their Private and Public Infrastructures. In *Network and Distributed System Security Symposium*.
[29] Damon McCoy, Jose Andre Morales, and Kirill Levchenko. 2012. Proximax: A Measurement Based System for Proxies Dissemination. In *Financial Cryptography and Data Security*. 260–267.
[30] Milad Nasr, Sadegh Farhang, Amir Houmansadr, and Jens Grossklags. 2019. Enemy At the Gateways: Censorship-Resilient Proxy Distribution Using Game Theory. In *Network and Distributed System Security Symposium*.
[31] Milad Nasr, Hadi Zolfaghari, and Amir Houmansadr. 2017. The Waterfall of Liberty: Decoy Routing Circumvention That Resists Routing Attacks. In *ACM Conference on Computer and Communications Security (CCS '17)*. 2037–2052. https://doi.org/10.1145/3133956.3134075
[32] Ramakrishna Padmanabhan, Arturo Filastò, Maria Xynou, Ram Sundara Raman, Kennedy Middleton, Mingwei Zhang, Doug Madory, Molly Roberts, and Alberto Dainotti. 2021. A Multi-Perspective View of Internet Censorship in Myanmar. In *Workshop on Free and Open Communications on the Internet (FOCI '21)*. 27–36. https://doi.org/10.1145/3473604.3474562
[33] Torben Pryds Pedersen. 1992. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Advances in Cryptology (CRYPTO '91)*. 129–140.
[34] The Tor Project. 2022. BridgeDB. [Online; accessed February 18, 2022] https://bridges.torproject.org/.
[35] The Tor Project. 2022. Users — Tor Metrics. https://metrics.torproject.org/userstats-bridge-country.html?start=2021-09-01&end=2022-02-27&country=ru. [Online; accessed February 27, 2022].
[36] Benjamin VanderSloot, Sergey Frolov, Jack Wampler, Sze Chuen Tan, I. C. Simpson, Michalis Kallitsis, J. Alex Halderman, Nikita Borisov, and Eric Wustrow. 2020. Running Refraction Networking for Real. In *Proceedings on Privacy Enhancing Technologies*, Vol. 2020. 321–335.
[37] Qiyan Wang, Zi Lin, Nikita Borisov, and Nicholas J. Hopper. 2013. rBridge: User Reputation Based Tor Bridge Distribution with Privacy Preservation. In *Network and Distributed System Security Symposium*.
[38] Philipp Winter and Stefan Lindskog. 2012. How the Great Firewall of China is Blocking Tor. In *Workshop on Free and Open Communications on the Internet (FOCI '12)*. https://www.usenix.org/conference/foci12/workshop-program/presentation/Winter
[39] Philipp Winter, T. Pulls, and Jürgen Fuß. 2013. ScrambleSuit: A Polymorphic Network Protocol to Circumvent Censorship. In *Workshop on Privacy in the Electronic Society*.
[40] Maria Xynou and Arturo Filastó. 2021. Russia started blocking Tor. [Online; accessed February 18, 2022] https://ooni.org/post/2021-russia-blocks-tor/.

A OVERVIEW OF ATTRIBUTE OPTIONS FOR LOX CREDENTIALS AND TOKENS

Having described the different attribute options utilized by the Lox system, we summarize the attribute options used for each attribute in all transactions between the user and LA in Tables 5–9, with the details in the following subsections.

A.1 Lox Credential Attribute Options Across Protocols

Lox credentials are initially issued by the LA with either an open invitation or an invitation from a trusted user. In both cases, the user must present their blinded credential id (Φ) to be contributed to and signed by the LA and then included as part of the issued credential. When the Lox credential is next presented to the LA, the credential id (Φ) is revealed to the LA so that it can be added to a database of seen credential ids, making the credential invalid for subsequent showings. The tables above show the attribute options for each protocol for issuing operations by the LA (Table 5) and credential presentation by the client (Table 6).

A.2 Invitation Credential Attribute Options

Trusted invitation tokens $\Psi_{\mathfrak{I}}$ (Table 7) can be generated by users with a trust level $L \geq 2$. To ensure that invitations can both be verified by the LA as valid and remain unlinkable to the user that requests them, the invitation id attribute $\Psi_{\mathfrak{I}}.\Phi$ is jointly issued and revealed by the invitee when requesting their Lox credential. The bucket β and blockage d attributes are hidden both at issue and claim time so the LA never learns the bucket of the inviter or invitee.

A.3 Bucket Reachability Credential Attribute Options

Bucket reachability credentials $\Psi_{\mathfrak{R}}$ (Table 8) are made available and refreshed each day by the LA with only the date t and bucket β as attributes. Users always present bucket reachability credentials along with their existing Lox credential to prove the liveness of their bucket β. Both β attributes are hidden from the LA but are able to be proved equal in zero knowledge.

A.4 Migration Token Attribute Options

As discussed above, the Migration Token $\Omega_{\mathfrak{M}}$ (Table 9) is created with the credential ID Φ and bucket β_{FROM} from the user's Lox credential and the decrypted migration table value β_{TO}. When presenting the Migration Token to the LA, Φ will be revealed with all β attributes hidden. However β_{FROM} will be proven in zero knowledge to match the user's presented $\Psi_{\mathfrak{L}}.\beta$ from their existing credential and β_{TO} will be proven in zero knowledge to match the user's prepared $\Psi_{\mathfrak{L}}.\beta$ (to be migrated to).

Table 5: Lox Credential Issuing: Lox credential attribute options in issuing operations by the LA. Note that trust promotion and check blockage protocols issue a migration token and so are not included in this table. These can be found in Table 9.

Name	Description	Open Invite	Trust Migration	Level Up	Issue Invitation	Redeem Invitation	Blockage Migration
$\Psi_{\mathfrak{L}}.\Phi$	ID	$\mathcal{J}$	$\mathcal{J}$	$\mathcal{J}$	$\mathcal{J}$	$\mathcal{J}$	$\mathcal{J}$
$\Psi_{\mathfrak{L}}.t$	Time	$\mathcal{S}$	$\mathcal{R}$	$\mathcal{R}$	$\mathcal{H}$	$\mathcal{S}$	$\mathcal{R}$
$\Psi_{\mathfrak{L}}.L$	Trust	0	1	$\mathcal{R}$	$\mathcal{H}$	1	$\mathcal{H}$
$\Psi_{\mathfrak{L}}.\beta$	Bucket	$\mathcal{S}$	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$
$\Psi_{\mathfrak{L}}.a$	Invites Left	0	0	$\mathcal{S}$	$\mathcal{H}$	$\mathcal{S}$	$\mathcal{S}$
$\Psi_{\mathfrak{L}}.d$	Blockages	0	0	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$

Table 6: Lox Credential Presentation: Lox credential attribute options in presentation transactions with the LA. Note that open-entry invitation and redeem invitation protocols are not included in this table. This is because they do not involve presenting Lox credential attributes. Redeem invitation attribute options can be found in Table 7.

Name	Description	Trust Promotion	Level Up	Issue Invitation	Check Blockage	Migration
$\Psi_{\mathfrak{L}}.\Phi$	ID	$\mathcal{R}$	$\mathcal{R}$	$\mathcal{R}$	$\mathcal{R}$	$\mathcal{R}$
$\Psi_{\mathfrak{L}}.t$	Time	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$
$\Psi_{\mathfrak{L}}.L$	Trust	0	$\mathcal{R}$	$\mathcal{H}$	$\mathcal{R}$	$\mathcal{R}$
$\Psi_{\mathfrak{L}}.\beta$	Bucket	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$
$\Psi_{\mathfrak{L}}.a$	Invites Left	0	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$
$\Psi_{\mathfrak{L}}.d$	Blockages	0	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$

Table 7: Credential: Invitation Credential

Name	Description	Issue	Redeem
$\Psi_{\mathfrak{I}}.\Phi$	Invitation ID	$\mathcal{J}$	$\mathcal{R}$
$\Psi_{\mathfrak{I}}.\beta$	Bucket	H	$\mathcal{H}$
$\Psi_{\mathfrak{I}}.d$	Blockages	H	$\mathcal{H}$

Table 8: Credential: Bucket Reachability Credential

Name	Description	Presentation
$\Psi_{\mathfrak{R}}.t$	Time	$\mathcal{R}$
$\Psi_{\mathfrak{R}}.\beta$	Bucket	$\mathcal{H}$

Table 9: Credential: Migration Token

Name	Description	Presentation
$\Omega_{\mathfrak{M}}.\Phi$	Lox credential ID	$\mathcal{R}$
$\Omega_{\mathfrak{M}}.\beta_{\text{FROM}}$	From Bucket	$\mathcal{H}$
$\Omega_{\mathfrak{M}}.\beta_{\text{TO}}$	To Bucket	$\mathcal{H}$