

Anonymous Communication within Highly Repressive Regions

by

Sina Kamali

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2025

© Sina Kamali 2025

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

A portion of this thesis incorporates material from a conference paper [125] for which I was first author. Some content, particularly in the introduction (Section 1.2.1) and the chapter discussing blackout-resistant technologies (Chapter 2), is taken verbatim. Another portion of this thesis (Section 1.2.2 and Chapter 3) incorporates material included in a draft manuscript, for which I am the first author, and that is being prepared for an upcoming submission. This corresponds to the discussion on a new Web-based internet censorship circumvention tool.

Abstract

Repressive governments are increasingly resorting to restricting the flow of information during times of political unrest or civil resistance. In recent years, this restriction has taken new forms, expanding beyond traditional censorship tactics such as IP and DNS blocking to the use of Intranets (internal networks) and complete internet blackouts. To address this, many tools have been proposed to support users during such bleak times. However, most of these tools either overlook the importance of user anonymity or underestimate the lengths to which a censor might go to de-anonymize users.

In this thesis, we propose two new tools to help users reclaim their freedom and privacy under these restrictive conditions. First, we present Anix, an anonymous, blackout-resistant microblogging app that supports message endorsing, allowing users to avoid falling victim to misinformation, even without internet access. Second, we introduce Huma, a censorship circumvention tool that operates via web protocol tunneling and leverages deferred traffic replacement to counter a wide range of adversarial attacks. Additionally, Huma can function as a blocking-resistant anonymous communication system within Intranets. Together, these tools offer secure and private solutions for users facing the aforementioned restrictive scenarios. We implement and evaluate both tools to demonstrate their practicality and release our code as an open-source contribution to the community.

Acknowledgements

I would like to express my deepest gratitude towards my mother for supporting me throughout my academic career. I would also like to thank my partner, for being a wall I could lean against in the tough times we went through over the two years of my master's program. Finally, I would like to thank all of my friends, who were there for me when I wanted to give up.

Dedication

I dedicate my research and this thesis to all the people from around the world who are affected by any form of censorship, especially the brave people of Iran.

Table of Contents

Author’s Declaration	ii
Statement of Contributions	iii
Abstract	iv
Acknowledgements	v
Dedication	vi
List of Figures	x
List of Tables	xi
1 Introduction	1
1.1 Previous Endeavors	2
1.1.1 Internet Blackout	2
1.1.2 Internet Censorship	3
1.2 Our Solutions	4
1.2.1 Anix: Anonymous Blackout-Resistant Microblogging with Message Endorsing	5
1.2.2 Huma: Censorship Circumvention via Web Protocol Tunneling with Deferred Traffic Replacement	6

2	Anix: Anonymous Blackout-Resistant Microblogging with Message Endorsing	8
2.1	Blackout-Resistant Messaging Apps	8
2.1.1	Communication Models	9
2.1.2	Anonymity Guarantees	10
2.1.3	Trust Systems	12
2.1.4	Revocable Identities	13
2.2	Threat Model	14
2.3	Anix	16
2.3.1	Overview of the Microblogging Workflow	16
2.3.2	Pseudonyms and Identity Management	18
2.3.3	Trust Management	21
2.3.4	Message Endorsing and Ranking	26
2.3.5	Point-to-Point Message Exchange	28
2.3.6	Anix Message Formats	31
2.4	Evaluation Methodology	32
2.4.1	Evaluation Goals and Approach	33
2.4.2	Experimental Testbed	33
2.4.3	Rationale on Simulation Parameters	34
2.5	Evaluation Results	37
2.5.1	Micro-benchmarks	37
2.5.2	Simulations	38
2.5.3	Comparing Rangzen’s and Anix’s Resilience	40
2.6	More Stringent Adversarial Settings	40
2.7	Related Work	42

3	Huma: Censorship Circumvention via Web Protocol Tunneling with De-ferred Traffic Replacement	46
3.1	Intermediary-based Circumvention Tools	46
3.1.1	Traffic Analysis Resistance	47
3.1.2	Safeguards against Sybil Proxies	50
3.1.3	Deployment and Collateral Damage	51
3.2	Huma	53
3.2.1	Threat Model	53
3.2.2	Core Architectural Components	54
3.2.3	Overview of the Huma Workflow	55
3.3	Details on Huma’s Operations	57
3.3.1	Managing Users and the Infrastructure	57
3.3.2	Achieving Resistance against Traffic Analysis	60
3.3.3	Achieving Resistance against Sybil Proxies	65
3.3.4	Deployment within Intranets	65
3.4	Evaluation	66
3.4.1	Evaluation Goals	67
3.4.2	Experimental Testbed	67
3.4.3	Resistance to Content Fingerprinting	69
3.4.4	Resistance to Traffic Fingerprinting	70
3.4.5	Network Performance	71
3.4.6	Network Performance w/ OUS	75
4	Conclusion	79
4.1	Limitations and Future Work	79
	References	81

List of Figures

2.1	Anix’s coverage in function of message delivery latency.	39
2.2	Anix’s coverage (under active DoS) in function of message delivery latency with increasing proportions of adversarial nodes.	42
2.3	Anix’s coverage and latency in alternative awareness scenarios.	45
3.1	Overview of Huma’s architecture and high-level workflow.	56
3.2	Bootstrapping information shared by the Huma authority.	59
3.3	Deferred processing mechanism applied to double-request receive.	61
3.4	Trade-off on content fingerprinting accuracy and bandwidth overhead (Tranco top-100) when storing responses in differently-sized data chunks.	69
3.5	Pages fetched via Raven and OUStral-based OUSes, with varying page sizes, as a function of available DWs across different user profiles.	76

List of Tables

2.1	Comparison of blackout-resistant communication apps.	9
2.2	Description of each parameter used in Anix’s simulator	35
2.3	Anix’s resilience to misinformation ($Adv = 0.02$)	36
2.4	Computation time (in ms) for Anix’s operations.	38
2.5	# of majorly shown/not shown messages in Rangzen.	41
2.6	Anix’s resilience to misinformation ($Adv = 0.05$).	42
2.7	Anix’s resilience to misinformation ($Adv = 0.10$).	43
2.8	Anix’s resilience to misinformation ($Adv = 0.25$).	44
3.1	Categorization and classification of censorship circumvention tools related to Huma.	47
3.2	Classifier accuracy and KS similarity across client locations	71
3.3	Huma’s latency and bandwidth overheads for the India client with different DW and target website page size combinations.	72
3.4	Huma’s latency and bandwidth overheads for the US client with different DW/target website page size combinations.	74
3.5	Huma’s latency and bandwidth overheads for the German client with different DW/target website size combinations.	75
3.6	OUS user profiles, and their average statistics per day, extracted from the data provided by Kulshrestha et al. [144].	77
3.7	OUStral user profiles w/ average stats per day, created by simulating OUS-tral using the activity times from Table 3.6.	78

Chapter 1

Introduction

Repressive governments tend to resort to restricting information flow in times of political unrest or civil resistance. While previously the most commonplace method used to control communications in such times was internet censorship [166], which restricted users' access to the internet by utilizing techniques such as DNS hijacking, IP block lists, and protocol blocking, in recent years, alternative methods have been used. Apart from aforementioned techniques, some countries have resorted to internet blackouts [64, 55], completely cutting off internet access in order to prevent information from being shared. However, internet blackouts are usually met with heavy pushback by citizens who have businesses and jobs that are bound to use the internet, thus generally lasting only for a few days [184, 39]. In cases where the censor wants to maintain the closed-off space longer than a few days, instances of utilizing Intranets [214, 273] are becoming more frequent. An Intranet acts similarly to an internal internet, an internet without access to any endpoints outside of the censored region. DNS hijacking, IP and protocol blocking, Intranets, and internet blackouts are being used in many countries around the world, sometimes in combination, for the censor to achieve its goals tied to the control of information flow.

To battle these restrictive tactics, citizens have been resorting to tools created to handle such restrictions. For internet censorship, users leverage censorship circumvention tools [225, 133] that allow them to access free internet. Conversely, for citizens affected by internet blackouts and Intranets, access to unrestricted internet is not possible, so the focus shifts to maintaining a secure channel for communications. During internet blackouts, users rely on mesh network-based messaging systems [108, 152], allowing them to exchange messages by utilizing wireless capabilities on mobile devices. However, there does not exist much prior work on accommodations for Intranets, but private messaging systems could serve as an outline of a useful solution to maintain user connections during Intranets.

1.1 Previous Endeavors

In the following sections, we will discuss previous solutions and their shortcomings, before introducing our solutions for each of the aforementioned restriction tactics.

1.1.1 Internet Blackout

To reclaim their ability to communicate freely during shutdowns, users have resorted to delay-tolerant messaging applications based on mobile mesh networks [108]. These applications, which we refer to as *blackout-resistant communication apps*, convey messages in a “gossip” fashion between end-user equipment devices (often smartphones) through wireless technologies such as Bluetooth and Wi-Fi Direct [152], obviating the need for an active data connection to relay messages. For this reason, these apps have become an attractive tool not just for citizens in general, but also for human rights activists interested in sharing information and organizing urban protests [15]. After their popularization during the 2014 pro-democracy outcries in Hong Kong (through FireChat [204]), blackout-resistant communication apps (e.g., Briar [9], Bridgefy [10]) have been consistently improved, scrutinized [16, 17], and used across multiple countries upon the threats of Internet shutdowns [137, 191, 254].

In the context of activism scenarios, protesters should be able to coordinate amongst themselves and/or expose corruption while avoiding the risk of retaliation by part of the censor. Given the important role of blackout-resistant communication apps in fulfilling these goals, we argue that such apps should strive to meet three core requisites. First, the app should allow users to participate in the network *anonymously*, i.e., a message should not be easily tied to a user’s real identity so that users may avoid harassment or prosecution due to dissent. Second, the app should provide a *one-to-many* communication capability to enable the swift sharing of information and facilitate the mobilization of large groups of individuals [186, 198, 279]. Third, since in this communication model any node is allowed to broadcast a message, users should have a way to both gauge the *trustfulness* of the messages they receive and be able to verify their *authenticity*. This would enable users to correctly prioritize messages and safely ignore those introduced by rogue agents; in the past, nation-states have purposely spread misinformation to foil the organization of protests [204, 176] or intimidate citizens [175].

Despite numerous academic and non-academic proposals for blackout-resistant communication apps, their functionalities are limited. After surveying prominent examples (see §3.1), we find that few of them simultaneously address the requisites mentioned above.

For instance, while Moby [192] and ASMesh [37] provide strong anonymity guarantees and enable users to verify the authenticity of messages authored by some trusted contact, they focus on one-to-one communication, making it difficult to disseminate information widely or coordinate large groups. Conversely, some popular apps support one-to-many communication but do not offer anonymity (e.g., Firechat), or fail to offer it despite multiple attempts (e.g., Bridgefy [17]). To date, Rangzen [152] is the only app that supports anonymous one-to-many communication, presenting users with a simple microblogging platform that implements a message prioritization functionality.

Though Rangzen introduces a mutual friendship-based trust scheme for prioritizing messages exchanged within its microblogging platform, this scheme requires users to manually exchange contact information and may fail to reflect most users’ practical expectations. As it stands, there is a significant disconnect between a message’s trust score and the perceived usefulness of the message’s content or the trust deposited in the message’s author itself. Briefly, a Rangzen node prioritizes a message by assigning it a trust score proportional to the number of mutual friends between itself and the node relaying that same message. Unfortunately, this trust scheme may prevent potentially useful messages authored by users outside a close circle of friends from achieving notoriety, potentially impacting the ability of the microblogging platform to raise awareness about specific events. Also, the scheme does not allow a user to gauge how a message is perceived by its own trusted contacts at large (e.g., how many have “upvoted” it).

1.1.2 Internet Censorship

To bypass censorship, users rely on circumvention tools designed to evade blocking mechanisms [266]. Over the years, a wide range of circumvention tools has emerged [225, 133], varying in threat models, underlying evasion techniques, and effectiveness. Most circumvention tools rely on intermediary infrastructure to conceal the true destination of user traffic, allowing access to blocked content by making it appear as legitimate and innocuous communication [177].

Among these, *protocol tunneling* circumvention tools emerged as a compelling alternative. They encapsulate covert data within allowed protocols (e.g., VoIP [116], video [153], games [107], e-mail [277], web [93]) and focus on adhering to legitimate traffic patterns, making it difficult for censors to identify circumvention activity among typical protocol usage. Recent research improved the resistance of such tunnels against detection [32, 199, 217, 243, 206], pushing the boundaries of unobservable communications.

Traffic tunneling techniques based on web protocols have shown particular promise in

resisting blocking attempts. Tools such as Balboa [199] and WebTunnel [208] (the latter inspired by HTTPPT [93]) encapsulate covert data within widely used web protocols—HTTPS and/or WebSockets—blending seamlessly with legitimate interactions towards overt and popular websites that deploy circumvention proxy backends via plug-in software modules. These tools rely on the assumption that censors risk substantial collateral damage [84] if they block/interfere with web traffic, as doing so may disrupt access to essential services or widely used platforms, raising the potential political and economic cost of censorship [93, 199].

Despite leveraging high-value collateral in the hopes of evading blocking, existing web protocol tunneling tools remain vulnerable to detection via network *traffic analysis* techniques. Since WebTunnel/HTTPPT does not shape traffic patterns (e.g., modulation of flow-level characteristics such as inter-packet timing, burst patterns, or response sizes) [93], a censor can use traffic fingerprinting techniques to distinguish accesses to covert destinations from legitimate pages fetched from the overt website [209]. Balboa enhances protection against traffic fingerprinting by directly replacing any legitimate HTTP content (e.g., HTML, CSS, images) found within TLS records exchanged with the website. However, this process adds non-negligible timing differences, enabling censors to identify Balboa’s activity with up to $\sim 90\%$ accuracy over different web clients and network conditions [199].

Furthermore, existing web protocol tunneling tools fail to provide *behavioral realism* [243], i.e., to generate user interaction patterns that are consistent with realistic browsing behavior towards the overt website. Thus, while web protocol tunneling raises the cost of censorship by riding on top of critical web infrastructure, current solutions fall short of hiding both the network-level and behavioral signatures required for effective covert communication.

1.2 Our Solutions

In this thesis, we propose two separate tools, each designed to aid users in their respective scenarios. Through the following sections, we will detail these tools and list their contributions.

1.2.1 Anix: Anonymous Blackout-Resistant Microblogging with Message Endorsing

In Chapter 2, we propose Anix, a blackout-resistant communication app designed to overcome the limitations of previous work by introducing two new key features: a *remote trust establishment* mechanism that allows users to progressively build trust relationships with other users across the mesh network without the need for in-person interaction, and an *anonymous message endorsing* scheme, enabling users to share their opinions on messages with their trusted contacts while keeping their identity hidden from the rest of the network. Additionally, Anix includes *identity revocation* primitives that provide users with fine-grained control over their trust relationships and offer stronger anonymity guarantees, even if users are coerced or their devices are compromised, aligning with other state-of-the-art anonymous mesh networking solutions.

We implemented a prototype of Anix and evaluated its practicality through a set of micro-benchmarks and simulations that aim to reflect its operation in a (down-scaled) city-wide environment. Our results reveal that Anix can efficiently exchange messages and endorsements between devices, while message exchange times still dominate over Anix’s lightweight local operations on mobile devices. Our simulations reveal that, even under active attack by a set of adversary nodes that compose 2% of the overall network and refuse to forward messages authored by legitimate users, Anix messages can reach over 90% of users within 23 simulation steps (roughly equivalent to 23 hours).

Contributions. We summarize our contributions as follows:

- We survey prominent blackout-resistant mesh communication apps, highlighting the disparities between their threat models and design features.
- We design Anix, a new blackout-resistant messaging app that enables users to remotely establish and manage trust relationships across the mesh network.
- We build an anonymous message endorsing system that enables Anix users to prioritize messages based on the perceptions of their trusted contacts.
- We develop a prototype of Anix [1] and evaluate it via extensive micro-benchmarks and simulations.

1.2.2 Huma: Censorship Circumvention via Web Protocol Tunneling with Deferred Traffic Replacement

In Chapter 3, we present Huma¹, a new censorship circumvention tool that addresses the unobservability challenges of existing web protocol tunneling solutions by coalescing strong traffic fingerprinting resistance and behavioral realism. To resist traffic fingerprinting, Huma operates a traffic replacement scheme similar to that of Balboa [199], replacing websites’ leaf elements with covert data. However, Huma operates in a deferred fashion; instead of performing data replacement operations directly at each data exchange, websites immediately respond to Huma users’ requests with legitimate content, while covert data is prepared in the background. Covert data is then delivered in response to the same client’s next request, thus ensuring that no noticeable delays are introduced in the communication. To enforce behavioral realism, Huma relies on an overt user simulator (inspired by OUS-tral [158] and Raven [243]) which models Internet users’ realistic web browsing activity. By incorporating this simulator, Huma avoids detection by adhering to expected browsing patterns.

Beyond enhancing the covertness of web protocol tunneling, Huma addresses two dimensions that are often overlooked in prior circumvention research. First, Huma provides native *privacy safeguards against Sybil proxies*, ensuring that the covert destination contacted by a user (as well as that communication’s contents) remain hidden, even when a Huma-enabled website is controlled by an adversary. Huma achieves this by having users encrypt their messages such that only a trusted intermediary node—located beyond the website itself—can decrypt them, ensuring that the website’s operator cannot access or infer meaningful information. This helps mitigate attacks where censors may compromise Huma-enabled websites or set up honeypot websites to log user activities for strengthening evidence which may lead to later prosecution [266].

Second, Huma is designed to operate as a fallback covert communication tool within rising national *Intranet environments* [214, 273], such as those preliminarily tested in Russia [245, 155] and Iran [101, 215, 21]. Intranets isolate domestic users from the global Internet, often allowing access only to a limited set of government-approved services [102]. In this scenario, Huma facilitates covert communications between users within the censored region, despite the censor’s full visibility over the national routing infrastructure. This functionality is enabled by Huma’s traffic replacement scheme, which lets users submit encrypted messages to be stored by the Huma-enabled website, and a private informa-

¹Huma (or Homa) is a mythical Persian bird known to live its entire life flying invisibly high above the Earth.

tion retrieval (PIR) enabled key-value store [24], which allows users to retrieve messages addressed to them without revealing which messages they are accessing to the website.

We implement a prototype of Huma and evaluate it across multiple dimensions. First, we assess Huma’s resistance to content fingerprinting, examining whether covert data relayed by Sybil nodes can leak information to an adversary. Then, we examine whether covert communications can be distinguished from legitimate web activity. Lastly, we assess Huma’s network performance, quantifying its latency and bandwidth overheads, as well as expected content download rates when enforcing behavioral realism through different overt user simulators, which aim to replicate web browsing activity patterns.

Contributions. Our main contributions are as follows:

- We survey prominent circumvention tools, comparing them on the basis of traffic analysis resistance, unlinkability guarantees, and ease of deployment.
- We design Huma, a web protocol tunneling tool which resists traffic analysis and provides secure, even in Intranet environments.
- We develop a novel technique for receiving covert data from endpoints without any time anomalies called Double-Request Receive.
- We develop a prototype of Huma and deliver an evaluation of its content fingerprinting resistance, covert communications, and network performance with and without using an over user simulator.

Chapter 2

Anix: Anonymous Blackout-Resistant Microblogging with Message Endorsing

In this chapter, we will motivate and present Anix, an anonymous blackout-resistant microblogging system that allows for message endorsing, secure referrals, and remote trust establishment (all features previously unavailable in the mesh messaging space), while providing users with various privacy guarantees. In §2.1, we analyze previous mesh messaging solutions and compare them across several dimensions while defining Anix’s goals. In §2.2, we outline our threat model assumptions, which help navigate the ideas Anix aims to present. In §2.3, we detail Anix’s inner workings, introducing features and describing how they fit together to provide Anix’s functionalities. Later in §2.4 and §2.5, we describe our evaluation methodology and then present our results, revealing Anix’s performance and impact. Finally, in §2.7, we examine related work to Anix.

2.1 Blackout-Resistant Messaging Apps

This section surveys the landscape of existing blackout-resistant messaging apps and categorizes each solution according to four different dimensions tied to their design and operational characteristics. While there is a history of prior blackout-resistant messaging apps [152, 192, 188, 37, 157], most of them differ on a) the communication model they support; b) the anonymity guarantees they provide; c) the way they allow users to trust

Table 2.1: Comparison of blackout-resistant communication apps. A green check mark means the feature is fully present, an orange one means it is somewhat present or can be easily added, and a red cross means it is not present.

Application	Communication			Anonymity			Trust System			Revocable IDs	
	O2O	S2S	O2M	SRA	FA	PCA	DT	DTM	TT	SR	HR
Firechat [204]	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
Bridgefy [10]	✓	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗
Briar [9]	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗
1am [157]	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗
Moby [192]	✓	✗	✗	✓	✓	✗	✓	✗	✗	✗	✗
Perry et. al. [188]	✓	✓	✗	✓	✗	✗	✓	✓	✗	✗	✗
ASMesh [37]	✓	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗
Rangzen [152]	✓	✗	✓	✓	✓	✗	✓	✗	✓	✗	✗
Anix	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

each other or the messages flowing through the mesh, and; d) the ability for refreshing user identities to recover from different attacks. Throughout our analysis, we contrast the features provided by existing blackout-resistant messaging apps, and make the case for specific design goals we aim to achieve with Anix. We also showcase how Anix fits amongst the existing solutions, supported by a detailed comparison shown in Table 3.1.

2.1.1 Communication Models

Messaging apps can generally support three communication models: a) *one-to-one* (O2O), or private messaging, in which two parties exchange messages directly between each other across the mesh network; b) *some-to-some* (S2S), or group messaging, in which two or more users exchange messages in a way that only they can read them; c) *one-to-many* (O2M), or broadcasting, in which a user sends messages to every other user on the mesh at once (akin to publicly posting on microblogging platforms such as X [13] or Reddit [11]). As shown in Table 3.1, most prior solutions are geared for a one-to-one or some-to-some scheme, in which the mesh network is used as an infrastructure to exchange private messages with other users.

Microblogging capabilities. Rangzen [152] is the single academic proposal for a blackout-resistant app that focuses on offering a one-to-many messaging scheme. Other solutions such as Bridgefy [10] and Firechat [204] support all communication models, but are engineered to provide more comprehensive support for specific models. For instance, Firechat

focuses on one-to-many messaging, while Bridgefy focuses on one-to-one and some-to-some communications.

Communication scale. While solutions such as Perry et al. [188] explore domain-specific optimizations for the efficient dissemination of messages between pockets of users who are densely packed within small areas, most blackout-resistant messaging apps aim to deliver messages to large community segments, typically comprising several thousand mobile users throughout a city, with reasonable reliability.

Design goal 1. One-to-many messaging at city-wide scale, while also supporting other communication models.

2.1.2 Anonymity Guarantees

In the context of mesh-based messaging apps, anonymity can be mapped to three main attributes [37]: a) *sender and receiver anonymity* (SRA), which requires the app to prevent an adversary from linking a given message to its sender and intended receiver(s); b) *forward anonymity* (FA), which requires the app to uphold the sender and receiver anonymity guarantees of older messages if the state of any of the parties engaged in communication is compromised in the future; and; c) *post-compromise anonymity* (PCA), which requires the app to uphold sender and receiver anonymity for future messages exchanged after a party involved in the protocol is coerced or compromised, although the re-establishment of these guarantees may require a brief recovery period.

FA and PCA are aligned with the notions of forward secrecy and post-compromise security, which normally relate to ensuring message confidentiality [37]. As highlighted in our threat model (§2.2), however, the typical adversary in an internet shutdown scenario seeks to track the activities of users engaged in mesh messaging without necessarily breaching confidentiality. This means that, for the O2O and S2S communication models where messages are usually encrypted, the adversary will seek to uncover who is communicating with whom. In the O2M model, where messages are sent in the clear, the adversary may instead attempt to uncover the author of a specific message. As a result, we argue that a blackout-resistant app should, at the very least, provide basic sender and receiver anonymity. Some solutions, however, fail to satisfy this requirement.

Anonymity in non-academic solutions. These messaging apps either do not intentionally hide information about a message sender or receiver, or fail to do it successfully. In Firechat [204], users have long-term public IDs that uniquely identify their accounts and are sent along with their messages to every user they share their messages with.

Bridgefy [10] included both the sender’s and receiver’s long-term identifier on each message. Albrecht et al. [16] reported that this practice was unsafe, compelling Bridgefy’s maintainers to enhance the app’s security. Yet, the deployed upgrades proved to be insufficient [17] and, in its current state, Bridgefy cannot be considered anonymous.

Briar [9] only allows two users in close physical proximity (e.g., Bluetooth range) to establish a connection if they have previously exchanged contact information, but the app does not attempt to conceal the identities of communication parties [8]. Restrictions on connection establishment may also help a passive adversary uncover a user’s social graph.

Anonymity in academic solutions. Mesh messaging apps described in the literature support different anonymity properties to varying degrees. 1am [157] does not provide any anonymity as its messages are exchanged in plaintext and no metadata protections are in place. On the other hand, Rangzen [152] achieves sender and receiver anonymity by explicitly avoiding including the sender and receiver IDs in transmitted messages. Despite its heavy focus on the microblogging scenario, Rangzen users can still find messages specifically targeted at them by actively trying to decrypt encrypted messages flowing through the mesh. (In §2.1.3, we address how users can exchange cryptographic material to engage in private communication over one-to-many networks.) Rangzen also naturally provides forward anonymity since messages are not linked to their corresponding authors.

Moby [192] achieves the same anonymity guarantees as Rangzen by relying on a simpler version of Signal’s [12] double ratchet which enables for sender and receiver anonymity with forward anonymity. ASMesh [37] improves upon Moby by supporting sender and receiver anonymity with both forward and post-compromise anonymity.

Perry et al. [188] seek to ensure sender and receiver anonymity against an adversary that has full visibility over all communications established across the mesh network, by using cover traffic to hide users’ messaging patterns. In these conditions, however, highly repressive adversaries may shift focus to outright arrest the individuals found to use a mesh networking app – while this issue is not specific to Perry et al. (and solving it remains an open challenge), it may prompt an adversary to take swifter action in this direction. Further, Perry et al. do not offer safeguards to users who are coerced or whose devices are compromised by the adversary, lacking forward and post-compromise anonymity mechanisms.

Design goal 2. Ensure anonymity despite partial network monitoring, user coercion, or device compromise.

2.1.3 Trust Systems

Blackout-resistant messaging apps have used different heuristic notions for defining the boundaries of how much a given user can trust messages or other users or messages.

No trust. Some apps avoid defining a trust system altogether (e.g., Firechat [204]). These apps place every single message received by the user at the same level of importance regardless of their originating user, or legitimacy, and as a result, cannot distinguish spam or misinformation from otherwise.

Direct trust. Another group of mesh messaging apps defined trust as simply knowing (with a reasonable certainty) from whom the incoming message is coming from (i.e., a trusted contact). We call these *direct trust* (DT) systems.

Apps using direct trust systems require some notion of identity to be exchanged when any two users wish to establish trust between one another. However, most apps using direct trust systems do not concern themselves with the method through which identities are shared, either mentioning that trust establishment takes place when Internet is available (as in Moby [192]), or that users can exchange keys manually or by scanning a QR code (as in Rangzen [152], ASMesh [37], or Perry et al. [188]). Bridgefy [10] uses a direct trust system in its private and group messaging but has no trust system in place for one-to-many communications.

Direct trust mediators. A third group of apps builds upon the notion of direct trust and improves it by introducing novel ways for helping users establish trust across the mesh network, even when users may not necessarily know each other. We call these *direct trust mediator* (DTM) systems.

Briar and Perry et al. support direct trust mediation by leveraging *introductions*, a scheme in which a common contact between two users can introduce both of them by privately relaying their identifiers to each other across the mesh network. While Perry et al.’s solution and Briar’s ultimately translate to direct trust, they present a useful mechanism to remotely establish trust across a mesh.

Note that existing direct trust mechanisms (mediated or not) currently lack the means to remove the trust placed on certain users. This may be a liability when one wrongfully trusts (potentially malicious) users and wishes to “un-trust” them. We address potential solutions for this issue in §2.3.3.

Transitive trust. Trust systems gain a heightened relevance when applied to the one-to-many communication model since, in a microblogging scenario, distinguishing legitimate messages from spam becomes a significant challenge [152]. In this scenario, it would be

useful for users to be able to tell whether (and whom amongst) their trusted contacts has vetted some message as being relevant, even if that message has not been authored by any of the user’s trusted contacts. This requires users not only to be able to establish direct trust relationships but also to have a way to perceive how their trusted contacts assess a given message transmitted across the network. We call these *transitive trust* (TT) systems, as they allow users to propagate their positive/negative assessment of a given message to their other contacts.

Rangzen attempts to provide a notion of trust that aligns with – albeit not exactly meets – our definition of transitive trust. To determine the “trustability” of a received message, it privately computes the cardinality of the intersection [66] between the contact list of two users when their devices exchange messages with one another. Briefly, the more contacts two users have in common, the more they trust the messages they receive from one another. This is reflected in a message’s *trust score*, which is a direct result of the multiplication of the trust value between two users exchanging the messages and the messages’ original trust scores.

We identify several particularities of Rangzen’s trust approach that prevent it from meeting our definition of transitive trust. First, the app is afflicted by issues related to *diminishing trust*, i.e., the longer the message has to travel through the network, the lower its trust score becomes, regardless of originating from a trustful source or not. Rangzen offers a way to “repost” a message by resetting its trust score back to its maximum value, but this is unreliable as a user still relies on the message’s current trust score to decide whether to repost it. Additionally, reposting feeds into the issue of *path dependency*, i.e., a message’s trust score can vary greatly based on the path it took to reach a given user. Thus, in Rangzen, a user has no way to gauge the overall sentiment that her trusted contacts (potentially spread across the mesh at large) placed on a given message.

Design goal 3. A network-wide and path-independent transitive trust system that improves the ability of users to place trust in messages transmitted across the network.

2.1.4 Revocable Identities

Existing apps tie users’ identities to long-lived and static identifiers (e.g., long-term public keys). Unfortunately, this may be detrimental for the anonymity of users and for the authenticity of messages sent across the mesh. For instance, if the adversary can gain the trust of a legitimate user, it may be able to permanently track that user’s activities over the mesh network. If a user’s device is compromised, the adversary may also acquire cryp-

tographic material that allows it to deanonymize users or impersonate the compromised user.

As mentioned before, direct trust mediation systems may further exacerbate anonymity breaches, as mediator nodes can establish irrevocable trust relationships between two users without their explicit consent. Indeed, Perry et al. [188] mention that should a malicious user be trusted, their solution has no safeguards to prevent de-anonymization.

We argue that, to preserve users' anonymity and ensure the authenticity of messages after a compromise, user identities can be designed to be ephemeral and do not need to be tied to a permanent long-term identity. Based on this assumption, we envision two types of identity revocation primitives, each suited for different scenarios. Next, we briefly describe these primitives and refer the reader to §2.3.3, where we describe how Anix realizes them in practice.

Soft revocation. A first kind of identity revocation, *soft revocation* (SR), allows a user to remove wrongfully trusted contacts from her trust circle, thus addressing the drawbacks of existing direct trust systems. Soft revocation helps a user revoke her current temporary identity, migrate to a freshly rolled temporary identity, and privately share this new identity with the users she wishes to keep within her trusted circle while excluding distrustful ones.

Hard revocation. An adversary who takes control of a user's device could henceforth impersonate that user. We envision *hard revocation* (HR) as a potential identity revocation primitive that allows a user to signal to her trusted contacts that her account has been compromised and that no further messages authored by this user should be trusted.

Design goal 4. Support identity revocation primitives that enhance anonymity and message authenticity safeguards.

With the above goals in mind, we now present the threat model Anix will operate in before addressing Anix's design.

2.2 Threat Model

Our threat model assumes a state-level adversary that will force users to switch to Anix for communicating during a blackout. In this setting, the goals of the adversary are threefold: a) to identify the real identity of the users responsible for creating certain public messages exchanged through Anix (e.g., the organizer of a protest); b) to manipulate the way Anix's users perceive the information shared throughout the mesh network (e.g., towards misleading a potential protest's participants with false rally point instructions), and; c) to disrupt

communication through the mesh network (e.g., by introducing Sybil nodes that refuse to forward messages, or temporarily jamming wireless signals in certain areas).

Below, we outline our assumptions about the environment where Anix is expected to operate, and the main capabilities and limitations of the adversary we consider.

Assumptions on the adversarial environment. We assume that the adversary is capable of arbitrarily disabling the Internet infrastructure within the country’s borders. Further, during an Internet blackout, there is no other safe external network through which users can communicate. This forces users to resort to a mesh networking solution that relies on wireless protocols, such as Bluetooth and Wi-Fi Direct, to connect with nodes within close physical proximity. Through epidemic protocols [234], a user’s message will eventually be propagated throughout multiple hops within the mesh network and reach its intended receiver(s).

Like Moby [192], we assume the adversary is only able to operate locally. Thus, while the adversary may intercept or drop messages in certain locations by deploying Sybil nodes in the network [152], the adversary will not be able to globally drop or intercept Anix messages.

Lastly, we assume Anix to be deployed within regions considered to be of medium risk [152, 267] where, though there might be restrictions on free speech, governments do not rely on violence and mass arrests to repress citizens who rely on alternative means to the Internet to communicate. Similarly to Rangzen [152], we consider this to be a reasonable assumption given that solutions such as Anix may either be unnecessary in low-risk scenarios (e.g., within democratic countries whose governments avoid cracking down on protests) or a severe liability in highly repressive countries (e.g., dictatorial regimes), where the simple usage of mesh networking can enact an extreme response from authorities.

Adversary’s capabilities. To accomplish its goals, the adversary can set up multiple Sybil nodes within Anix’s mesh network and pose as a legitimate user. This allows the adversary to passively intercept messages flowing through the network to, for instance, learn the content of public messages or screen message headers for finding specific identifiers. By playing the role of a Anix node, the adversary can also simply refuse to forward messages. Participating in the network also allows the adversary to fabricate messages and attempt to manipulate the contents of incoming messages before relaying them to other peers in the network. We assume the adversary will also strive to build trust with other Anix users over time and try to influence the actions of benign users by purposely spreading messages carrying misinformation. In addition, we assume that there is the possibility that the adversary will coerce users or seize users’ devices to compromise the state held by the Anix app (e.g., message logs or cryptographic material).

Adversary’s limitations. The adversary has a limited capacity to convince users to trust its Sybil nodes, resulting in a restricted number of social connections between the adversary and the rest of the network. Thus, the adversary is expected to be less well-connected to other legitimate Anix users within the mesh network. (In §2.5, we expand on the advantage acquired by an adversary as the ratio of social connections to legitimate users increases.) Furthermore, the adversary is assumed to be computationally bounded and unable to break standard cryptographic primitives.

Attacks outside the scope. Anix does not prevent the adversary from learning that a given individual is using the application. Thus, Anix will not protect users against adversaries that may retaliate simply based on the fact that the app is in use. Anix also does not protect users from targeted attacks where the adversary can single out a legitimate user within a close physical vicinity. In such cases, the targeted user may be limited to directly exchanging newly produced messages with Anix nodes controlled by the adversary, thus forfeiting sender anonymity. Additionally, Anix does not explicitly aim to defend against denial of service attacks where the adversary interrupts communication between nodes in the mesh network (e.g., via jamming) or where nodes selectively refuse to forward certain messages. However, similarly to previous approaches [192], Anix leverages the propagation capabilities of epidemic protocols to deliver messages across the mesh despite local and/or temporary disruptions.

2.3 Anix

This section discusses the inner workings of Anix, a new blackout-resistant messaging app. We start by outlining Anix’s workflow on the *one-to-many* communication model before addressing its core design features and message exchange mechanisms. Then, we detail how Anix can easily accommodate other communication models.

2.3.1 Overview of the Microblogging Workflow

Anix’s microblogging workflow relies on three core operational constructs, which we describe below. Their synergies allow for the creation of an anonymous messaging scheme that allows users to reason about whether they can trust messages originating from potentially unknown users.

1. Pseudonyms and identity management. Central to Anix’s operation is the use of one-time-use pseudonyms (*PSUs*), which partly act as public keys. Every time a user

(e.g., Alice) wishes to send a message, she generates a new pseudonym and places it in the “sender” field of her message. While this ensures sender anonymity for every new Alice message, Anix allows another user to track multiple messages sent by Alice, as long as Alice provides this user the necessary cryptographic material required to link multiple of her pseudonyms together. We expand on the management of pseudonyms in §2.3.2. Further, given that Alice may wish to prevent certain previously trusted parties from recognizing her messages (e.g., if her device is compromised and used to spread misinformation, or she outright loses trust in certain parties), Anix provides identity revocation primitives that can be used to refresh cryptographic material. We will also cover this aspect in more detail in §2.3.3.

2. Trust management. We say that *Alice trusts Bob* when Alice sends Bob the cryptographic material that allows him to track Alice’s *PSUs* over various activities. This cryptographic material can be securely propagated to Bob through the mesh network by encrypting it with one of the public keys exposed in any of Bob’s *PSUs*. Bear in mind that during this process, Alice does not necessarily know the identity of Bob. Upon receiving this cryptographic material, Bob can opt to reciprocally trust Alice immediately (e.g., if Alice and Bob are real-life friends who are meeting in person and actively seeking to establish a bi-directional trust relationship in Anix), to trust Alice after tracking her messages over time (e.g., in the case where Bob can only judge whether to trust a user by the content of her messages), or end up deciding not to trust her at all. We elaborate on Anix’s trust management primitives in §2.3.3. Different from existing approaches, this process enables Anix users to progressively build trust in a given user based on the contents of its messages. It also allows for remotely establishing trust during a blackout, without the need to rely on in-person encounters or temporary internet access.

3. Message endorsing and ranking. Anix allows for message endorsing through “votes”, as in microblogging platforms such as Reddit [11]. In a nutshell, Anix leverages the concept of *up/downvotes* in a message to attribute an importance score (i.e., rank) to this message. Votes for a given message are propagated across the mesh network when a message is directly exchanged between two nodes, but Anix allows for multiple deliveries of the same message so they can accrue votes over time. Since votes are authored by users’ identity keys (§2.3.2), Alice may not only track the total number of up/downvotes on a message but also track the number of users she trusts who (dis)approve the message. While total up/downvotes could better reflect the overall sentiment about a message being propagated in the network, such votes may be inflated by Sybil nodes. Thus, for messages dealing with sensitive topics, Alice may prefer to reason about how many of her trusted contacts (and/or which of these contacts) upvoted or downvoted the message. We address Anix’s message voting mechanisms in §2.3.4, and point-to-point message exchange

Algorithm 1 Generate pseudonym (PSU)

Input: $PubK_{SID}, PrivK_{SID}$, random r, r'
1: $PubKE_{OTU}, PrivKE_{OTU} \leftarrow \text{generateRandomKeyPair}(r)$
2: $PubKS_{OTU}, PrivKS_{OTU} \leftarrow \text{generateRandomKeyPair}(r')$
3: $bsig \leftarrow \text{BSign}_{PubK_{SID}, PrivK_{SID}}(PubKE_{OTU} || PubKS_{OTU})$
4: $PSU \leftarrow PubKE_{OTU} || PubKS_{OTU} || bsig$
5: **return** $PSU, PubKE_{OTU}, PrivKE_{OTU}, PubKS_{OTU}, PrivKS_{OTU}$

in §2.3.5.

Next, we expand on each of the above three constructs and how they enable us to meet the goals outlined in §2.1.

2.3.2 Pseudonyms and Identity Management

Anix nodes rely on a set of cryptographic material to establish their identity in the network and communicate with one another via (potentially anonymous) pseudonyms. This cryptographic material also allows users to identify whether and how their trusted contacts voted on a given message. Specifically, Anix relies on a two-tiered set of *hierarchical keys* to manage user identities within the network.

Cryptographic material and operations. Each user has access to two types of randomly generated asymmetric keys: temporary *identity keys* ($PubK_{ID}, PrivK_{ID}$) and *one-time-use keys* ($PubK_{OTU}, PrivK_{OTU}$). These keys are instrumental in the creation and management of a user's identity via pseudonyms and, at any given instant, a user may hold multiple identity keys and one-time-use keys. Each of the mentioned key pairs includes separate keys for encryption and signing; for simplifying our exposition, we refer to both the signing and encryption keys using the same notation, making explicit distinctions where appropriate. We use elliptic curve cryptography schemes to generate the above keys (see §2.4), as Anix operates in a bandwidth-constrained setting where it is desirable to transmit public keys with shorter lengths (e.g., vs. longer RSA keys for equivalent levels of security). Lastly, we refer to the cryptographic operations required by Anix—e.g., encryption (**Enc**), decryption (**Dec**), or signing (**Sign**)—as $\text{Operation}_{Key}(Message)$, and use $||$ to denote concatenation.

Pseudonym generation. Every time a user, e.g., Alice, wishes to send a message to the network, she must generate a new one-time-use pseudonym (PSU) which anonymously identifies the sender. Algorithm 1 describes the creation of a pseudonym. First, Alice randomly generates two new one-time-use (OTU) keypairs (lines 1 and 2) used for the encryption ($PubKE_{OTU}, PrivKE_{OTU}$) and signature ($PubKS_{OTU}, PrivKS_{OTU}$) scheme,

Algorithm 2 Signature with key blinding

Input: Message m , verification/signing keypair $PubKS_{ID}, PrivKS_{ID}$, algorithm [70] for generating a signature with public key blinding ($B\text{Sign}_{PrivKS_{ID}, bk}(m)$), where bk is the blinding factor.

Output: Message m gets signed with public key blinding

```
1: func  $B\text{Sign}_{PubKS_{ID}, PrivKS_{ID}}(m)$ :  
2:    $bk \leftarrow \text{Hash}(m || PubKS_{ID})$   
3:   return  $B\text{Sign}_{PrivKS_{ID}, bk}(m)$ 
```

respectively. Then, Alice generates a signature ($bsig$) of the concatenation of the public components of her one-time-use keypairs ($PubKE_{OTU} || PubKS_{OTU}$) using the (blinded) private component of one of her temporary identities' signing keypair ($PrivKS_{ID}$) (line 3). We describe the process of signing with key blinding ($B\text{Sign}$) later in this section, as it is fundamental to ensure the unlinkability of pseudonyms. Finally, Alice generates a *pseudonym* (PSU) composed of the concatenation of $PubKE_{OTU}$, $PubKS_{OTU}$, and $bsig$ (line 4). Since each of the three components of a PSU has a fixed size, they can easily be split and interpreted separately.

Ensuring pseudonym unlinkability. Anix's pseudonyms (and the keys contained therein) must adhere to three important properties: a) signatures of one-time-use keys (included in different $PSUs$) under the same $PrivKS_{ID}$ cannot be linked together and traced back to a given user without the knowledge of the corresponding $PubKS_{ID}$; b) messages encrypted with a given $PubKE_{OTU}$ can only be decrypted by the user possessing the corresponding $PrivKE_{OTU}$, and; c) $PSUs$ are unforgeable.

As mentioned above, ensuring pseudonym unlinkability – i.e., that each message generated and signed by different $PSUs$ of the same user cannot be associated together – is an important design consideration for Anix. Unfortunately, common signature schemes (e.g., based on ECDSA [121]) which would directly use $PrivKS_{ID}$ as a signing key – on both $PSUs$ and votes (see §2.3.4) – are prone to public key recovery attacks [201] which enable an adversary to recover the public identity key associated with a signature, and link all of a user's activities performed under different $PSUs$.

To address this issue, we explored Denis et al.'s [70] signature scheme with key blinding (which has already been the focus of a formal security analysis by Eaton et al. [77]). Briefly put, this scheme allow us to *blind* (i.e., randomize) the key pair associated with a signature, such that this signature is not linkable to the corresponding public verification key without knowledge of a particular witness. It also ensures unlinkability, in that a verifier cannot distinguish between two signatures based on the same private signing key from two signatures based on distinct signing keys.

Leveraging signatures with public key blinding. Algorithm 2 defines the function

Algorithm 3 Verify a signature generated with key blinding

Input: Signature with key blinding s , message m , verification key $PubKS_{ID}$, algorithm [70] for verification of signatures with public key blinding ($Verify_{bpk}(m, s)$), where bpk is the blinded public verification key.

Output: The signature s of message m is verified

```
1: func BVerPubKSID( $m, s$ ):  
2:    $bk \leftarrow \text{Hash}(m || PubKS_{ID})$   
3:    $bpk \leftarrow \text{BlPubKey}_{bk}(PubKS_{ID})$   
4:   return  $Verify_{bpk}(m, s)$ 
```

BSign, used in Anix’s pseudonym generation and vote signing procedures. Internally, this function can leverage any public key blinding scheme and, in our current implementation, **BSign** leverages Denis et al. [70] signature scheme with public key blinding (abstracted as **BlSign**). This function takes in a user’s identity private signing key ($PrivKS_{ID}$) alongside a blinding factor (bk) and uses them to create a signature that cannot be linked back to the corresponding public verification key $PubKS_{ID}$ without the knowledge of bk . Under [70], a signer generates a different blinding factor bk (line 2) for each new public key blinded signature (line 3).

In turn, Algorithm 3 defines **BVer**, Anix’s procedure for verifying a signature generated via **BSign**. Internally, this function leverages Denis et al. [70] signature verification procedure (abstracted as **Verify**), in which a user can verify a public key blinded signature if they have access to the corresponding blinded public verification key (bpk) – i.e., the public component of the key used to sign the message (Algorithm 2) blinded using a blinding factor (bk) under the public key blinding function (**BlPubKey**). While it is typically assumed that either bk or bpk must be communicated to verifiers out-of-band [77], this is unfeasible in our scenario. However, Anix users (both signers and verifiers) can locally generate bk (and, subsequently, bpk) for a message, e.g., by hashing the message to be signed/verified (M) concatenated with its authors’s identity signing key public component ($PubKS_{ID}$), i.e., $bk = H(M || PubKS_{ID})$. This ensures Alice’s signatures (included in her *PSUs* and votes) become linkable only to other Anix users who have access to Alice’s $PubKS_{ID}$ (i.e., users who are trusted by Alice). In the case of *PSU* generation, $bk = H(PubKE_{OTU} || PubKS_{OTU} || PubKS_{ID})$.

Tracking message authorship and compartmentalization. The choice of identity keys provides a user with the ability to control the groups of nodes that can track the messages she sends to the network, without necessarily having her disclose her real identity. For instance, consider that Alice creates two different identities, ID_1 and ID_2 , with public components ($PubKE_{ID_1}$, $PubKS_{ID_1}$) and ($PubKE_{ID_2}$, $PubKS_{ID_2}$). Alice could then choose to exchange $PubKS_{ID_1}$ in person with some trusted close contacts, e.g., to enable friends and family to track the set of messages whose authoring *PSUs* are based on ID_1 .

Algorithm 4 Establish a one-way trust relationship ($A \rightarrow B$)

Input: Public component of an identity key-pair of user A ($PubK_{ID_A}$) which contains both the public encryption and verification keys,
Encryption keys of user B's *PSU* ($PubKE_{OTU_B}, PrivKE_{OTU_B}$),
Output: User B securely receives user A's identity key

```
1: func OWT(receiverOTU, senderID):  
2:   OWTMessage  $\leftarrow$  EncreceiverOTU(senderID)  
3:   sendMessage(B, OWTMessage)  
4:   OWT( $PubKE_{OTU_B}$ ,  $PubK_{ID_A}$ )
```

```
5:  $PubK_{ID_A} \leftarrow$  DecPrivKEOTU_B(OWTMessage)
```

Further, upon sharing $PubKS_{ID_2}$ to place trust in other (anonymous) users in the network, Alice enables these users to track a different set of messages whose authoring *PSUs* are tied to ID_2 . Thus, Alice can compartmentalize the groups of users she wishes to be able to link different messages, while still being able to build anonymous trust relationships within the network. In case Alice wishes to prevent a previously trusted contact from tracking her new messages, Anix provides mechanisms for revoking identities. We cover these in the next section, after discussing how Anix manages trust between anonymous users.

2.3.3 Trust Management

Like in previous work, we expect users to bootstrap their list of trusted contacts by exchanging identity keys in person. However, Anix introduces a new direct trust primitive – denoted as *one-way trust* – that enables users to remotely establish trust between each other across the mesh network. In addition, Anix introduces a novel direct trust mediation mechanism – in the form of *referrals* – with significant security enhancements when compared to the one available in previous apps. Lastly, Anix provides a set of identity revocation primitives that better protects users' anonymity.

The notion of one-way trust. Anix offers a way for Alice to place her trust in another anonymous user within the mesh, even if they have never met before. In contrast to popular social media platforms which typically enable Alice to decide to “follow” some other user whose posts seem interesting to her, Anix provides the concept of “allowed to follow”. We call this construct a *one-way trust* (*OWT*), which can eventually result in a bi-directional trust relationship once two users are “allowed to follow” each other.

One-way trust establishment. *OWT* works as follows. Assuming Alice finds a message which is useful to her or that is endorsed by some of her trusted contacts, she can grant that message's author the right to track her messages and hope to be “allowed to follow”

back. Alice grants this right by sending one of her own identity keys to the owner of the pseudonym that authored the message of interest. Apart from simply judging the content of a message before depositing her trust in the Anix user that authored it, Alice can also reason about whether (and whom amongst) her trusted contacts have upvoted the contents of this message.

To illustrate a unidirectional trust establishment leveraging *OWT*, consider the following example aided by the description of *OWT*'s steps in Algorithm 4. Assume that Alice decides to trust a message's author via its pseudonym (that we somehow know is from Bob), after verifying that the content of the message is useful and that many of her trusted contacts have upvoted it. Alice proceeds to encrypt one of her identity (e.g., ID_A) keys' public encryption and signing components (jointly represented as $PubK_{ID_A}$) with the $PubKE_{OTU_B}$ key found in Bob's pseudonym (line 2) and sends this ciphertext over the mesh network (line 3), such that only Bob can open this message and access Alice's $PubK_{ID_A}$ (line 5). (In other words, *OWT* messages are encrypted using a receiver's one-time-use encryption key included in one of their *PSUs*, thus being indistinguishable from other O2O/S2S messages; see §2.3.5 – Alternative communication models.) As a result, Bob has now been “allowed to follow” the messages that Alice sends under any pseudonym generated based on her identity ID_A .

Bi-directional trust establishment. Eventually, Bob can decide to trust Alice back, e.g., after receiving useful messages authored by Alice's pseudonyms generated under (ID_A), or verify that his trusted contacts also tend to upvote messages authored by the same pseudonyms. To trust Alice, Bob simply follows an *OWT* establishment towards Alice. Once she receives this message, Alice can verify that Bob has allowed her to follow him back, thus establishing bi-directional trust.

Until Bob decides to allow Alice to follow him back, Alice's $PubK_{ID_A}$ identity key will sit on a *validation list*, i.e., a separate list for user identities who have one-way trusted Bob, but whom Bob has not yet decided to trust back. As we shall see in §2.3.4, identity keys in the validation list will not be used for the calculation of message trust scores.

Should Bob choose not to follow Alice back, the presented scheme would allow Bob to track Alice's activities throughout the mesh network indefinitely. To limit her exposure, Alice can set a timer for a period she considers acceptable for having Bob establish bi-directional trust (see §2.3.5). If this period elapses and Alice has not received an *OWT* message from Bob, she can remove the trust she deposited in Bob by revoking her temporary identity. We elaborate on trust revocations later in this section.

Trust via referrals. Anix supports direct trust mediation (see §2.1.3) via *referrals*, allowing a user to refer two of their trusted contacts to one another. Anix's referrals

Algorithm 5 Refer two users (A and B) to one another

Input: A PSU of user A (PSU_A) and the corresponding one-time-use public encryption key of user A ($PubKE_{OTU_A}$)
A PSU of user B (PSU_B) and the corresponding one-time-use public encryption key of user B ($PubKE_{OTU_B}$)

Output: Users A and B securely establish bi-directional trust

- 1: $referralMessageToA \leftarrow Enc_{PubKE_{OTU_A}}(PSU_B)$
- 2: $referralMessageToB \leftarrow Enc_{PubKE_{OTU_B}}(PSU_A)$
- 3: $sendMessage(A, referralMessageToA)$
- 4: $sendMessage(B, referralMessageToB)$

- 5: $PSU_B \leftarrow Dec_{PrivKE_{OTU_A}}(referralMessageToA)$
- 6: $PubKE_{OTU_B} || PubK_{SOTU_B} || bsig_B \leftarrow PSU_B$
- 7: $OWT(PubKE_{OTU_B}, PubK_{ID_A})$

- 8: $PSU_A \leftarrow Dec_{PrivKE_{OTU_B}}(referralMessageToB)$
- 9: $PubKE_{OTU_A} || PubK_{SOTU_A} || bsig_A \leftarrow PSU_A$
- 10: $OWT(PubKE_{OTU_A}, PubK_{ID_B})$

are reminiscent of Briar and Perry et al.’s introduction scheme, albeit with a significant distinction. In these previous works, introductions would enable for the establishment of direct trust between two users that had not met before, without their explicit consent; the mediator node would simply reveal each user’s long-term identities to one another. This solution could prove disruptive in cases where the mediator misjudges the trustfulness of one party (e.g., introducing a legitimate user to an adversary’s agent), or in cases where one of the legitimate users would simply like to avoid other unvetted users accessing their identity.

Anix leverages $PSUs$ to improve upon existing introduction schemes, requiring Alice and Bob’s active consent to trust one another, should Charlie refer them. As outlined in Algorithm 5, a referral unfolds by first having Charlie send a PSU of Alice to Bob (and vice versa) to *refer* them to each other (lines 1–4). Then, Alice and Bob can each independently choose to leverage Anix’s one-way trust mechanism to establish bi-directional trust (lines 5–10). As previously discussed in this section, if only one of the parties (e.g., Alice) feels comfortable issuing an OWT message, she may later revoke the one-way trust placed in Bob, aborting the unfinished bi-directional trust establishment.

Revocable identities. Anix allows its users to *revoke* their identities to help them preserve their anonymity and signal to other users in the mesh that their cryptographic material may have been compromised. To this effect, Anix supports the two identity revocation primitives described below.

Soft revocation. This form of identity revocation allows Alice to selectively remove the trust she placed on a particular user, e.g., Bob (or a set of users). Soft revocation relies on Alice’s ability to update the compartments of users that can track her messages in the

Algorithm 6 Soft identity revocation

Input: Old identity's encryption $(PubKE_{ID_{old}}, PrivKE_{ID_{old}})$ and signing $(PubK_{SID_{old}}, PrivK_{SID_{old}})$ keys, trusted users list (u) , random r, r'

Output: New list of trusted users u'

```
1:  $PubKE_{ID_{new}}, PrivKE_{ID_{new}} \leftarrow \text{generateRandomKeyPair}(r)$ 
2:  $PubK_{SID_{new}}, PrivK_{SID_{new}} \leftarrow \text{generateRandomKeyPair}(r')$ 
3:  $u' \leftarrow \text{new contact list}$ 
4: for  $u_i$  in  $u$ :
5:   if  $(u_i \text{ is still trusted})$ :
6:      $sig \leftarrow \text{Sign}_{PrivK_{SID_{old}}}(PubKE_{ID_{new}} || PubK_{SID_{new}})$ 
7:      $srm \leftarrow \text{Enc}_{PubKE_{ID_{old}}}(PubKE_{ID_{new}} || PubK_{SID_{new}} || sig)$ 
8:      $\text{sendMessage}(u_i, srm)$ 
9:      $u'.\text{append}(u_i)$ 
10:  $\text{Del}(PubKE_{ID_{old}}, PrivKE_{ID_{old}}, PrivK_{SID_{old}}, PrivK_{SID_{old}})$ 


---


11:  $PubKE_{ID_{new}}, PubK_{SID_{new}} \leftarrow \text{DecryptAndVerify}(srm)$ 
12:  $h \leftarrow \text{Hash}(PubK_{SID_{old}})$ 
13:  $\text{addEntryToRevokedKeys}(h, PubK_{SID_{old}}, PubK_{SID_{new}})$ 
14:  $\text{Del}(PubKE_{ID_{old}}, PubK_{SID_{old}})$ 
```

mesh network (see §2.3.2). Hence, to revoke the trust deposited in Bob, Alice must dispose of the identity key she previously forwarded to Bob via *OWT*, and rebuild a compartment that includes all users with access to this key, except for Bob. In practice, Alice generates a new identity key pair, encrypts both the identity key she wishes to revoke and her new public identity key with the public identity key of the contacts she is currently “allowed to follow”, and sends this message to them privately on the mesh network. As a result, the soft revocation process allows trustworthy contacts who were tracking Alice’s messages to continue doing so once she stops using the identity key also shared with Bob. The steps for Anix’s identity revocation mechanism are detailed in Algorithm 6.

When Alice’s contacts receive the soft revocation message, they hash her old public identity’s signing key and store this hash before deleting the key itself. This hash is added to a linked list that contains the hashes of all previous identity keys that Alice may have used (and revoked) since she deposited her trust in that user. This list of expired identity keys can later allow a user to stop trusting messages authored by Alice’s *PSUs* if her device is compromised.

Lastly, we note that a seized device may expose the identity keys of a user’s trusted contacts, enabling an adversary to link all activities tied to these identity keys, before affected users may be warned and issue soft revocations for those identities. However, Anix users can generate different identity keys to engage with different parties (see §2.3.2) which helps mitigate this issue.

Hard revocation. In the case where Alice’s device is seized or compromised by an adver-

Algorithm 7 Hard identity revocation

Input: Old identity’s signing keys ($PubK_{SID_{old}}, PrivK_{SID_{old}}$)

Output: Signal hard revocation of the old identity

- 1: $sig \leftarrow \text{Sign}_{PrivK_{SID_{old}}}(PubKE_{ID_{old}}, PubK_{SID_{old}})$
 - 2: $\text{revocationMessage} \leftarrow PubKE_{ID_{old}} || PubK_{SID_{old}} || sig$
 - 3: $\text{broadcastMessage}(\text{revocationMessage})$
-

sary, one must assume the adversary has gained complete access to Alice’s cryptographic material. This would allow the adversary to author messages on behalf of Alice and potentially leverage the trust that other users deposited in Alice to spread misinformation or manipulate their behavior. In such cases, Alice can use hard revocation as a last resort, compelling all users not to trust any activity from the revoked identity or any subsequent identities generated via “soft revocation”. Hard revocation requires Alice to store her current identity key pairs on some storage medium considered secure and not under the control of the adversary (e.g., stored in a flash drive Alice keeps at home). It also requires Alice to be able to interact with the Anix network with some Anix-compatible device.

Hard revocation works as follows. Alice signs one or more revocation certificates, where each certificate contains the public encryption and signing components of one of her current identity keys, signed by the identity’s private signing key. Then, Alice can join the mesh network with some device and broadcast messages containing these revocation certificates. Similar to a PGP revocation certificate [98], each message signals any node that includes this key as part of their trusted contacts to neither trust any further communications coming from the user identified by that key, nor any other identity key that may be a result of a soft revocation of the key mentioned in the certificate (should the adversary perform soft revocation itself before Alice had the chance to broadcast her revocation certificates). Users can track this by hashing the revoked identity and checking it against their list of hashes of Alice’s previous identities (as mentioned in “Soft revocation”). Algorithm 7 details the hard revocation steps. While hard revocations could also be preemptively issued by adversaries that seized a device (i.e., before Alice does it herself), this simply foils the adversary’s (temporary) opportunity to impersonate Alice.

After having described how Anix users can manage trust relationships, we will now address how Anix’s trust notions can help users rank and prioritize the messages they receive.

2.3.4 Message Endorsing and Ranking

We now describe votes and their role in the endorsement of Anix messages. Then, we detail how users can make use of votes, together with message authorship information, to reason about a message’s trustability.

Votes and message endorsement. Anix allows for message endorsement in the form of votes, which we consider to be *upvotes* or *downvotes*, used for respectively approving and disapproving a message. For instance, if Alice is responsible for forwarding a message whose content she knows is misleading (regardless of whether she has established a trust relationship with the message’s author), she can downvote that message to let her trusted contacts know about her opinion. As we will see later, messages can accrue votes over time, allowing Alice to issue votes on messages that she has already forwarded but had not yet voted on.

In practice, a vote is a signature of the hash of a message’s full data (including its content, sender’s *PSU*, and signature) denoted as M , concatenated with a single bit b , whose value is 1 for an upvote or 0 for a downvote. This signature incorporates the same public key blinding scheme used for *PSU* generation, as it uses the voting user’s identity key’s private signing component ($PrivKS_{ID}$), alongside their identity key’s public signing component ($PubKS_{ID}$) to sign the message in an unlikable fashion, i.e., $vote = \mathbf{BSign}_{PubKS_{ID}, PrivKS_{ID}}(H(M)||b)$. This ensures that only the users who know the identity key’s public signing component $PubKS_{ID}$ can identify the vote’s author and check the integrity of the received message. Recall that, in this case, bk is computed as $bk = H(M||b||PubKS_{ID})$.

Next, we showcase the utility of votes when reasoning about the trustability of messages exchanged through Anix.

Triaging incoming messages. Once Alice receives messages from another Anix node, she first attempts to pinpoint messages authored by her trusted contacts. To do so, she simply attempts to validate the signature included in the message’s sender *PSU* using each one of her contacts’ public identity keys. Then, Alice repeats a similar process to check whether some of her trusted contacts have cast any up/downvotes. The complexity of this process is of order $O(m \times c)$ where m is the number of all received messages that have to be verified, and c is the number of trusted contacts Alice possesses. While the delay to verify the signatures included in the *PSUs* scales linearly for both the number of contacts and messages, our experiments in §2.5.1 suggest that this process can be completed rather efficiently with each signature verification being completed within a few dozen milliseconds on commodity smartphones.

During message processing, nodes will also attempt to recognize *OWT* messages directed to them and either a) add the identity key included in a new *OWT* message targeted at them to their validation list, or; b) add the identity key obtained as a response to one of their previously sent *OWT* messages to their trusted contact list, thus completing the establishment of a bi-directional trust relationship.

Ranking trust on incoming messages. Once Alice verifies which of the received messages have been authored (and which votes have been cast) by her trusted contacts, she will compute a trust score (t) for each message, based on a formula ruled by the following factors: whether the message’s author is a trusted contact (TCA), and how many trusted contacts upvoted (TCU) or downvoted (TCD) the message. Though we do not directly consider it in our design, other factors (e.g., for how long a user has been trusted) could also be considered when computing a message’s trust score.

$$t = a.TCA + b.(TCU - TCD)$$

In this formula, a and b are coefficients used to tune the relevance given to each of the considered factors when computing a message’s trust score. This formula will, first, largely increase the trust score of the message should it be authored by one of Alice’s trusted contacts. The second most important component for attributing trust to a message is the number of up/downvotes generated by Alice’s trusted contacts. Note that no importance is given to the trend of up/downvotes cast over the message coming from users who are unknown to Alice. Should a message not be authored or voted upon by any of Alice’s trusted contacts, Anix displays these messages in a well-marked section of the app, outlining that they are not to be trusted since votes cast on them could have been artificially influenced by adversary-controlled nodes and do not necessarily reflect Alice’s trusted circle sentiment about a message.

Also, as previously discussed when mentioning Anix’s bi-directional trust relationships (see §2.3.3), the identities of users who have attempted to establish a bi-directional trust relationship with Alice but are still awaiting Alice to trust them back are kept in a validation list. Identities kept in the validation list are not accounted for in calculations depending on Alice’s trusted contacts and, therefore, do not affect the messages’ trust scores. This feature prevents adversaries from being able to artificially manipulate messages’ trust scores by inflating a user’s list of known public identities via the sending of *OWT* messages generated by Sybil identities.

Accruing message votes over time. Multiple copies of the same message can travel many different paths across the mesh network and eventually get delivered to the same user. Given that the votes that are tied to a given message will also be dependent on the

path each of these copies takes, Anix allows votes to be accrued at a given device (should different copies of the message be received). This allows Anix to dynamically recompute the trust score that should be attributed to a message and, for instance, accordingly reorganize messages in Anix’s microblogging feed.

Anix accrues votes on messages as follows. When receiving the first instance of a given message, Anix creates a string set [83] for holding the votes tied to that message (i.e., a *message vote set*). If a copy of the same message (and corresponding set of votes) is received again, Anix adds the new (non-duplicated) votes to that message’s vote set.

Since the number of votes on a message can become large, we hypothesize that Anix could use probabilistic data structures such as Bloom filters [97] or bit-string caches [188] to optimize the vote-exchanging process. However, these data structures are prone to false positives, which could result in some votes being inaccurately recorded or lost (even without adversarial involvement). Exploring alternative data structures that would allow for a more efficient exchange of votes is a compelling direction for future work.

2.3.5 Point-to-Point Message Exchange

This section describes Anix messages’ structure and how they are exchanged between peering devices. We start by focusing on the exchange of messages used in Anix’s one-to-many communication model before addressing messages tailored to Anix’s alternative communication models.

Message structure. The overall structure of a Anix message in the one-to-many setting consists of three sections:

1. *Message content:* For simplicity’s sake, each message in Anix is considered a text-only message.
2. *Sender’s PSU and signature:* A sender’s *PSU* and a signature of the message, based on that *PSU*.
3. *Votes:* A compressed representation of the votes cast on the message itself.

In §2.3.6, we detail the message formats used for Anix’s one-to-many communication, alongside those used for its one-to-one and some-to-some communication models. We discuss these communication models later in this section.

Endpoint discovery and selection. Similarly to existing work [192, 152], a user can discover other Anix users in her vicinity by using a mix of Wi-Fi Direct [47] and Bluetooth functionality. To make itself discoverable, a device sets the name of its Wi-Fi Direct beacon to some specially formatted name to broadcast the phone’s Bluetooth MAC address (we use `ANIX-{\text{MAC\_ADDRESS}}`). Once a device detects another endpoint following this naming convention, it attempts to connect to it via Bluetooth. If multiple endpoints are available, the device arbitrarily selects one and exchanges data with it, before moving on to the next. We use this hybrid approach since neither setting the name of a Wi-Fi Direct beacon nor connecting to another Bluetooth device whose MAC is known requires explicit user input.

Message exchange and storage. During a message exchange, both parties send the messages they hold in storage in a random order, along with their corresponding votes. We choose random ordering as opposed to prioritizing messages with higher trust scores since the latter could leak information about the trusted contacts of the forwarding node.

Considering that the devices where Anix is installed may also have limited space for storing messages, the app allows users to define a storage quota for saving messages and their associated votes. Anix then provides two mechanisms (a proactive one and a reactive one) for helping users manage the storage space that the app uses on their devices.

Firstly, to proactively help manage the growth of their message storage, Anix nodes keep a timestamp that records the time at which each message was received (rt_m). This timestamp is used in an aging process which is responsible for calculating the time at which a message should be deleted after being received (dt_m). Messages are deleted after being kept in storage for a persistence time (pt_m), such that $dt_m = rt_m + pt_m$. In addition, while we do not address this in our evaluation, the persistence time of different message types can be separately configurable. For instance, *OWT* messages can be configured to persist for a longer time than regular microblogging messages, allowing users additional time for reasoning about the intention to establish bi-directional trust relationships (see §2.3.3).

Secondly, Anix will reactively delete messages if an exchange causes the quota to be exceeded. To this end, the app will first randomly delete messages that have not been authored or voted upon by any trusted contact. In the event that all of these messages are deleted but the user is still above the storage quota, Anix further deletes the remaining messages according to their trust score (messages with lower trust get deleted first) in order to meet the quota.

Bounding message propagation. Epidemic routing often leverages a maximum hop count to determine the number of epidemic exchanges that a message is subject to [106, 234]. In Anix’s case, however, deterministically setting a hop count (even if randomly within some bound [203]) could leak information to an adversary about their proximity

to the author of a message [103]. To avoid disclosing information about the source of a given message while limiting the propagation of a message throughout the network, Anix keeps a table connecting a message’s hash to the corresponding timestamp of the time the message was received (rt_m). The message hash includes all the components of the message except the potential votes, so as long as the message is not tampered with, the hash remains static. This table’s entries persist even after a message has been deleted, so upon receiving a message, a node can refer to this table and verify whether the message has already been received, in which case the node acts according to the table entry for the received time, so if the local persistence time pt_m for this message has elapsed, the message will just get dropped. On the other hand, if the message was not previously received, the message gets added to the table and gets processed as normal. Table entries persist for a time duration $lt_m \gg pt_m$ to ensure that proactively deleted messages are not erroneously accounted again as new messages.

Alternative communication models. Besides its focus on message broadcasting, Anix also natively supports one-to-one and some-to-some messaging as follows.

One-to-one communication. Anix facilitates one-to-one (O2O) communication by encrypting the message’s payload with the receiver’s public key before transmitting it over the network. In contrast to microblogging messages that are meant to be interpreted by all mesh users, private O2O messages do not expose identifiable public metadata; their payload is concatenated with the sender’s key blinded signature over the message (see §2.3.2) while conforming to a fixed length (padded if needed), and then encrypted. The key used to encrypt a O2O message may be the public encryption key tied to a *PSU* (e.g., when generating an *OWT*), or, more commonly, the public encryption component of a temporary identity key (e.g., for private communication between two users who mutually trust each other).

Some-to-some communication. In Anix, group communication (S2S) is a direct extension of one-to-one messaging. To create a private messaging group, Alice generates a symmetric *group encryption key* and builds a group invitation message for each contact she wishes to invite by encrypting the group key with each user’s identity key’s public encryption component. Then, she broadcasts these messages. Once the intended users receive their group invitation, they can access and send messages encrypted with the group’s key. As in Anix’s O2O scheme, messages sent to a group can be signed using a key blinded signature. Should the group key be leaked, messages can be decrypted but cannot be trivially linked back to temporary user identities due to key blinding.

2.3.6 Anix Message Formats

In this section, we will detail Anix's data formats across different message types.

PSU structure.

PSUs include two main (see §1) components: 1) keys: including a public encryption key and a public signature verification key, and 2) a public key-blinded signature which is used to verify the senders identity for users who know them.

$PubKE_{OTU}$ (32 Bytes)	$PubKS_{OTU}$ (32 Bytes)
$B\text{Sign}_{PubK_{ID}, PrivK_{ID}}(PubKE_{OTU} PubKS_{OTU})$ (64 Bytes)	

One-to-many (O2M) message structure.

One-to-many messages are similar in structure to most micro-blogging messages, as they include the senders PSU, a signature verifying the authenticity of the message signed by the private counter part of the public key found in the PSU, and a message that could be up to 256 Bytes long.

PSU_{sender} (128 Bytes)	$\text{Sign}_{PrivKS_{OTU_{sender}}}(m)$ (64 Bytes)
Message m (max. 256 Bytes)	

One-way-trust (OWT) message structure.

One-way-trust messages are simple, as they are simply messages encrypted by a users public encryption key, that include the sender's public encryption and verification keys.

$$\text{Enc}_{PubKE_{OTU_{receiver}}}(PubKE_{ID_{sender}} || PubKS_{ID_{sender}})$$

(405 Bytes under ECIES, where the payload is the concatenation of public identity keys padded to 320B)

One-to-one (O2O) message structure.

One-to-one messages are similar to OWT messages in structure, as an OWT message is basically a specific case for a O2O message. An O2O message is a encrypted blob under the receiver's encryption key, that includes the contents of the message along a signature used for verification.

$$\text{Enc}_{PubKE_{ID_{receiver}}}(\text{BSign}_{PrivKS_{ID_{sender}}}(m) || m)$$

(405 Bytes under ECIES, where m is 256 Bytes – padded if needed – and the signature is 64 Bytes)

Some-to-some (S2S) group invitation message structure.

Following the same structure as O2O and OWT messages, some-to-some invitation messages are a blob of data encrypted under the receiver's public key, which includes the symmetric key used for the group communications.

$$\text{Enc}_{PubKE_{ID_{receiver}}}(SimKE_{group})$$

(405 Bytes under ECIES, where the payload is a symmetric group encryption key padded to 320 Bytes)

2.4 Evaluation Methodology

This section outlines our evaluation goals and the approach we used for measuring the quality of our solution.

2.4.1 Evaluation Goals and Approach

Our evaluation aims to assess the practicality of Anix over three main aspects:

Performance. We aim to assess the efficiency with which Anix can exchange messages between two devices. To this end, we conduct a set of micro-benchmarks based on a Anix software prototype, together with its deployment on real smartphone devices. We collect metrics such as the transmission time required for completing message exchanges, the computation time required to accomplish Anix’s basic operations, and Anix’s impact on battery consumption.

Resilience. We rely on simulations to gauge an adversary’s ability to successfully manipulate a user’s perception of misinformation transmitted through Anix and to build trust with legitimate Anix users. To assess the former, we compare the number of benign and misinformation messages that were up/downvoted by a majority of benign users. To assess the latter, we consider the number of adversarial nodes that received *OWT* messages authored by benign users.

Latency and coverage. We use simulations to gauge Anix’s message propagation times across the mesh when assuming the presence of different ratios of adversarial nodes.

2.4.2 Experimental Testbed

This section describes the deployment settings of our Anix prototype and the parameters used in our simulation.

Anix prototype and hardware settings. We implemented a proof-of-concept of Anix [1] and installed it in two different smartphones to perform our hardware-based micro-benchmarks. We developed our prototype for Android v8.0 (or later) using 2 200 lines of Java code and implemented a simple epidemic routing protocol for disseminating messages. In our implementation, we assume the payload of each O2M message to have a maximum size of 256B, and we explicitly pad the message payload of *OWT*, O2O and S2S messages to a fixed size to ensure these messages exhibit the same size on the network. We use BouncyCastle’s Java ECIES [164] implementation over secp256r1 for performing encryption/decryption of *OWT*, O2O, and S2S messages, and use the same library’s implementation of EdDSA [122] over Ed25519 for signing/verifying Anix O2M messages. We also implemented Denis et al. [70] EdDSA-based [122] key blinding scheme in Java to create unlinkable signatures used in *PSUs* and votes.

We deployed our prototype in a Samsung Galaxy A04 (ARM Cortex-A53 CPU, 4GB RAM) and ZTE Blade V40 Smart (Unisoc ums9230 CPU, 3GB RAM). We rely on Bluetooth v5.0 (or later) for performing point-to-point communication between devices due to its widespread adoption and because it allows for automatic message routing without the need to root the device (unlike Wi-Fi Direct) [152].

Mesh network simulator and considered parameters. We implemented a simulator (1500 lines of Python code) to reproduce the activities of legitimate Anix users and adversarial nodes within a city during a blackout. Below, we briefly describe the parameters we consider (see a full rationale for these choices in §2.4.3), and distinguish which of these are fixed throughout our simulations from those whose influence we directly evaluate in our experiments via different scenarios; we describe these scenarios later in this section, after introducing all parameters of interest.

Movement of Anix nodes. Based on ASMesh’s [37] simulator, we represent the world as an $A \times B$ discrete grid, where a total of N users, including a ratio of Adv adversarial nodes, move around and interact throughout T steps. At each step, each user can move up to m cells in any direction, exchanging messages with other users when co-located in the same cell. The movement of adversarial nodes follows the same rules. We consider a blackout with a duration of 5 days and one hour-long steps, resulting in a total of $T = 120$ simulation steps. We fix other parameters as $A = B = 25, N = 600, m = 2$. In practice, each cell of the grid is equivalent to an area of 0.25 square kilometers. Lastly, we leave Adv as a configurable value.

Message generation and storage. We assume that users pick up their phones and interact with the Anix app at any given simulation step with probability P_{inter} and create and send out new messages with probability C_m . We also assume each user configures the Anix app with a storage space of S GBs. Finally, users set the persistence time of each message (tp_m), in steps, to steer Anix’s proactive message aging and deletion. We fix all these parameters as $P_{inter} = 0.15, C_m = 0.05, S = 3, tp_m = 24$.

2.4.3 Rationale on Simulation Parameters

Table 2.2 summarizes the parameters of our simulations. Below, we detail the rationale for each parameter’s value.

Network scale. We drew the configurations for parameters (A, B, m, N, β) – related to the scale of the network – directly from ASMesh’s model [37]. We set $K=15$, meaning that, at the start of the simulation, each user is deemed as a trusted contact of an average 2.5% of all other users within a city (where the total number of users is given by $N=600$).

Table 2.2: Description of each parameter (and respective configuration) used in Anix’s mesh network simulator.

Parameter / Category	Description	Value	Rationale
A and B	Dimensions of the simulation world ($A \times B$ grid)	25×25	ASMesh’s [37] simulation (default)
m	Maximum distance that a user can move in a simulation step	2	ASMesh’s [37] simulation (default)
N	Total number of users	600	ASMesh’s [37] simulation (default)
β	Connectivity of the network given by the Watts-Strogatz model	0.5	ASMesh’s [37] simulation (default)
K	Average social graph degree in the Watts-Strogatz model	15	ASMesh’s [37] simulation (adjusted)
T	Total steps of the simulation (1 step = 1 hour)	120	5-day blackout period
Adv	Fraction of adversarial nodes amongst all users	2% – 25%	Increasingly stringent adversarial settings
S	Maximum device storage space allotted to the Anix app	3 GB	Reasonable storage use (128 GB/device total)
P_{inter}	Probability of a given user interacting with the Anix app at any step	0.15	Average daily social media apps usage
C_m	Probability for a user to send out a message in a given step	0.05	Twitter users’ posting behaviour
OWT_{ud}	Required ratio of a message’s known upvotes/downvotes to OWT the author	0.66	Performance/resilience trade-offs
U_{ud}	Required ratio of a message’s known upvotes/downvotes to upvote it	0.55	Performance/resilience trade-offs
R	Ratio of an adversary’s friends to benign user’s friends	0.1 – 0.9	Tuned during experiments
UV	Probability of a user who has no information about a message to vote on it	0.01 – 0.2	Tuned during experiments
UM	Probability that a user upvotes a message containing misinformation	0.1 – 0.5	Tuned during experiments
UN	Probability that a user upvotes a benign message	0.5 – 0.8	Tuned during experiments
t_{pm}	Persistence time of a message on a user’s device	24h	Performance & operational concerns

Simulation duration. We set $T=120$ simulation steps of 1 hour each (5 days) as this interval sits within a common duration for reported Internet shutdowns, e.g., during civil unrest and high-profile political events [170], also being longer than a sizable fraction of shutdowns consistently reported to last less than a day (e.g., as in India [163]).

Adversarial power. Our evaluation throughout § 2.5 assumed a ratio of 2% adversarial nodes ($Adv=0.02$) following the reasoning provided in Rangzen (see [152] – Sec.4.1). In §2.6, we provide results for the effects of larger Adv ratios on the degradation of Anix’s latency and coverage.

Anix device configurations. We set S to 3GB, amounting to 2% of the storage available in each of our smartphones.

User interactions with the Anix app. We fixed P_{inter} at 0.15, based on an average Anix usage of 4 hours per day ($4/24= 0.167$). This is consistent with recent surveys on social media app usage [216], from where we estimate an average total usage of social media apps (extrapolated from the combined usage time for the top seven trending apps) of $\approx 4h/day$. We fixed C_m at 0.05. While being conservative, we believe this value to be consistent with existing studies that revealed that most Twitter users rarely issue posts [255].

Anix-specific parameters. We fixed OWT_{ud} and U_{ud} at 0.66 and 0.55, respectively, based on preliminary experiments that suggested reasonable trade-offs in Anix’s performance and resilience. Parameters R and (UV, UM, UN) were tuned in § 2.5.2. We fixed t_{pm} at 24 hours after experimenting with values ranging from 1 hour to multiple days (up to 5 days, equaling to no deletion in our $T=120$ hours shutdown scenario), before converging on a day-long persistence for operational reasons: longer t_{pm} would make phone

Table 2.3: Messages majorly up/downvoted and OWTs issued by benign users, for different awareness scenarios ($Adv = 0.02$).

Scenario ($Adv = 0.02$)	Parameters				Benign		Misinformation		OWTs	
	R	UV	UM	UN	Upvoted	Downvoted	Upvoted	Downvoted	Benign	Adversarial
Very naive	0.9	0.2	0.5	0.5	495	2522	204	1164	33581	106
Naive	0.7	0.1	0.4	0.55	1087	1874	40	1301	32278	43
Default	0.4	0.05	0.3	0.6	1510	1416	25	1320	31207	11
Aware	0.2	0.02	0.2	0.7	2111	704	15	1314	26115	5
Very Aware	0.1	0.01	0.1	0.8	2549	348	5	1297	15497	2

storage run out faster, and, for larger values of T , force the storage allocated to Anix to be consistently maxed out (consequently prompting for forced message deletion).

Users’ social graph. To bootstrap the initial social graph between Anix users, we rely on the Watts-Strogatz model for small worlds [37, 250]. We assume that at the start of the simulation, some users already had the chance to become trusted contacts from one another, e.g., by manually exchanging identity keys or via Anix’s bi-directional trust establishment. The model uses two parameters, K and β , for the mean degree and graph creation randomness, respectively. We set $K = 15$ to reflect that each user is initially connected to 2.5% of the overall population and set $\beta = 0.5$.

As mentioned in §2.2, we assume the adversary is less connected with legitimate users than an average benign user. We reflect this through R , where $R = 1$ means that the adversary’s trusted contact circle is equally well connected to legitimate users as that of an average benign user, while $R = 0$ means that adversary nodes have no initial connections to benign users. In §2.5, we analyze the influence of R on an adversary’s ability to earn benign users’ trust.

Trusting messages and their authors. Our simulation also models the interactions between users (legitimate or otherwise) by simulating the creation of new trust relationships and interaction with messages from unknown sources. For simplicity during our simulations, we obviate specific values for the coefficients of our trust formula in §2.3.4 (a and b), and abstract them as the parameters mentioned below. We note however, that these coefficients can be experimentally calibrated as more realistic Anix usage data is gathered. OWT_{ud} and U_{ud} represent the minimum required ratios between the upvotes and downvotes on a message issued by a user’s trusted contacts so that a user chooses to send an OWT to the message’s author or, respectively, to upvote the message. We fix these as $U_{ud} = 0.55$, $OWT_{ud} = 0.66$.

In addition, parameter UV denotes the probability that a user chooses to vote on a message for which they have neither authorship nor voting information provided by a trusted contact. In case a user chooses to vote on such a message, parameter UM denotes

the probability that a user upvotes a message containing misinformation, and parameter UN denotes the probability that a user upvotes a legitimate message authored by a benign user. ($1 - UM$ and $1 - UN$ denote the probability that the user downvotes instead.) We vary these parameters throughout our experiments.

Anix user awareness-based scenarios. Adversaries targeting Anix will strive to spread misinformation while trying to gain the trust of benign users towards changing their perception of malicious messages. A fair part of the adversary’s effectiveness thus depends on how aware benign users are of the adversary’s tactics. Throughout our simulations, we rely on five different scenarios that aim to model the awareness of benign users and how this translates into the ability of Anix users to keep misinformation at bay. The more aware the benign user base is, the less they trust adversaries (R), vote on messages with unknown authors and voters (UV), and upvote misinformation messages (UM) because they can somehow be critical of them. They also identify benign users’ messages more easily (UN).

The scenarios we consider, as well as the values for the above parameters considered in each scenario, can be seen in Table 2.3. While the description of our simulation results (see §2.5.2) focuses on an adversary whose nodes make up to 2% of the overall Anix network ($Adv = 0.02$), §2.6 addresses more stringent scenarios where the adversary makes up to 5%, 10%, and 25% of the overall user-base within the Anix network. Nevertheless, the trends observed below still hold for these alternative scenarios.

2.5 Evaluation Results

We now analyze Anix’s performance and the impact of various attacks on Anix’s coverage and resilience. Then, we compare Anix’s resilience to that of Rangzen’s.

2.5.1 Micro-benchmarks

Latency of node discovery. We measured the time required for a Anix device to be able to find and connect to another device in its vicinity (via the hybrid Wi-Fi Direct and Bluetooth pairing described in §2.3.5). Our experiments revealed that our devices can complete this operation in $\approx 1.8s$.

Message exchange times. We measured the necessary time for our devices to conclude a message exchange. In this experiment, each device sends 100 messages to the other. Following Rangzen’s [152] average message payload sizes, we set the length of each O2M

Table 2.4: Computation time (in ms) for Anix’s operations.

Op./Device	Gen. PSU	Create Msg.	Create Vote	Verify Sig.	BVer (Alg. 3)
Samsung A40	175.06 ± 1.05	46.30 ± 0.01	84.61 ± 1.14	61.33 ± 0.21	67.68 ± 0.21
ZTE Blade V40	64.95 ± 0.29	19.75 ± 0.01	38.76 ± 0.32	43.29 ± 0.28	47.30 ± 0.48

message payload to 140B. This amounts to a total of 332B, accounting for each message’s corresponding *PSU* (128B) and signature (64B). We spread 10 000 votes (64B each) amongst the messages held by each device, amounting to a total communication volume of ~ 1300 KB. Our results reveal that this exchange is completed in 11.58s, indicating an average Bluetooth transmission speed of ~ 1.28 Mbps (akin to recent work [188]).

Computation time for basic operations. We measured the time required to complete the operations that support Anix’s main functionalities, namely: generating *PSUs*; creating messages (excluding the generation of the *PSUs* that are appended to them); creating votes, and; verifying signatures (i.e., the backbone of Anix’s operations when processing received messages). Table 2.4 shows the time (in milliseconds) spent on each operation, averaged after 10 000 trials. These numbers suggest that Anix’s basic operations can be performed swiftly and that the necessary cryptographic operations to assess messages’ authenticity and triage incoming messages/votes (see §2.3.5) do not add substantial overheads.

Battery consumption. We use Anix to continuously exchange messages between two devices for a full hour and measure the battery consumption imposed when running the app through Android’s internal battery monitor. We find that Anix consumes 87.68mAh charge on our ZTE device per hour, which amounts to 1.5% of its 6000mAh battery.

2.5.2 Simulations

Anix users can weed out misinformation. Table 2.3 shows how many benign (and misinformation) messages were up/downvoted by a majority of benign users across different user awareness scenarios, suggesting that the more aware users are of the adversary, the less pronounced the effect of the adversary’s actions; users tend to distrust misinformation and better assess the trustworthiness of benign messages. However, even for the *naive* user scenario, only 40 out of 1341 misinformation messages are majorly upvoted.

Anix users can avoid trusting adversarial nodes. Table 2.3 also shows how many benign/malicious users were able to gain the trust of other benign users (via *OWT*). We see

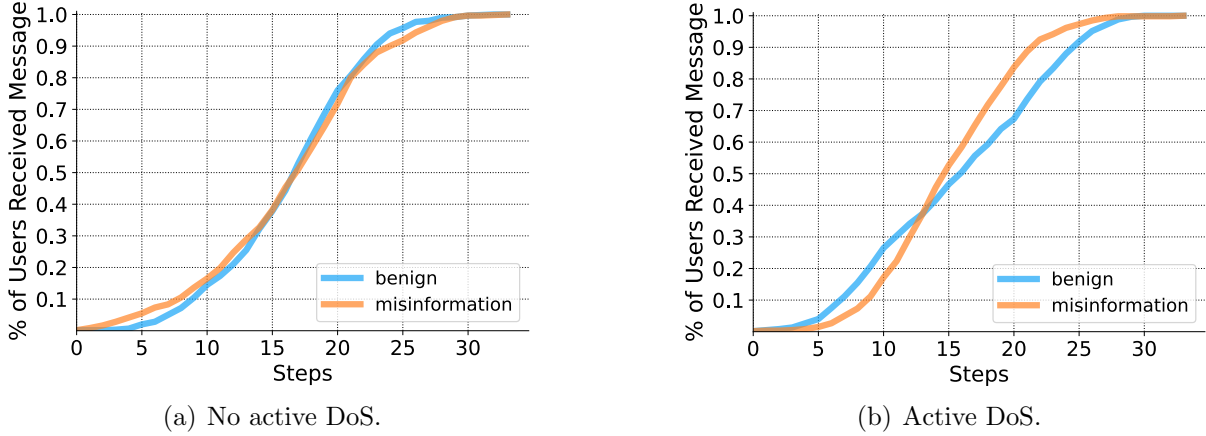


Figure 2.1: Anix’s coverage in function of message delivery latency.

that the vast majority of *OWT* messages are targeted at benign users instead of adversarial nodes. While users in the default or aware scenarios hardly ever gain enough confidence to trust adversary nodes (e.g., 11 *OWT* messages in the *default* scenario), Anix prevents even *Very naive* users from heavily trusting adversarial nodes (106 *OWT* messages). Users that mistakenly trust the adversary may still make use of identity revocation primitives (see §2.3.3) at some point in time.

Anix achieves a reasonable coverage and latency. We measured the time it takes for new messages to reach different fractions of the network in the *default* user awareness scenario (an analysis for other scenarios can be found in §2.6). We assess this metric separately for benign and misinformation messages to understand whether the presence of adversarial nodes can favor the spread of misinformation. As depicted in Figure 2.1(a), our results show that, on average, a benign message takes 23 steps to reach 90% of all Anix users. As each step correlates to one real-world hour in our simulation, an average message takes roughly a day to reach more than 90% of the total users of the network. The figure also shows that, in this scenario, misinformation messages move at a similar speed across the mesh.

DoS attacks do not significantly degrade coverage or latency. In addition to the above, adversary nodes can choose not to forward any kind of benign messages, and spread only misinformation messages authored by themselves. Figure 2.1(b) reveals that, while misinformation messages can now travel faster through the mesh (reaching 90% of all users in about 21 steps), these attacks do little to impede legitimate communications since the average time to deliver benign messages to 90% of all Anix users only increases 2 steps (up to 25) when compared to the absence of DoS.

2.5.3 Comparing Rangzen’s and Anix’s Resilience

In this section, we compare the resilience of Anix to that of Rangzen’s in regards to the spreading of misinformation messages. To perform this comparison, we reproduced the operation of Rangzen in our mesh simulator (with the same world and adversary configurations used for Anix in §2.4.2).

As described in §2.1.3, the messages that Rangzen shows to users are prioritized based on their trust scores. Further, messages below a certain trust threshold T_{thr} can also be hidden from users to limit the spreading of misinformation [152]. Given this, we configured an experiment aimed at assessing how many benign and misinformation messages are effectively shown to (i.e., deemed trusted by) more than 50% of Rangzen’s users. (This is intentionally reminiscent of the concept of majorly upvoted benign/misinformation messages in Anix to facilitate comparisons.)

Table 2.5 depicts the results of our experiments for different trust thresholds T_{thr} (we omit results for $T_{thr} > 1e - 07$ as they remain unchanged). Our results suggest that, without defining a threshold ($T_{thr} = 0$), there is a substantial risk that Rangzen may allow adversaries to disseminate misinformation – indeed, 1184 of the misinformation messages exchanged throughout the simulation are shown to at least 50% of users. While implementing a threshold helps reducing the number of misinformation messages shown to users, this also limits the number of benign messages shown to them. For instance, while 171 misinformation messages are seen by at least 50% of users when $T_{thr} = 1e^{-10}$, only 369 benign messages get to be seen ($\approx 2\times$ vs. misinformation).

In contrast, Anix achieves a more effective balance between the benign messages shown to users and the misinformation messages it can weed out. For instance, even when users are *naive* about the activities of the adversary (see Table 2.3), Anix users only majorly upvote (i.e., trust) 40 misinformation messages while majorly upvoting (i.e., trusting) 1087 benign messages ($\approx 27\times$ vs. misinformation).

2.6 More Stringent Adversarial Settings

In this section, we will focus on more stringent versions of the evaluation results presented in previous sections, to show Anix’s resiliency and practicality even in the harshest scenarios.

A. Weeding out misinformation with additional adversarial nodes. Tables 2.6, 2.7, and 2.8 showcase the number of majorly up/downvoted benign and misinformation mes-

Table 2.5: # of majorly shown/not shown messages in Rangzen.

Msg. Status	T_{thr}	0	$1e^{-13}$	$1e^{-12}$	$1e^{-11}$	$1e^{-10}$	$1e^{-8}$	$1e^{-7}$
Shown to >50% users	Benign	2901	1917	1558	769	369	3	0
	Misinf.	1184	868	703	305	171	0	0
Hidden from >50% users	Benign	600	1584	1943	2732	3132	3498	3501
	Misinf.	218	534	699	1097	1231	1402	1402

sages, for scenarios where the proportion of adversarial nodes in the network is increased to 5%, 10%, and 25%, respectively.

We can observe that, in general, the numbers shown in the tables follow the trend observed in our running example throughout §2.5.2 when the proportion of adversarial nodes is set to 2%. Briefly, a user base that is more aware of an adversary’s activities can better distinguish benign from misinformation messages. While our results suggest this task is increasingly difficult for a larger proportion of adversary nodes in the network (where misinformation messages tend to be more leniently upvoted), Anix can still ensure that misinformation messages are usually perceived negatively. In addition, the tables also show the amount of *OWT* messages received by adversarial nodes throughout the simulations. Our results suggest that the advantage of an adversary in gaining the trust of benign users remains limited, despite the adversary’s increased presence.

B. Coverage and latency for an increased adversary presence (*default scenario*)

Figure 2.2 depicts the coverage and latency of Anix in the default user awareness scenario, while increasing the presence of adversarial nodes in the network to 5%, 10%, and 25% and assuming adversaries choose not to forward benign users’ messages. We can observe that, while the presence of additional nodes tends to favor the faster spread of misinformation messages, Anix can tolerate this attack and still ensure the delivery of benign messages to 90% of the mesh in about 25 simulation steps. This is aligned with the results reported in §2.5.2, where one can see that the epidemic nature of Anix’s message exchanges can adequately overcome DoS attacks brought forth by adversarial nodes. However, we can also see that the effects of the ratio of adversarial nodes brings a more pronounced effect on latency in the initial stages of a message’s propagation. For instance, for benign messages to reach 30% of users under a 2%, 5%, 10%, and 25% adversarial node ratio, they take on average 11, 12, 14 and 16 steps, respectively.

C. Coverage/latency on different scenarios. Figure 2.3 depicts the coverage and latency of Anix in alternative user awareness scenarios. We see that users who are more

aware of the adversary’s activities can positively influence the propagation speed of benign messages, when compared to naive counterparts (where we observe the opposite effect). For instance, for the *very aware* scenario, benign messages reach 90% of all users in the mesh after 19 steps, whereas this number grows to 26 steps in the *very naive* scenario.

Table 2.6: Anix’s resilience to misinformation ($Adv = 0.05$).

Scenario ($Adv = 0.05$)	Benign		Misinformation		OWTs	
	Upvoted	Downvoted	Upvoted	Downvoted	Benign	Adversarial
Very naive	444	2496	517	2834	45218	353
Naive	1002	1921	121	3227	59541	269
Default	1462	1423	44	3182	59861	185
Aware	2096	761	34	3236	44947	125
Very Aware	2614	285	11	3212	37155	76

2.7 Related Work

Blackout-resistant data exchange. In addition to the use of mobile devices to build ad-hoc mesh networks, other research efforts have analyzed the deployment of scalable data exchange mechanisms that can resist blackouts.

Community-based mesh networks. These networks rely on users who coordinate their ef-

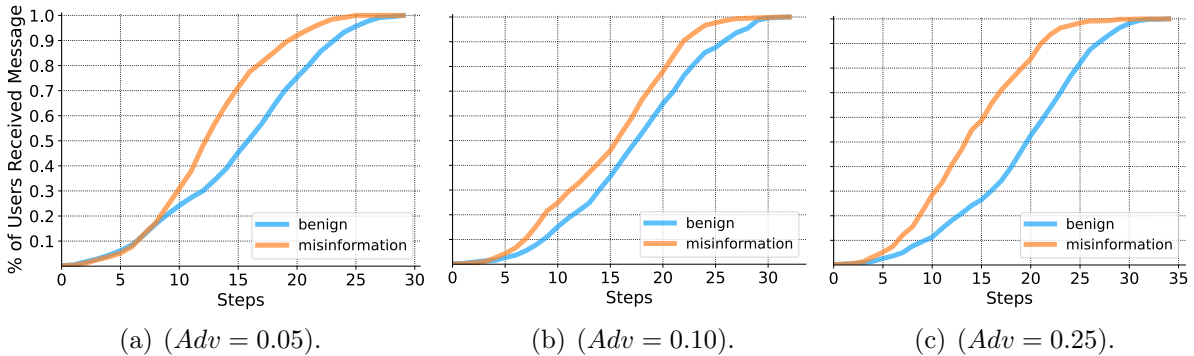


Figure 2.2: Anix’s coverage (under active DoS) in function of message delivery latency with increasing proportions of adversarial nodes.

forts to build resilient communication infrastructure making use of wireless and/or wired mediums [4]. Besides extending coverage to remote regions, these meshes have been touted to help resist internet censorship campaigns and increase net neutrality [67, 132]. Prominent examples include the `guifi.net` [29] network in Spain and AlterMundi project [119] in Argentina. In contrast to Anix, these networks demand more specialized communication equipment with higher maintenance efforts [95].

Satellite-based communication. Other solutions for tackling shutdowns have explored satellite-based data transmissions. Projects such as Toosheh [272] and Outernet [127], embed digital content within video streams transmitted over satellite TV channels. Yet, these can only be used to consume content that is curated by the projects’ operators. Recently, Starlink has enabled Internet connectivity to disconnected areas during warfare [28]. However, satellite internet alternatives require users to possess specialized equipment which oftentimes must be smuggled into affected regions [238].

Cellular-based data exchange. Dolphin [207] allows for the exchange of data by modulating it over cellular voice connections. In contrast to Anix, Dolphin assumes that cellular voice infrastructure is operational during shutdowns.

Practical anonymous communication systems. There are several practical and usable anonymous communication systems, such as Freenet/Hyphanet [57], Tor [73], I2P [275], or Nym [71]. Anix draws inspiration from these systems but aims to provide a secure platform for anonymously exchanging information during internet shutdowns.

Privacy-preserving reputation systems. Solutions like reputation transfer systems, or coin- and ticket-based reputation systems can provide reputation tracking without hindering users’ privacy [104]. However, some of these solutions rely on trusted entities for handling the management of reputation [44], while others require internet access [23]. In

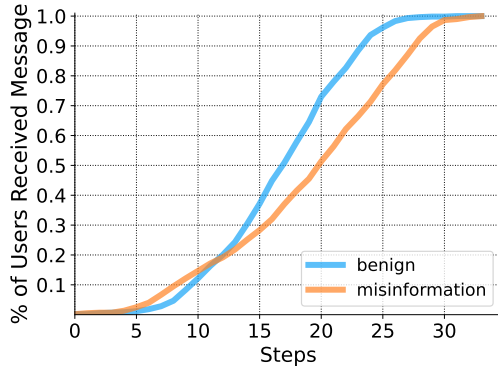
Table 2.7: Anix’s resilience to misinformation ($Adv = 0.10$).

Scenario ($Adv = 0.10$)	Benign		Misinformation		OWTs	
	Upvoted	Downvoted	Upvoted	Downvoted	Benign	Adversarial
Very naive	519	2328	1238	5512	44430	261
Naive	947	1829	192	6522	40319	159
Default	1392	1437	109	6547	32883	73
Aware	2016	726	57	6481	22193	64
Very Aware	2308	347	35	6333	12961	41

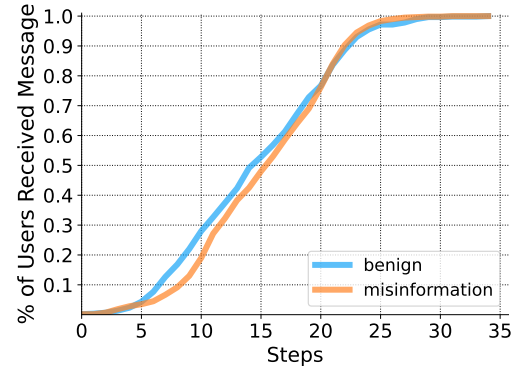
turn, Anix allows a mesh network’s users to progressively build trust relationships through the analysis of messages’ content and message endorsements, while retaining anonymity.

Table 2.8: Anix’s resilience to misinformation ($Adv = 0.25$).

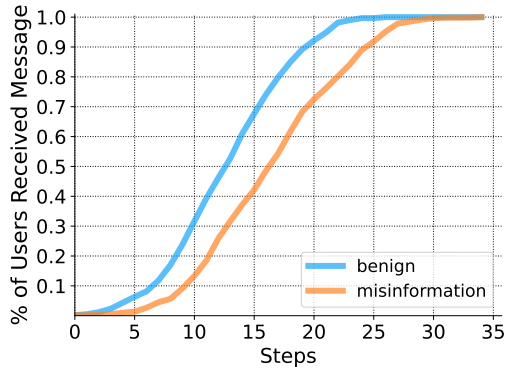
Scenario ($Adv = 0.25$)	Benign		Misinformation		OWTs	
	Upvoted	Downvoted	Upvoted	Downvoted	Benign	Adversarial
Very naive	528	1736	3954	12749	36193	11
Naive	1002	1146	584	16157	15554	2
Default	1269	919	284	16238	6444	1
Aware	1545	527	149	16067	3350	0
Very Aware	1837	285	79	15953	1961	0



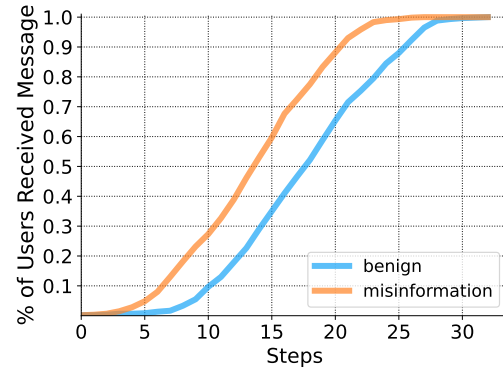
(a) Coverage in the aware setting.



(b) Naive.



(c) Very aware.



(d) Very naive.

Figure 2.3: Anix’s coverage and latency in alternative awareness scenarios.

Chapter 3

Huma: Censorship Circumvention via Web Protocol Tunneling with Deferred Traffic Replacement

In this chapter, we will motivate and present Huma, a censorship circumvention tool that operates via web protocol tunneling using deferred traffic replacement to avoid detection, while leveraging anonymous credentials to allow private usage even when using compromised endpoints. In §3.1, we analyze previous intermediary-based censorship circumvention tools, outlining Huma’s design goals in the process. In §3.2, we outline Huma’s threat model assumptions, before introducing Huma and its functions components without diving deep into the inner workings. Later in §3.3, we detail Huma’s inner workings and functions, while describing how the components interact together, resulting in achieving Huma’s goals. Finally, in §3.4 we evaluate Huma’s resistance to content fingerprinting, its observability, and measure its throughput in various scenarios.

3.1 Intermediary-based Circumvention Tools

We now examine the landscape of censorship circumvention tools that rely on intermediary nodes to fetch blocked content, such as proxies or specialized routers that inconspicuously redirect user traffic. We classify prominent tools based on three dimensions that reflect their security features and operational deployment needs (see Table 3.1), aiming to showcase whether and how each tool: a) resists detection against traffic analysis; b) provides privacy

Table 3.1: Categorization and classification of censorship circumvention tools related to Huma.

Category	Circumvention System	Traffic Analysis Resistance		Safeguards against Sybil Proxies			Deployment Environment			
		Unobservability	Behav. Real.	Dest. Conceal.	Dest. Conceal.	Inf.	IRI	CEI	WEB	SERV/END
Refraction Networking	Slitheen [41]	✓	✗	✓		✗	•			
	Waterfall of Liberty [179]	✓	✗	✓		✗	•			
	Conjure [91]	✓	✗	✓		✗	•			
CDN-Assisted	Cache Browser [113]	✓	✗	✗		-		•		
	CDN Reaper [280]		✗	✗		-		•		
	Domain Fronting / <i>meek</i> [86]	✗	✗	✓		✗		•		
	Domain Shadowing [251]	✓	✗	✓		✗		•		
Moving Target Defenses	Snowflake [40]	✗	✗	✓		✗				•
	NetShuffle [138]	✗	✗	✓		✗		•		
	SpotProxy [139]	✗	✗	✓		✗		•		
Fully Encrypted Protocols	obfs4 / Lyrebird [222]	✗	✗	✓		✗				•
	Shadowsocks [182]	✓	✗	✓		✗				•
Programmable Frameworks	Proteus [241]	✓	✗	✓		✗				•
	WATER [52]	✓	✗	✓		✗				•
	UpGen [242]	✓	✗	✓		✗				•
Protocol Mimicking	StegoTorus [253]	✗	✗	✓		✗				•
	SkypeMorph [172]	✗	✗	✓		✗				•
	FTE [75]	✗	✗	✓		✗				•
	Marionette [76]	✓	✗	✓		✗				•
Protocol Tunneling	Protozoa [32]	✓	✗	✓		✗				•
	Camoufler [206]	✓	✗	✓		✗				•
	Telepath [217]	✓	✗	✗		-				•
	Raven [243]	✓	✓	✗		-				•
Web Protocol Tunneling	WebTunnel [208]	✗	✗	✓		✗			•	
	HTTPT [93]	✗	✗	✓		✗			•	
	Balboa [199]	✓	✗	✓		✗			•	
	Huma	✓	✓	✓		✓			•	

safeguards against Sybil proxies, and c) enables for a straightforward deployment while providing reasonable collateral damage. Throughout our analysis, we establish a suite of design goals Huma is set to achieve.

3.1.1 Traffic Analysis Resistance

Unobservability. Regarded as the ability for circumvention tools to resist traffic fingerprinting [114], unobservability is typically achieved by having circumvention tools generate traffic patterns that blend in with those of legitimate (and inconspicuous) applications. Fingerprinting attacks involve an adversary that relies on the interception and analysis of network traffic metadata (e.g., packet timing or size information, communication volume) to detect the use of censorship circumvention tools among legitimate network traffic [268]. Though mostly conducted in a passive fashion, fingerprinting may also rely on the active manipulation of network conditions [114] or interaction with suspected circumvention endpoints [78, 92] to elicit inconsistencies between the traffic generated by circumvention tools and that of legitimate applications [114].

Refraction networking systems rely on backbone routers provided by friendly Internet service providers (ISPs) to relay steganographically-marked packets aimed at covert

destinations while seemingly connected to an unblocked overt destination [262, 115, 128]. Slitheen [41] resists traffic fingerprinting by hiding covert content within the leaf HTTP elements of an overt website’s response, thus preventing the generation of traffic patterns that would arise from accessing the covert destination directly. Follow-up systems consider traffic fingerprinting attacks [179], allowing compatibility with existing defenses (e.g., as in the case of Conjure [91]).

CDN-assisted tools make use of content delivery networks (CDNs) to evade detection. For instance, CacheBrowser [113] enables access to censored content stored in CDN caches, but was shown to be susceptible to traffic fingerprinting. This was later addressed by CDN Reaper [280], which adds a scrambling unit that alters the volume of traffic exchanged between clients and endpoints. Domain fronting/*meek* [86] hides covert traffic (e.g., aimed at a proxy backend) inside HTTPS connections seemingly targeted at allowed hosts on CDN domains. However, domain fronting also does not defend against traffic fingerprinting [247]. Alternatives following similar techniques, such as domain shadowing [251], provide hypothetical solutions against these attacks by segmenting data requests.

Moving target defenses such as SpotProxy [139] and NetShuffle [138] aim to change proxy endpoints faster than the censor can block them. While being compatible with existing traffic shaping [257] schemes, these tools do not inherently resist traffic fingerprinting. Snowflake’s WebRTC traffic patterns can be identified [85, 162], but considerations exist [40] to obfuscate them resorting to Encoded Transforms (as in TorKameleon [239]) or media embedding (as in Protozoa [32]).

Fully encrypted protocols (FEPs) were designed to be hard to detect, and shroud arbitrary data within a layer of encryption, effectively making generated traffic look like random bytes. Modern FEPs shape these random bytes to look similar to usual traffic (e.g., HTTPS connections), aiming to make them resistant to traffic fingerprinting. Yet, widespread tools such as Shadowsocks [182] and obfs4/Lyrebird can be detected through entropy tests [19, 258, 247].

Programmable frameworks are designed to disguise covert traffic as multiple configurable protocols. Proteus [241] enables developers to define custom protocol behaviors and transformations to bypass blocking. WATER [52] packages evasion techniques as portable WebAssembly modules, allowing different tools to share and deploy new circumvention transports without changes to the underlying applications. UPGen [242] builds on similar ideas and automatically generates vast swaths of plausible encrypted protocols that resemble typical encrypted traffic but which are distinct from known protocols.

Protocol mimicking tools aim to imitate a popular and uncensored network protocol. The expectation is that a censor may refrain from blocking this traffic due to potential

economical and/or political collateral damage [84]. For instance, Skypemorph [172] mimics videoconferencing traffic, while StegoTorus [253], imitates HTTP server traffic. However, they failed to account for implementation quirks that could be exploited by adversaries to find inconsistencies in the mimicked traffic, rendering them observable [114]. Format-Transforming Encryption (FTE) [75] generates ciphertexts that conform to the expected format of an allowed protocol, but is vulnerable to entropy tests [247]. In later efforts, Marionette [76] offers programmable mimicry of stateful protocols; however, imitation involves a labor-intensive effort [114].

Protocol tunneling tools tackle the difficulties of providing faithful protocol imitations by directly embedding data into the inputs of a cover protocol. For instance, FreeWave [116] modulated covert data into audio transmitted via VoIP calls. However, despite adhering to the cover protocol’s specification, mismatches between the characteristics of legitimate/covert inputs transmitted via the cover protocol could still reflect on abnormal traffic patterns, making this and other similar tunneling approaches vulnerable to detection [96, 33]. Recent tools enhanced tunneling mechanisms, ensuring that the injection of covert data will not lead to the cover protocol to produce abnormal traffic patterns. They rely on applications such as e-mail (Raven [243]), videoconferencing (Protozoa [32]), games (Telepath [217]), or instant-messaging (Camoufler [206]).

Web protocol tunneling tools can be perceived as a sub-category of the aforementioned systems and aim to circumvent censorship by embedding data in overt web requests to websites that act as proxies to covert destinations. In HTTPPT [93] and WebTunnel [208], the client initiates a standard TLS connection, then upgrades it to a WebSocket while embedding the covert destination’s address in the upgrade request. The overt website relays this to the covert destination and completes the WebSocket handshake, enabling the client to communicate with the covert server through the established WebSocket channel. Although these methods aim to blend with legitimate traffic, they modify the expected communication behavior with the overt website, failing to preserve expected traffic patterns [93, 268]. Similarly to Slitheen [41], Balboa [199] exchanges leaf content on TLS-protected elements (such as a website’s HTML, CSS, images, and videos) to resist traffic fingerprinting. However, data replacement introduces timing variations which may allow censors to detect Balboa [199].

Behavioral realism. So far, we focused on understanding whether a circumvention tool can produce a network flow that can plausibly sit among the legitimate flows produced by a cover network application. *Behavioral realism* prescribes that the observable behavior of the cover application should be similar to the behavior exhibited by typical, genuine usage of the application [243]. As an example, the use of a circumvention tool that inconspicuously tunnels arbitrary data via e-mails might become detectable by a censor that observes

a user exchanging hundreds of e-mails per minute.

Few existing circumvention tools account for behavioral realism. Raven [244] operates via e-mail exchanges and targets a realistic e-mail usage inspired by users’ historical behavior; it accomplishes this by training a generative model based on existing datasets of e-mail exchanges (considering their frequency, length, and sending times), restricting covert data exchanges to follow these patterns. OUStral [158] is an add-on meant for web traffic replacement systems (e.g., Slitheen [41], Balboa [199]) which aims to autonomously browse the web as a benign user would. It generates activity patterns by following probabilistic models designed after a decade-old human browsing studies (e.g., [74, 240, 156]). OUStral-generated activity can bypass bot detection filters, but does not adhere to users’ historical browsing profile.

Design goal 5. Protect against traffic fingerprinting attacks and provide behavioral realism guarantees.

3.1.2 Safeguards against Sybil Proxies

The deployment of strong privacy safeguards against Sybil proxies is a desirable (but often overlooked) requirement in scenarios in which a censor may be willing to deploy its own proxies among the legitimate circumvention infrastructure. The lack of these safeguards allows censors to temporarily allow user connections to blocked destinations while silently observing user activities, thus potentially collecting incriminating evidence that could be used for prosecution [266].

Destination concealment. This property refers to a privacy protection offered to the users of circumvention tools, in which the identity of the intended destination of circumvention traffic is hidden from the proxy handling the user’s traffic. Most intermediary-based circumvention tools acknowledge this risk, treating proxies as untrusted, but relegate destination concealment to auxiliary schemes—typically onion routing via Tor [73]. In this deployment model, client traffic is encrypted and routed through Tor, preventing proxies from directly identifying a user’s final destination. Secure connections to the Tor network can be established twofold [231]: a) directly, having the proxy act as the first hop in a Tor circuit—such as in Conjure, *meek*, Lyrebird, or WebTunnel; or b) indirectly, first to the proxy server itself, and then through Tor for a total of four hops between client and destination—such as in Shadowsocks, Snowflake, or Camoufler. While this form of destination concealment is not intrinsic to the circumvention tools themselves, most tools mentioned in Table 3.1 support it.

Despite aiding in destination concealment, Umayya et al. [231]’s study reveals that pairing Tor with existing circumvention approaches aimed at evading detection is rather slow and unreliable. Moreover, given some tools’ (e.g., Telepath, Raven) focus on achieving strong traffic analysis resistance, they lack the necessary throughput and latency requirements to establish connections via Tor, and thus to achieve destination concealment through it. This calls for a native mechanism for achieving destination concealment which does not depend on other sub-systems whose integration may be unfeasible or further hamper network performance.

Destination concealment against inference. While destination concealment mechanisms ensure that the destination of circumvention traffic is not outright revealed to a proxy operator (e.g., should circumvention traffic be forwarded via Tor), a censor who controls a proxy may still deduce a circumvention tool user’s destination using techniques such as website fingerprinting [209]. This is similar in spirit to attacks that assume an adversary may become a guard node for a user’s Tor traffic, thus being in the position to launch this kind of traffic analysis attacks [123]. A tool offering destination concealment against inference would explicitly thwart such attacks. Yet, as we can observe in Table 3.1, no existing circumvention tool provides these safeguards.

Design goal 6. Develop native privacy safeguards against Sybil proxies for censorship circumvention tools.

3.1.3 Deployment and Collateral Damage

We define deployability as the extent to which a circumvention tool can be set up and operated by entities involved in the anti-censorship enterprise, ranging from requiring significant third-party/industry support to being fully self-deployable by volunteers. In general, tools that are harder to deploy also tend to be harder to block, as doing so entails greater economic/political collateral damage to censors [138].

Internet routing infrastructure (IRI). Refraction networking tools depend on the cooperation of routing infrastructure operators, typically ISPs, who are willing to operate and support the costs for the routing hardware required to enable the service [90, 237, 42]. However, this dependence on broad ISP participation poses a significant deployment challenge, as the associated costs can be substantial [100], and ISPs may be hesitant to align themselves with circumvention initiatives [138]. Still, if successfully and widely deployed, refraction networking becomes difficult to disrupt; censors attempting to divert network traffic away from Internet routes that deploy refraction networking face high operational costs [178, 42].

Cloud- and edge-native infrastructure (CEI). Cloud and edge-based infrastructure, such as CDNs, organizational networks, and cloud platforms, provide the substrate for several circumvention tools. In certain cases, circumvention tools exploit features provided by these infrastructures in originally unintended ways, leading service providers to remove key features leveraged for circumvention (e.g., the disabling of domain fronting by some CDN providers [20, 25]). While new tools frequently emerge to adapt to such changes (e.g., domain shadowing [251] on CDNs, or high-rotation proxy deployments [139] on transient cloud instances), their reliance on undocumented and/or prone-to-change platform behaviors also places these tools toward the end of the deployability spectrum. Other tools, such as NetShuffle [138], explicitly rely on individual edge network operators, making them relatively easier to deploy when compared to other circumvention systems deployed in industry-backed CEI infrastructure. Overall, circumvention tools within the CEI category impose significant collateral damage if blocked, as censors often need to block CDNs or entire IP address blocks to effectively deter them.

Websites (WEB). Another set of circumvention tools depends on the cooperation of website operators. Certain tools in this category depend on specific functionalities as prerequisites for deployment, potentially limiting the pool of websites suitable for aiding circumvention efforts. For instance, HTTPPT [93] and WebTunnel [208] require websites that support WebSocket to establish a covert channel (§3.1.1). Yet, WebSocket is mostly used for real-time communication workloads and its deployment has stagnated, with a reported adoption rate as low as 6.3% in 2021 [174]. Instead, tools such as Balboa [199] or cloak [246] are straightforward to deploy, requiring only minimal configuration via plug-and-play software modules. Tools in this category pose a moderate risk of collateral damage for censors; the more popular the cooperating website is, the greater the economic or political consequences the censor may incur by blocking it [93].

Commodity servers and endpoints (SERV/END). The most basic requirement for a proxy-based circumvention tool is simply the availability of a server. As a result, most tools within traffic mimicking, traffic tunneling, and fully encrypted protocols belong to this category—requiring only a machine capable of running the proxy, whether it’s a dedicated server or a temporary, ad-hoc endpoint such as a laptop [40]. Thus, tools in this category are generally the most straightforward to deploy by volunteers, requiring no third-party support. However, ad-hoc proxy support—such as that provided by friends or family members operating protocol tunneling proxies [32, 206]—limits broader accessibility, leaving the majority of users in need of circumvention without viable options. This ends up hindering the scalability and widespread adoption of such tools. Moreover, should a censor be able to find a reliable method to identify these proxies, it can block them with little to no collateral damage implications [138].

Design goal 7. Enable reasonable collateral and low 3rd-party dependence circumvention via Web deployments.

3.2 Huma

We now introduce Huma, a new censorship-resistant communication tool based on website deployments. This section describes Huma’s threat model, architecture, and workflow.

3.2.1 Threat Model

Adversary’s capabilities. We consider a state-level adversary whose main goal is to *identify* any Internet-based information flows it deems sensitive, and which traverse both within and across the country’s border. This adversary controls all ISPs within the censored region, enabling passive interception of all network traffic within and across national borders. In practice, real-world adversaries are known to deploy traffic analysis apparatuses at cross-border [110], regional [261], and AS-level [269] vantage points. The adversary can deploy deep packet inspection (DPI) filters to detect objectionable content and apply advanced traffic analysis techniques, e.g., based on machine learning, to identify the use of circumvention tools.

Instead of immediately blocking information flows deemed objectionable, we consider our adversary to be interested in learning about the Internet destinations and/or contents that circumvention tools’ users might access, to later prosecute them with aggravating legal grounds. Thus, the adversary may compromise existing proxy-based circumvention infrastructure and/or deploy Sybil circumvention proxies to tap on users’ connections and unveil the destination of circumvention traffic. This concern is supported by the study of Xue et al. [266], where participants expressed concerns that circumvention tools could be operated as honeypots designed to monitor user activities. Also, should the circumvention tools leveraged by users provide destination concealment (see §3.1.2), the adversary will attempt to fingerprint the encrypted connections established through proxies under its control to infer users’ destinations.

Lastly, we admit that the adversary will choose to block any information flows it deems objectionable should it be unable to inspect their contents or observe/infer their intended destination. This will be the case should: a) the adversary be able to compromise the proxy infrastructure, but users leverage circumvention tools that provide destination concealment

against inference, or; b) in cases where the adversary has no control over proxies, and may only resort to traffic analysis to identify (and then block) circumvention tools.

Adversary’s limitations and out-of-scope attacks. We assume an adversary that seeks to minimize collateral damage from coarse-grained measures such as applying a blanket ban to HTTPS connections. The adversary is also computationally bounded and unable to break standard cryptographic primitives (e.g., allowing it to snoop into TLS-protected traffic). Like other web protocol tunneling tools (e.g., Balboa [199]), Huma also relies on the assumption that TLS connections are not intercepted by censors. In adversarial environments where the use of TLS is man-in-the-middle (e.g., by forcing users to install rogue root certificates) [195], the usage of Huma would become detectable. However, such scenarios remain relatively rare due to the high operational complexity involved and pressure from international peers. For instance, Kazakhstan’s earlier attempts at traffic interception led the local government to face lawsuits from ISPs, banks, and foreign entities, all concerned that intercepting TLS traffic would undermine the security of all Internet communications originating from the country and jeopardize related commercial interests [56].

3.2.2 Core Architectural Components

This section introduces Huma’s main architectural components, which are illustrated as purple boxes in Figure 3.1.

Huma authority. The *Huma authority* (HA) is located outside the censored region and is considered to be free from adversarial influence. It embodies a trusted component of Huma’s infrastructure, and is the initial point of contact for: a) new users (e.g., Alice) interested in leveraging Huma to circumvent Internet censorship, and b) volunteer website operators hoping to contribute to Huma’s infrastructure (§3.3.1). Our design assumes that the HA leverages a robust proxy distribution service (e.g., Lox [227]), precluding an adversary from registering as multiple users and successfully enumerating the entirety of the Huma proxy infrastructure. To allow users’ registrations, the HA adopts mechanisms inspired by Tor’s bridge distribution system, offering different alternative means for users to contact it, such as e-mail or instant-messaging [221].

Secure proxies. Huma relies on two interconnected components that provide a secure proxy abstraction: a) volunteer-operated *Decoy Websites* (DWs), and b) trusted *Shade Proxies* (SPs) maintained by Huma operators. Decoupling proxy functionality among these components allows Huma to achieve resistance against Sybil proxies (see §3.3.3).

Similar to WebTunnel bridges [105], DWs appear as ordinary, benign websites, which serve as entry points into Huma. While providing legitimate content to typical web users, DWs inspect incoming traffic for steganographic tags embedded within HTTP payloads that signal that a message is intended for the Huma sub-system—in such cases, DWs explicitly interface with SPs for mediating users’ circumvention data. In practice, SPs are responsible for fetching the actual contents requested by Huma users (from their intended destinations), while shrouding this information from DWs, thus tackling content/destination fingerprinting concerns (§3.1.2).

Huma client. Huma users run a client assigned with two main tasks: a) encrypting and embedding a user’s circumvention requests into benign-looking encapsulating HTTP requests sent to SPs via DWs (§3.3.2), and b) retrieving the requested data from the DW backend via a *double-request receive* protocol (§3.3.2). When Huma is run by a user, her browsing patterns—towards DWs and the web in general—are ruled by an *overt user simulator* (OUS) (§3.3.2), which ensures that the user’s browsing patterns are consistent with her typical behavior. This OUS is informed by historical browsing data, helping Huma achieve *behavioral realism* (§3.1.1).

3.2.3 Overview of the Huma Workflow

This section outlines Huma’s operational workflow, whose individual steps (in yellow ) are also illustrated in Figure 3.1.

User registration and fetch of Huma credentials. To issue a registration request (§3.3.1), Alice first generates a key pair and queries the HA for obtaining a set of bootstrapping data for providing her access to the system (step ). Once the HA has validated Alice’s request, it will reply with the requested bootstrapping data, which includes a user identifier for accessing Huma-enabled websites, cryptographic materials required to participate in the bridge distribution protocol (e.g., anonymous Lox credentials if using Lox [227]), together with a list of DW addresses and key-exchange material that enable Huma proxy-based circumvention. After responding to Alice, the HA distributes Alice’s public key and her identifier to her assigned DWs and SP, while also sending her key-exchange material to the SP (step ). This ensures that DWs can validate Alice’s authorization to access the system, and that SPs can conceal the contents and destination of Alice’s covert communications from DWs themselves.

Placing requests towards Huma proxies. To issue a covert request for an otherwise blocked website’s content (e.g., `cnn.com`), Alice encrypts the request intended to be read by the SP using a symmetric key exchanged with the SP via the HA during registration

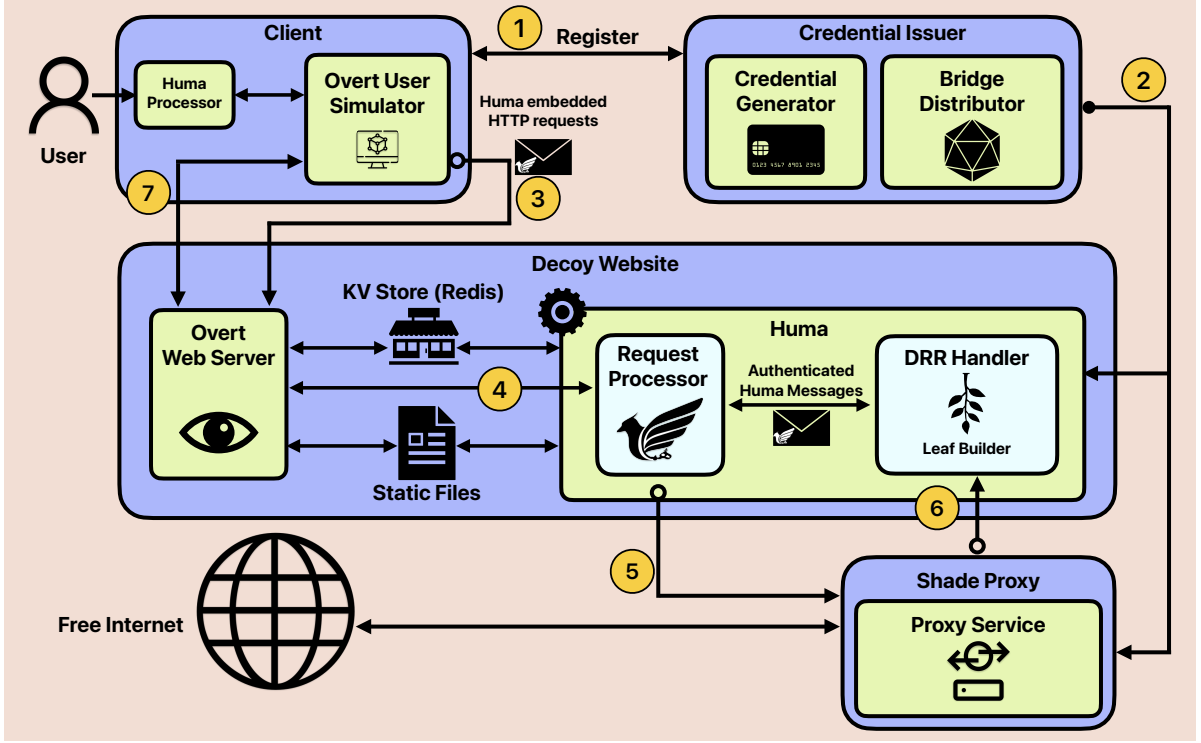


Figure 3.1: Overview of Huma’s architecture and high-level workflow.

(see §3.3.1). Alice then embeds this encrypted payload, along with her public key and the user identifier given by the HA, within the body of a benign-looking HTTPS request (see §3.3.2) directed to a DW (step 3).

To ensure that timing anomalies (which could reveal the covert channel [199]) are avoided when processing this special request, the DW immediately processes the encapsulating HTTP request, replying with a legitimate webpage to the client (see §3.3.2). However, after replying, the DW forwards the original HTTP payload to its local Huma backend, where the request is scanned for Huma-related content (step 4).

Once the Huma backend receives a request for analysis, it authenticates the user using her credentials (i.e., the public key and identifier facilitated by the user), terminating silently if authentication fails (§3.3.1). If the user’s credentials are correctly validated, the DW forwards the user’s request to the paired SP, where it is decrypted and processed (step 5). Throughout this process, a DW operator does not learn any details about the contents or destination of a user’s request—only whether a specific user can access Huma

(§3.3.3).

Once the SP processes the request and fetches the user’s desired content, it encrypts the response using the symmetric key shared between Alice and the SP, before padding and returning it to the DW (step 6). The DW must then allow the client to retrieve this response. To this end, the response is processed by a *double-request receive* (DRR) handler that prepares Alice’s data to be fetched (§3.3.2).

Retrieving responses from Huma proxies. The DRR handler embeds Alice’s encrypted response within a copy of a legitimate webpage served by the DW, following a data replacement scheme akin to that of Slitheen [41] or Balboa [199]. This modified page retains the size and format of the original page, appearing unchanged to an adversary observing the encrypted HTTPS traffic (§3.4.4. Once the modified page is prepared, the DRR handler signals Huma’s request processor to serve Alice’s upcoming request using the modified webpage. When Alice sends her next request, whether benign or Huma-embedded, the DW immediately replies with the prepared response, thus fulfilling the original request and finishing DRR (step 7). This immediate response is made possible through a fast key-value store that the DW always checks before responding to *any* legitimate or Huma request (§3.3.2).

3.3 Details on Huma’s Operations

This section discusses the innards of Huma’s operations tied to the management of its users and infrastructure components, providing details about how Huma achieves resistance against traffic analysis and Sybil proxy infrastructure. Lastly, we discuss how Huma can be adapted to Intranet scenarios.

3.3.1 Managing Users and the Infrastructure

The *Huma Authority* enables users to connect to Huma proxies and allows volunteer website operators interested in running DWs to contribute to the circumvention infrastructure.

User Registration and Credential Distribution

The bootstrapping process followed by a new user interested in leveraging Huma is illustrated in Figure 3.2. This process includes three main phases, responsible for a) issuing

credentials to users; b) provisioning DWs and SPs with necessary information to authenticate Huma users, and c) establishing a shared key among the user and the SP. We describe these phases below, before addressing how DWs verify Huma requests.

Credential issuance. To access Huma, a user must first generate a key pair (U_{Pub}, U_{Priv}) to enable her authentication before the system through the use of cryptographic signatures. The authentication scheme used in each region is selected by the Huma authority (HA) based on local network conditions. In low-bandwidth regions, compact signature schemes such as EdDSA [122] may be preferred, while in high-bandwidth regions, faster signature schemes such as RSA-based signatures [63] can be used to reduce the load on DWs.

After generating her key pair, Alice registers with the HA using her U_{Pub} . The HA responds with a unique user ID (U_{ID}), credentials for participating in the bridge distribution protocol, and a list of DW domain names—each corresponding to an entry point into the Huma that the user can now access.

Provisioning DWs and SPs with authentication material. To complete the registration workflow, the HA must ensure that both DWs and SPs are equipped to authenticate and authorize requests from the new user. For each DW in the user’s list, the HA shares the required data for validating the user’s credentials, which includes the user’s ID (U_{ID}) and her public key (U_{Pub}). The HA also shares this info with the SPs.

Key agreement between users and SP. During registration, the HA will also act as a secure intermediary between the user and a given SP, allowing them to run a key agreement protocol (such as Diffie-Hellman (DH)) to produce a symmetric key for their future communications, without requiring the user and SP to directly contact each other. At this step, the user generates a public and private DH share and then sends the public DH share to the HA. The HA forwards this public DH share to the SP, which will, in turn, generate its own public and private DH share, sending its public DH share back to the HA so that it can be forwarded to the user. After completing the DH key agreement, both the user and the SP now share a secret key that they can use for later encrypting/decrypting messages. This setup allows DWs to act as blind relays, preventing them from observing or modifying the encrypted data flowing between the user and SP (§3.3.3). The shared secret key can be ratcheted [187] upon each request to provide forward secrecy.

DW Registration and DW-to-SP Assignment

Operators who wish to volunteer and turn their websites into DWs must first register with the Huma authority (HA).

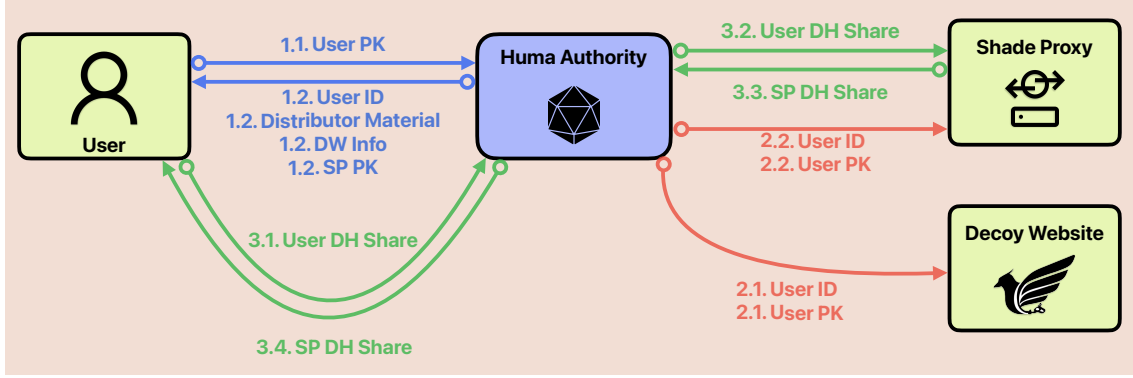


Figure 3.2: Bootstrapping data shared by the Huma authority during user registration. Each phase is color-coded: blue for issuing user credentials, red for provisioning servers with client information, and green for key agreement.

Website ownership verification. The registration process begins with the operator initiating contact and providing verifiable proof of domain ownership. To prevent malicious registration attempts, Huma requires that website operators provide evidence of control over the website they wish to enroll. A simple and effective mechanism is to request that the operator produce a signature over a challenge message using the private key associated with the website’s HTTPS certificate. This signature can then be verified by the HA using the public key embedded in the website’s certificate.

Centralized DW-to-SP assignment. Once ownership is verified, the HA assigns an SP to the newly registered DW. Important to Huma’s resilience against Sybil proxies, SPs are not chosen by operators. Instead, they are centrally managed by the HA, thus ensuring that each DW–SP pair operates independently. The aim is to prevent the possibility of collusion between DWs and SPs, which could lead to compromising confidentiality and destination concealment (§3.1.2).

Scalable SP deployment. Since we assume SPs to be trusted infrastructure components supported by the operator of the overarching Huma infrastructure, considerations must be made regarding the costs involved in running SPs. Fortunately, the costs of deploying SPs can be amortized by hosting them as lightweight proxies atop existing cloud infrastructure or embedded within systems like SpotProxy [139], which substantially reduce the costs of operating cloud proxies via cloud spot instances. In addition, a single SP can support users connecting through multiple DWs. Because SPs operate from network locations outside the censor’s control, compromising one DW may allow the censor to block its connection

to the SP, but it does not affect the ability of other DWs to reach the same SP.

3.3.2 Achieving Resistance against Traffic Analysis

This section discusses how Huma resists traffic analysis and fingerprinting attempts.

Huma Request Generation and Authentication

When placing an encrypted request R towards a SP (via a DW), Alice generates and includes a signature of this request $Sig_{U_{Priv}}(R)$ alongside her U_{ID} . This data is then embedded into a benign-looking request sent to a valid DW endpoint. Next, we detail how Huma users covertly embed encrypted request data and how DWs can authenticate the requests sent by users.

Request generation and embedding. Huma clients use prior information on benign exchanges between users and a DW to craft benign-looking, Huma-embedded requests. Similar to Nasr et al. [179], Huma clients periodically send and collect benign requests to DWs when no Huma requests are pending to sent out, recording relevant metadata such as request size and format. This data later serves as a template for embedding Huma content to help maintain consistency in request size and structure. For instance, a DW may offer HTTP POST endpoints for actions such as commenting, posting reviews, or uploading files. A Huma client that has recorded past interactions with these endpoints can select the most appropriate one for embedding a covert request based on the content size, adhering to benign activity patterns before the DW.

Requests and credential verification. After preparing and embedding her covert request, Alice will send it to the DW. Upon responding to Alice with a benign response as per the deferred reply protocol (see §3.3.2), Alice’s request is forwarded to the Huma backend, where the encapsulated information is used to verify her authorization to access the DW. To this end, the DW checks if the U_{ID} included as part of the user’s request had been communicated to it by the HA, and then verifies whether $Sig_{U_{Priv}}(R)$ can be validated using the U_{Pub} matching that U_{ID} . If any of the above checks fail due to missing (as in the case of typical web users) or invalid U_{ID} and $Sig_{U_{Priv}}(R)$ information (as in the case of probing attempts by censors), the DW will simply silently fail. This *prevents active probing* attacks from Huma-aware censors who may send Huma-compatible requests to websites at large, hoping to elicit website behavior leading to the identification of DWs.

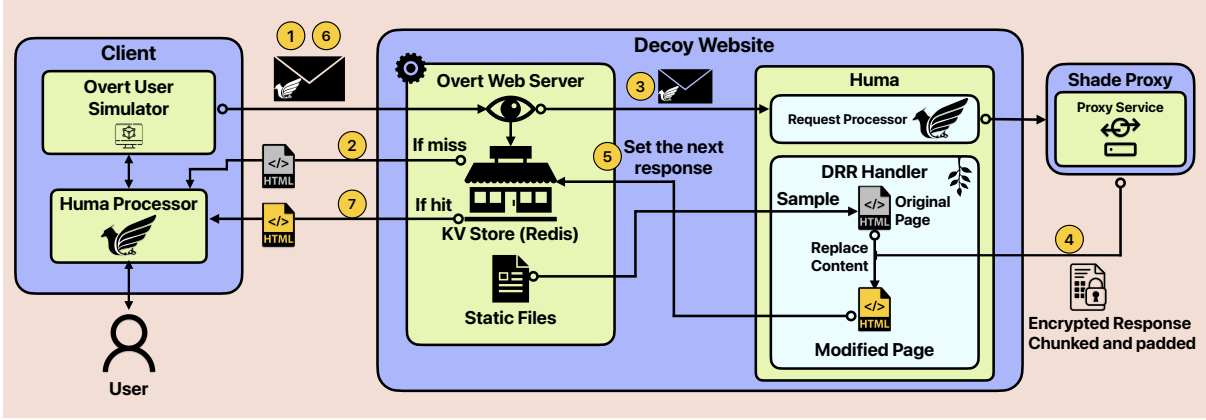


Figure 3.3: Deferred processing mechanism applied to double-request receive.

Secure Data Retrieval Via Deferred Replies

To resist timing-based fingerprinting, Huma adopts a *deferred reply* strategy where all DWs reply immediately to incoming requests, regardless of whether the request carries covert Huma traffic. Any additional processing—such as authenticating the user (§3.3.2) or querying the SP—is deferred to a backend that operates asynchronously. Unlike Balboa [199], this ensures that Huma-related requests are indistinguishable from legitimate traffic based on response timing alone, thus eliminating telltaling cues that a DW may be serving circumvention traffic.

Double-request receive. The deferred reply processing model lays the foundation for Huma’s double-request receive (DRR) protocol for data delivery, depicted in Figure 3.3. DRR starts when Alice requests a covert webpage through a DW (step ❶). The DW will always immediately respond to the user, per the deferred response processing protocol. To respond to users, the DW always queries a local key-value (KV) cache to check if the requesting user has a modified page in-store for them. Since Alice is just issuing a request, we assume that she has no modified page in-store for her and her cache will miss, thus immediately receiving a benign page upon request (step ❷).

After responding to Alice, her request is forwarded to the Huma backend for authentication (§3.3.2). Upon successful verification, her request will be forwarded to the SP for fulfillment (step ❸). Then, the SP attempts to decrypt Alice’s request using the agreed-upon symmetric key (§3.3.1), failing silently if decryption is unsuccessful or no key is found for the user. It then acts as a proxy, fetching the requested content, padding and encrypt-

ing the response, before splitting it into predetermined chunk sizes (§3.3.3). Finally, the encrypted chunks of data are submitted back to the DRR handler at the DW, where they are prepared for retrieval by the user (step 4).

The DW’s DRR handler prepares the response using the following steps. First, it samples a copy of a legitimate webpage served by the DW along with all its leaf nodes (e.g., media content and scripts). This page (which can be chosen at random, or informed via some minimum-size constraint) will act as a cover for delivering the actual response. If the combined size of the leaf nodes on this page is insufficient to hold the full response, Huma falls back on one of its multi-page delivery mechanisms (§3.3.2) to send the data over several pages. Otherwise, the DRR handler overwrites the encrypted response chunks into the leaf files, ensuring that the resulting objects preserve the original file sizes. This ensures that, from the perspective of a network eavesdropper, the TLS records generated by the website when conveying covert data remain indistinguishable from those of an ordinary webpage fetch. Each leaf node includes metadata tags indicating its order and content length, allowing the Huma client to reconstruct the response. Leaf nodes are written to disk on the DW, as if they were legitimate leaf contents. The HTML of the sampled page is then modified to reference the newly created leaf files, resulting in a complete response page. Finally, the path to this page is added to Alice’s queue in the KV store (step 5).

When Alice issues another request, either benign or Huma-embedded (step 6), the DW immediately attempts to respond to her request as per the deferred reply strategy. As previously described, the DW will first query the KV cache using a `uuid` key included in all users’ cookies (Huma or otherwise). This time, the cache query results in a hit, as covert content is available for Alice. Ultimately, the DW responds using the cached webpage (step 7) and removes it from Alice’s queue by popping it from the key-value store. If Alice’s request is Huma-embedded, it is forwarded as usual to the Huma backend (step 3) and re-initiates the deferred reply cycle.

Behavioral Realism via Overt User Simulation

To maintain behavioral realism, Huma ensures that all client-to-DW interactions resemble those of ordinary web users. This is achieved through an overt user simulator (OUS), which intercepts client-initiated communications and wraps them in realistic browsing activity. Without such a mechanism, clients could inadvertently issue an unusually high number of requests to a small set of DWs, representing anomalous behavior that censors can detect through statistical modeling [244]. Huma’s OUS emulates human browsing by orchestrating benign-looking requests that blend Huma traffic with legitimate HTTPS traffic. Its design

draws inspiration from prior work on heuristic-based modeling of human-like browsing (as in OUStral [158]), or the synthetic generation of plausible user behavior (as in Raven [244]).

Huma’s OUS training and deployment modes. Huma supports two deployment modes for its OUS: a) a *personalized* training mode, and b) a *pretrained* mode.

First, in the personalized mode, the user runs the OUS in a “learning phase” over several days, allowing it to record their natural browsing habits passively. This results in a customized behavior model that helps provide strong resistance against profiling-based attacks by reflecting characteristics such as realistic inter-request timing and website revisiting patterns.

Second, users may download a pretrained OUS instance during their registration process with the Huma authority. This model aims to reflect typical usage patterns of users within a given region or demographic, and could be used without a warm-up phase. While convenient and lowering the barrier of entry to new Huma users, it may present higher risks of steering away from a user’s typical browsing patterns. Huma delegates the choice of deployment mode to the user, allowing them to trade off between behavioral fidelity and ease of setup.

Huma OUS’ runtime behavior. The core of the OUS is a headless browser instance that simulates user behavior such as page visits, clicking internal and external links, and revisiting previously accessed content. While the client is active, the OUS schedules outgoing requests based on a behavior script derived from its model, issuing benign requests to DWs even when no covert data is pending. These benign requests can include a variety of behaviors, such as refreshing previously visited pages or accessing randomized third-party resources to accurately reflect the modeled browsing profile.

When Huma users wish to place requests (or retrieve covert data from requests placed in the past), the OUS draws from a queue of pending Huma requests and embeds them into realistic browsing sessions. Requests are delayed and issued only when a behavior rule allows, thereby preventing anomalous browsing patterns that might reveal the presence of hidden communication. The integration of the OUS with Huma’s deferred content replacement and request scheduling ensures that operations tied to cross-site multi-page responses (§3.3.2) are hidden within a flow of plausible user behavior.

Multi-page Fallback Mechanisms

Ideally, the covert response to a user’s data request fits entirely within a single webpage’s leaf objects, allowing for straightforward delivery. However, in cases where the response exceeds this capacity, Huma falls back on two more elaborate strategies.

Same-site multi-page fallback. If the size of the leaf objects of the originally sampled page is insufficient to carry the entire SP response, Huma invokes a fallback mechanism that allows the DW to split the user’s response across multiple requests by selecting additional webpages from the same DW and performing the same substitution process. The user’s Huma client software is aware of the mapping between requests and response fragments, allowing it to reassemble the complete response after fetching all necessary pages. Each individual page load still appears benign, while collectively enabling the covert delivery of larger response payloads.

Cross-site multi-page fallback. To reflect plausible browsing behavior and avoid raising suspicion through repeated accesses to the same DW, Huma’s overt user simulator (§3.3.2) will direct users to visit other seemingly unrelated, legitimate websites. However, this would prevent a user relying on *same-site multi-page fallback* from gathering her intended response by repeatedly querying the DW she is currently connected to—at least, until the OUS instructs the user to revisit the website.

To tackle this issue, Huma supports an improved fallback mechanism that ensures service continuity across multiple DWs. It does so by allowing users to retrieve different segments of a response by visiting multiple DWs, which are all linked to the same SP. This process calls for two requirements: a) the SP stores the entirety of the response to a user’s request locally for a predetermined period, and b) the client includes a progress counter that keeps track of how many data segments it has received in each DRR request placed towards a DW. If a user switches to a different DW before fetching the entirety of a response, the new DW can query the SP for only the remaining segments the user has yet to receive. The combination of these mechanisms allows clients to switch between DWs, possibly accessing them in any order, while still correctly reassembling the full response. Altogether, this cross-site fallback mechanism can improve both covertness and reliability: it is compatible with behavioral realism enforcement by distributing access patterns across different DWs while being robust against the potential temporary downtime (or even blocking) of any individual DW.

As described above, this fallback mechanism requires keeping state at the SPs. To avoid starving SPs’ storage resources, SPs keep responses to user requests for a short tunable period (e.g., 30min), deleting them either when the user fetches the response completely or the above period elapses. In the former case, DWs are expected to cooperate with the SP, signaling whether a user has finished fetching a pending response.

3.3.3 Achieving Resistance against Sybil Proxies

To defend against traffic analysis and protect user privacy, Huma separates duties between DWs and SPs. Untrusted DWs relay encrypted messages and serve data, while trusted SPs decrypt requests and contact covert destinations. This prevents any single entity from linking a user’s identity with their communications. Below, we explain this design point.

Encrypted communication and write obfuscation. After a SP fetches a response for a user’s request (§3.3.2), it pads and splits the data into fixed-size data chunks before encrypting the result and submitting that back to the DW. This mechanism provides an important layer of protection against malicious DWs. Each data chunk is padded to a uniform size, complicating content-based fingerprinting. The chunk size is controlled by a configurable parameter that balances resistance to fingerprinting against network overhead; we evaluate various settings in §3.4.3. As a result, a DW only learns the number of padded chunks (and the sum of their size) retrieved by the user; our evaluation shows this knowledge can be made rather uninformative for a fingerprinting adversary.

Curtailing the attack surface with compromised DWs. Even if a DW is compromised by a censor, the censor gains no visibility into what information is being exchanged or which covert destinations are being accessed by users. Although the SP can decrypt user requests, it possesses access only to a user’s randomly assigned U_{ID} , but no network-layer information about the user for whom it is resolving requests. Even if a rogue DW deliberately sends such information to the SP, SPs are considered trusted elements of the Huma infrastructure and are configured to simply ignore supplemental data conveyed to them, other than a user’s U_{ID} and the network destination to access and fetch content from.

3.3.4 Deployment within Intranets

In Intranet scenarios, censors possess complete visibility over all network links established within the censored region.

Re-purposing Huma for Intranet scenarios. Huma can be repurposed as a covert messaging platform for Intranet scenarios, providing a secure channel for users to exchange sensitive information. Unlike traditional anonymous messengers (such as X [60] and Y [81]), Huma ensures both user privacy and *blocking resistance*, protecting users not only from potentially malicious messaging servers but also from adversaries attempting to detect or disrupt covert communication altogether.

The primary architectural change in Huma’s Intranet mode is the removal of SPs and the integration of private information retrieval (PIR)-enabled databases within DWs to serve as user mailboxes. This prevents DWs from learning communication patterns or metadata about conversation participants. We adopt Pung [24] as our PIR database due to its fitting key-value store design, which aligns with Huma’s existing components.

Modified client exchanges in Intranets. In this setup, users write encrypted messages to the DW using a process similar to Huma’s original scheme, with minimal modifications. In Pung, users are assumed to possess each other’s public keys, a requirement fulfilled in Huma via the HA. Following the Pung protocol, users can derive shared secrets K_L and K_E through a key derivation scheme [140], used for mailbox label generation and message encryption, respectively.

In the Intranet scenario, each Huma user request includes the encrypted message $R = Enc_{K_E}(M)$, the sender’s U_{ID} , the signature $Sig_{U_{Priv}}(R)$, a label for sending the message $label_S(r)$ and a PIR query $Q = Query(label_R(r))$ to retrieve incoming messages. Mailbox labels are deterministically generated using K_L and the current round number r , following Pung’s algorithm [24]. Private query Q is generated as part of the PIR scheme, which enables private message retrieval without revealing what content was accessed. As a result, each Huma request performs both a write and a read simultaneously, in accordance with Huma’s deferred processing (§3.3.2).

After the DW immediately responds to Alice, it will send the Pung request to the Huma backend for processing. There, Alice’s message R is written to Pung’s private key-value store, while her PIR query Q is executed in Pung to retrieve messages from her mailbox. The resulting response is then sent to the DRR handler to be prepared for retrieval. With each subsequent request, Alice receives the response to her previous query while simultaneously submitting new read and write operations. If a user’s OUS (§3.3.2) sends a request to a DW when their outgoing message queue is empty, the Huma client generates random messages addressed to random mailboxes as chaff, ensuring the privacy guarantees of Pung [24].

3.4 Evaluation

We start by detailing our evaluation goals (§3.4.1) and experimental testbed (§3.4.2). Then, we assess Huma’s resistance against content fingerprinting (§3.4.3) and unobservability (§3.4.4), followed by an assessment of its throughput both with and without an OUS that enforces behavioral realism (§3.4.5).

3.4.1 Evaluation Goals

Our evaluation of Huma is guided by the assessment of its capabilities across three main dimensions.

Resistance to content fingerprinting. We wish to assess whether the data chunks relayed by DWs convey useful information about the browsing habits of Huma users. To this end, we devise an experiment that exposes DWs to a pool of content of interest (e.g., payloads requested by Huma users) transmitted by SPs under different chunk sizes. A classifier trained to distinguish different webpages based on the number of received chunks should then be unable to correctly pinpoint the website fetched by a given Huma user. We leverage classification *accuracy* as the metric for this experiment.

Resistance to traffic fingerprinting. We wish to assess Huma’s resistance against traffic fingerprinting. To this effect, the network fingerprint generated when fetching Huma-relayed content from a website acting as a DW should match the fingerprint generated by a benign TLS connection towards the same website. This includes both volumetric characteristics about the connection (e.g., number of transmitted packets, total bytes exchanged), but also timing characteristics (e.g., verifying the absence of noticeable delays introduced by Huma when serving content hosted by the DW). In this setting, a traffic classifier trained to distinguish benign data fetches from Huma data fetches should achieve a low *accuracy*.

Network performance. Lastly, we wish to assess the network *bandwidth* and *latency* overheads imposed by Huma when fetching webpages via its proxy-based system. Latency overhead is defined as the additional time required to load a page, while bandwidth overhead refers to the amount of extra bytes required to download a page via Huma. We also wish to assess the rate at which users can access content via Huma when deploying Raven- and OUStral-based OUSes.

3.4.2 Experimental Testbed

Here, we detail our testbed. We describe our laboratory environment, followed by a description of a corpus of websites we admit Huma users to be interested in fetching across our experiments. Then, it describes a dataset of realistic user browsing behavior which we use to feed our Raven-style OUS.

Huma prototype. We implement our prototype [1] for Huma clients, DWs, and SPs using 3 600 lines of Python code. We develop the Huma DW using the Django-Rest framework [183], which uses a local instance of Redis [159] as its KV store. We chose Redis

due to its performance and support for atomic queue operations. We created the SP as a light Flask [185] web app. Lastly, we built the Huma client using Python code, coupled with an OUS which, similarly to Wails et al. [244], uses the SDV library [118] for synthetic user behavior generation. We also use PyCryptodome [194] to perform our cryptographic computations for all components.

Hardware and network deployment scenario. We deployed three clients, one DW, one SP, and one webserver hosting a static website using VMs on DigitalOcean, each provisioned with 4GB RAM and 2 vCPUs running Ubuntu 24.04. The DW and SP are co-located in San Francisco (U.S.), while the clients are geographically distributed across San Francisco (U.S), Frankfurt (Germany), and Bangalore (India). The webserver, which acts as the target of user’s Huma requests, is placed in Toronto (Canada). This setup allows for comparisons that take advantage of the clients’ locations to analyze the impact of WAN network conditions on Huma’s performance, while isolating them from DW-SP interactions. By co-locating the DW and SP, we focus on assessing the effects of client-to-infrastructure distance, rather than intra-infrastructure coordination, as Huma’s HA can assign users with DW-SP pairings that intentionally decrease the latency experienced by users.

DW and target destination page sizes. Evaluating Huma’s network performance entails understanding the trade-offs between the page sizes of DWs and those of the websites of interest to users (referred to as *target websites*), as well as the size of the data chunks exchanged between users and SPs (via a given DW). To reduce the computational overhead of evaluating all possible combinations of DW and target destinations’ content sizes within an arbitrary range, we grouped page sizes into three tiers. Specifically, we measured frontpage resource sizes for the top 100 Tranco websites [190] (which served a frontpage via HTTP, as of March 2025), and used the 25th, 50th, and 75th percentiles of the resulting distribution to define *small*, *medium*, and *large* page sizes, as 1.3MB, 3.2MB, and 9.2MB, respectively. We select DW and target page sizes from these tiers, allowing us to explore multiple combinations of parameters during our experiments. For instance, we evaluate the impacts involved in having a DW with a large frontpage while covertly downloading a small target website,

Web browsing behavior dataset. To drive our Raven-style OUS, we rely on the real-world dataset of web browsing behavior provided by Kulshrestha et al. [143], collected from 2148 German users over the course of October 2018. The dataset captures 9.1 million URL visits across nearly 50 000 unique domains and includes, for each user, the anonymized URL and domain of each visited page. Using this dataset, Kulshrestha et al. [144] found that individuals’ Web routines are rather repetitive, with variation across users partly explained

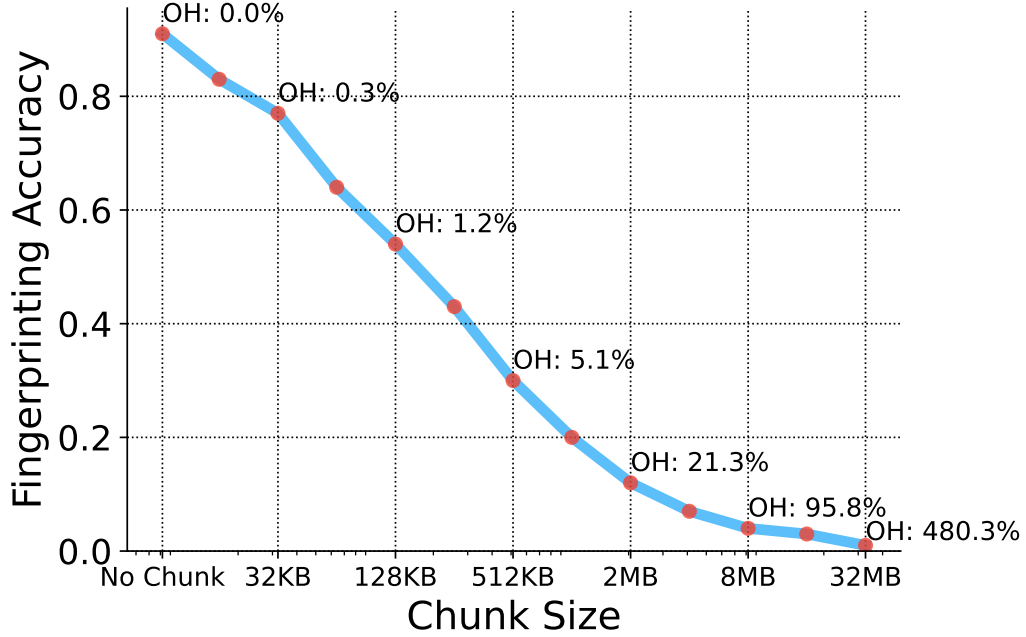


Figure 3.4: Trade-off on content fingerprinting accuracy and bandwidth overhead (Tranco top-100) when storing responses in differently-sized data chunks.

by their demographic and behavioral traits, supporting the dataset’s realism as a model of human browsing behavior.

OUStral parameterization. To compare the rate at which users fetch covert contents when using different OUSes, we also experimented with the OUStral browsing simulator, following its original system parameters [158] modeled after past studies of how users browse the web (see §3.1.1).

3.4.3 Resistance to Content Fingerprinting

In our first experiment, we aimed to understand the impact of our configuration of data chunk sizes (written by the SP on the DW, and then later fetched by Huma users) on the ability of an adversary controlling a malicious DW to fingerprint the contents retrieved by users. To this end, we trained an XGBoost [51] classifier for identifying the correct website fetched by a client, when having the SP split (and potentially padding) a target website’s data according to chunk sizes in the range of 64KB to 16MB. We conducted this experiment

by having the adversary attempt to identify the original website pages contained in our Tranco top-100 list, by leveraging the total combined size of chunks used to transmit a page as the only feature discernible by an adversary (e.g., for a target website of size 1.5MB and a chunk size of 1MB, the SP would write two 1MB blocks for a total size of 2MB). Besides content fingerprinting protection, one must also consider the overhead involved in defining the chunk sizes; while larger chunk sizes may provide better protection, they are expected to increase the network bandwidth consumed by Huma clients.

The results of our experiment are illustrated in Figure 3.4. Without any chunking, the classifier can identify websites with 91% accuracy. However, with chunking, we observe the general trend that larger data chunks provide better protection against fingerprinting. For instance, larger blocks (e.g., 16MB) approximate the classifier’s accuracy to random guessing, while even relatively small blocks offered some degree of protection (e.g., small chunks of size 64 KB decrease the classifier’s accuracy to 64%). From the figure, we can gather a sweet spot between content fingerprinting protection and modest bandwidth overhead. For instance, chunks of size 2MB decrease the classifier’s accuracy to 12%, while adding a limited bandwidth overhead of 21.3%. For this reason, we use this chunk size (2MB) for the remainder of our experiments.

3.4.4 Resistance to Traffic Fingerprinting

Our next set of experiments aims to verify whether Huma’s leaf content replacement, in conjunction with deferred reply and DRR functionality, can successfully prevent traffic analysis attacks targeting the activity of Huma users among those of legitimate users visiting a given DW website.

In this experiment, we configured our DW with a *medium* page size, and configured a target website of *small* size in our webserver, such that the entirety of the target website can fit within a single covert page fetched by the Huma user. This suffices for experimenting with the assessment of Huma’s unobservability since, when the target website page is larger than the DW one, the user will simply issue additional page fetches from the DW (which all undergo the same leaf content replacement for each individual page fetch).

We built three datasets of 200 TLS network traces each, referring to each of the three clients placed in different network locations. For each client, we collected 100 traces where it behaves as a legitimate user fetching the DW’s frontpage, and 100 traces where it runs Huma and fetches the same frontpage but with its leaf contents replaced by data from the target website. We evaluate the similarity between these two kinds of traces using two different methods.

Table 3.2: Classifier accuracy and KS similarity across client locations

Client Location	XGBoost Acc.	KS Statistic
San Francisco (U.S.)	$53 \pm 5\%$	D=0.003, p-value=0.98
Frankfurt (Germany)	$52 \pm 1\%$	D=0.06, p-value=0.47
Bangalore (India)	$54 \pm 4\%$	D=0.05, p-value=0.76

First, we leverage the machine learning-based covert channel detection classifier of Barradas et al. [33]. This classifier is fueled by XGBoost and contains over 150 manually-engineered traffic features comprising summary statistics that characterize different dimensions of network traces, including fine-grained information about packet sizes and inter-arrival timings, as well as statistics about a trace’s communication volume. In addition, we extended the feature set of the classifier to consider the timing information of page loads (by adding features such as total page loading time and percentiles of page load time), which are required to evaluate Huma’s robustness to attacks based on page loading timing anomalies that afflicted previous web protocol tunneling systems.

Second, we use the two-sided Kolmogorov-Smirnov (KS) test [165] applied explicitly to absolute page load timing distributions, as used before by Nasr et al. [179] to screen for timing anomalies introduced by leaf content replacement primitives used in refraction networking systems.

The results of our experiments are showcased in Table 3.2. We can observe that the XGBoost classifier can only distinguish between Huma flows and benign flows with an accuracy of at most 54%, approximating random guessing. The results of the KS test further suggest that an adversary cannot feasibly distinguish between the kinds of flows as their distributions are tightly overlapped. For instance, the KS test conducted over the TLS traces generated by the client located in San Francisco yielded a D-value of 0.03 with a P-value of 0.98; for reference, the indistinguishability results of Waterfall of Liberty [179] obtained only a D-value of 0.11 under a less strict P-value of 0.5. These results suggest that both the ML-based classifier and the KS test fail in distinguishing between regular and Huma connections with significant confidence.

3.4.5 Network Performance

Our last set of experiments is geared towards gauging the network performance of Huma. As part of this evaluation, we consider three different scenarios. First, we are interested in uncovering the *raw* bandwidth and latency overhead introduced by Huma when fetching a

Table 3.3: Huma’s latency and bandwidth overheads for the India client with different DW and target website page size combinations.

DW Size	Target Size	Fetch Count	Latency OH	BW OH Bytes
Small	Small	2	214% $\pm$ 30	100%
Small	Medium	4	385% $\pm$ 25	62%
Small	Large	8	712% $\pm$ 46	13%
Medium	Small	1	105% $\pm$ 37	146%
Medium	Medium	2	198% $\pm$ 30	100%
Medium	Large	4	396% $\pm$ 28	39%
Large	Small	1	139% $\pm$ 30	607%
Large	Medium	1	114% $\pm$ 33	187%
Large	Large	2	248% $\pm$ 41	100%

target webpage, assuming no OUS is deployed. While providing no behavioral realism, this setting allows for a better understanding of the overhead imposed by Huma vs. directly fetching a target website. Then, we assess the rate at which users can access content via Huma when deploying the Raven- and OUStral-based OUSes.

Second, we assess the latency overhead introduced by Huma when paired with the Raven-style OUS, using different user profiles sourced from Kulshrestha et al. [144]’s dataset. Third, we assess the latency overhead imposed by Huma when paired with the OUStral-based overt user simulator.

We guide our exposition using results from the client in India, which is not only hosted on a different continent from where the censorship circumvention infrastructure is situated, but also in a country known to experience Internet censorship. Results from clients in San Francisco (co-located) and Germany (also remote but with better connectivity) are included in §3.4.5, and show similar trends.

Raw overheads. Table 3.3 shows Huma’s latency and bandwidth overheads for the India client, when assessing different combinations of DW and target website page sizes, covering the *small*, *medium*, and *large* size tiers (§3.4.2). Each number in the table is obtained as the avg. of 100 fetches of the webpage in each configuration. We include std. dev. for latency overheads, while bandwidth overheads are fixed since response sizes do not change for our statically-sized target pages.

In general, reducing the number of requests required to download a page also reduces

the overhead introduced by Huma. For a *small* DW page and a *small* target, a Huma client must place two requests to fetch a complete response, resulting in a latency overhead of 214%. However, when the setting is changed to a *medium* sized DW, which can satisfy the request within a single fetch, the latency overhead substantially drops to 105%, while the bandwidth overhead increases from 100% to 146%. However, once the DW size is large enough to fulfill the request with a single fetch (e.g., *large* DW for *small* target page), further increasing the page size increases bandwidth overhead and provides no additional latency savings.

Additional Throughput and Overhead Results

We now dive deeper into the results obtained in Huma’s latency testing, discussing insights gathered from experiments conducted over the Huma clients located in the U.S. and Germany. We begin by discussing the U.S. client, before expanding on why the German client experiences more significant slowdowns despite having a higher bandwidth capacity than that available in the India-based client discussed in §3.4.5.

How geographical proximity affects overhead. In the U.S. client, geographical proximity to the target and high bandwidth capacity enable a lower latency overhead compared to other clients, as shown in Table 3.4. For instance, in the setting involving a *small* DW page and a *large* target website, requiring multiple DW requests, Huma’s latency overhead is 174% in the U.S. client, compared to 712% in India and 1239% in Germany, while the bandwidth overhead is similar in all examples. This observation highlights how the latency overhead caused by performing several fetches can rise when clients are geographically distant from DWs.

The effect of rising latency overheads due to several fetches can also be seen in Table 3.5, where the latency overhead of using a *small* DW to fetch a *small* target website is measured at 425% due to requiring two fetches to complete the request, while introducing a bandwidth overhead of only 100%. However, in the case of using a *medium* DW to fetch the same target website, even though the bandwidth overhead has increased to 146%, due to requiring only one request to be completely fetched, Huma introduces a latency overhead of only 247%, nearly half of the first scenario. We can verify our findings from the India client (§3.4.5) by analyzing the results from the Germany and U.S. clients. In the Germany client, the latency overhead reduction achieved by increasing the DW size to reduce the number of requests (similar to the cases discussed above) remains significant, though smaller than that observed on the India client. For example, on the India client, the latency overhead drops from 214% down to 105% when changing the setting from a

Table 3.4: Huma’s latency and bandwidth overheads for the US client with different DW/target website page size combinations.

DW Size	Target Size	Fetch Count	Latency OH	BW OH Bytes
Small	Small	2	66% ^{±86}	100%
Small	Medium	4	104% ^{±66}	62%
Small	Large	8	175% ^{±69}	13%
Medium	Small	1	59% ^{±94}	146%
Medium	Medium	2	95% ^{±61}	100%
Medium	Large	4	172% ^{±60}	39%
Large	Small	1	64% ^{±53}	607%
Large	Medium	1	98% ^{±81}	187%
Large	Large	2	155% ^{±43}	100%

small DW page and a *small* target size, to the same target size and a *medium* DW page, a reduction of $\sim 51\%$. In contrast, the Germany client sees a drop from 425% to 247% under similar changes, which is still substantial but is a smaller reduction of $\sim 42\%$. This trend is further confirmed by the U.S. client, where the latency overhead decreases by only $\sim 10\%$ under the same change.

Examining the effects of bandwidth capacity on latency overhead. Comparing overhead results between the India and Germany clients, we generally observe a higher latency overhead in the Germany client compared to the India client. This increased latency overhead in Germany counterintuitively stems from its higher bandwidth capacity. While increased bandwidth capacity improves direct page load times without using Huma, the fixed request latency impacting every Huma request does not decrease with additional bandwidth capacity. As a result, the latency overhead percentage becomes larger in distant servers with good bandwidth capacity, such as the Germany server, despite having better raw page load times than the India server.

Providing better service to Huma users. The Huma authority can leverage this observation by assigning users located farther from available DWs to DWs with larger page sizes, capable of satisfying their requests in fewer pages than smaller pages. Similarly, users residing closer to where DWs are hosted can be matched with DWs that have smaller page sizes.

Table 3.5: Huma’s latency and bandwidth overheads for the German client with different DW/target website size combinations.

DW Size	Target Size	Fetch Count	Latency OH	BW OH Bytes
Small	Small	2	425% $\pm$ 74	100%
Small	Medium	4	670% $\pm$ 68	62%
Small	Large	8	1239% $\pm$ 92	13%
Medium	Small	1	247% $\pm$ 61	146%
Medium	Medium	2	394% $\pm$ 81	100%
Medium	Large	4	746% $\pm$ 72	39%
Large	Small	1	305% $\pm$ 88	607%
Large	Medium	1	264% $\pm$ 64	187%
Large	Large	2	494% $\pm$ 74	100%

3.4.6 Network Performance w/ OUS

Daily page fetches w/ Raven-style OUS. For setting up this experiment, we sorted the users included in Kulshrestha et al. [144]’s dataset along three usage profiles, based on their average number of page visits per day. We selected the 25th, 50th, and 75th percentiles as representatives of lightly to very active users (see Appendix 3.4.6 for profiling details). Further, we assume that a) all Huma users’ visits to DWs are intended to fetch covert data, and b) that DWs’ frontpage sizes follow a distribution of 25% small, 50% medium, and 25% large DWs.

Figure 3.5(a) shows the number of pages of varying sizes that can be fully fetched (per day) as a function of available DWs when Huma is used with an OUS based on different user models. The figure reveals that the number of small pages that lightly active and medium active users can fetch eventually plateaus. This occurs when the number of available DWs exceeds the number of unique page visits they can make per day. Another notable observation is that the medium active user performs better than others when fewer DWs are available. This is due to their browsing pattern, visiting fewer websites but doing so with higher frequency. Given the overhead of accessing a new DW, caused by sending the first request of the DRR process (see §3.3.2), users with more visits per unique website can utilize each DW more efficiently to complete more page retrievals by using fewer requests. After issuing the first fetch request to a website, Huma clients can use the second request to ask for another endpoint, thereby chaining fetch requests. This technique allows users

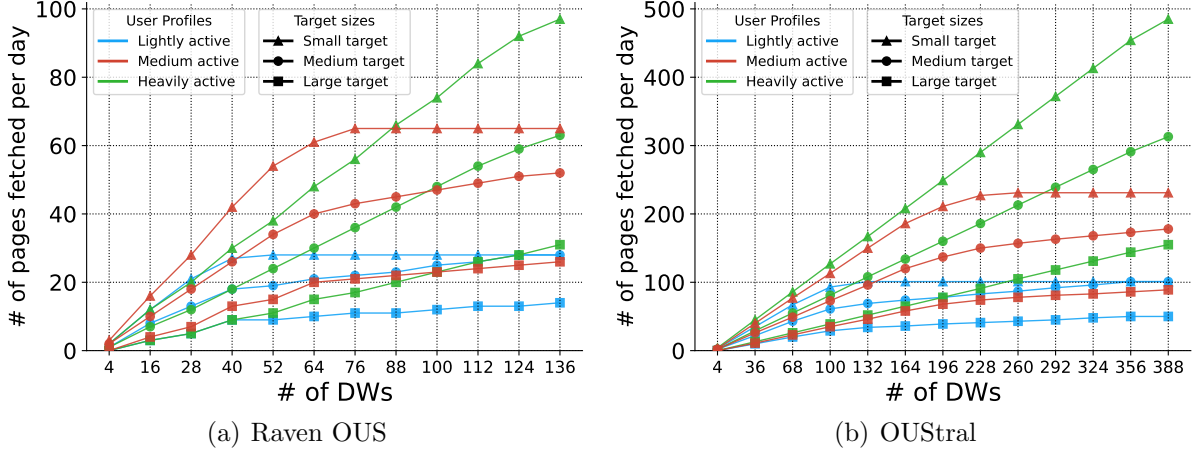


Figure 3.5: Pages fetched via Raven and OUStral-based OUSes, with varying page sizes, as a function of available DWs across different user profiles.

to maximize the utility of a single DW. However, users with lower average visits per day eventually reach their maximum throughput. In contrast, heavily active users, who visit enough unique websites per day that can match the number of available DWs, can use their heavy usage to their advantage to load higher volumes of content per day as more DWs become available.

Daily page fetches w/ OUStral-style OUS. Lorimer et al. [158] introduce a pluggable OUS that supports automated browsing while a user is active. To simulate OUStral usage, we resorted to the same three user profiles used in our Raven-style OUS test, and computed their average active browsing time (see Table 3.6). Using these values, we simulated 100 days of OUStral activity and recorded total page visits and unique page visits for each profile. We then treated this synthetic data as genuine browsing behavior and executed a similar experiment to the Raven-style OUS.

Figure 3.5(b) shows the number of pages of varying sizes fetched per day as a function of available DWs. Compared to the Raven-style results, OUStral consistently yields a higher fetch count, primarily due to its more active browsing behavior (see Appendix 3.4.6–Table 3.7). We can observe that an OUStral-based OUS following a *medium activity* profile fetches 73 medium-sized targets when 100 DWs are available, while a Raven-based OUS is able to fetch only 47 in the same scenario. The range of tested DWs is also expanded, reflecting OUStral’s more active browsing sessions, which allows it to access more DWs per day, thus reaching higher page fetch counts. While shown to evade bot detection algorithms [158], OUStral’s activity patterns may no longer follow current user browsing

behavior. As a result, we posit that this OUS is better suited for moderate-risk scenarios, offering a higher number of page fetches per DW, compared to Raven.

OUS User profiles

To test the practicality of our OUS, we chose the 25th, 50th, and 75th percentiles of the users based on the average number of pages they visited per day, derived from the data provided by Kulshrestha et al. [144]. The data on these chosen users can be seen in Table 3.6. In this table, we can see that the medium active user, despite having fewer page visits per day, has a higher average visits per unique page per day (~ 2.22) than the heavily active user (~ 1.86), which allows them fetch more pages from each DW on average due to a better utilization of each DW. As a result, the medium active user has a higher fetch count when fewer DWs were available, as evident by Figure 3.5(a). However, this higher fetch count is short-lived, as increasing the number of DWs past a user’s unique visits per day begins to degrade their fetch count until it plateaus.

We then used the active time values from Table 3.6 to simulate OUStral and calculate the number of page visits it does within the same amount of active time. We recorded this data over 100 tests, and reported the average in Table 3.7, which we used to evaluate OUStral’s throughput and potential number of page fetches per day.

Comparing browsing behavior in OUStral and Raven. By comparing the values in Table 3.7 with Table 3.6, we can observe the behavioral differences of these two systems. Raven’s OUS tries to mimic real browsing data, and visits fewer pages per day, while OUStral is based on preset probabilities, which can lead to much more active browsing patterns. Although Lorimer et al. [158] verified that OUStral can evade bot detection, it is unknown whether OUStral can successfully evade a censor who can deploy machine learning models to detect its operation (in contrast to Raven-generated synthetic browsing profiles). As a result, we believe the Raven-based OUSes to be safer than OUStral in

Table 3.6: OUS user profiles, and their average statistics per day, extracted from the data provided by Kulshrestha et al. [144].

User Profile	ID	Visits	Unique Visits	Active Time
Lightly active	1194	58.12	30.50	1199.50s
Medium active	1392	118.60	53.50	2666.93s
Heavily active	1462	233.11	125.07	6029.61s

Table 3.7: OUStral user profiles w/ average stats per day, created by simulating OUStral using the activity times from Table 3.6.

User Profile	Visits	Unique Visits	Active Time
Lightly active	188.29	87.0	1199.50s
Medium active	407.33	176.0	2666.93s
Heavily active	919.23	373.0	6029.61s

high-risk scenarios, where the censor might be able to perform behavior matching between user profiles.

Chapter 4

Conclusion

We introduced Anix and Huma, two novel tools each contributing to user freedom during internet restrictions in different Scenarios. Anix, a novel blackout-resistant messaging app, allows users to conduct secure and anonymous microblogging-alike communication during internet shutdowns. Huma, a new censorship-resistant communication tool based on a Web deployment environment, allows users to use it as an intermediary-based censorship circumvention tool that can function as an undetectable anonymous messaging system during Intranets.

4.1 Limitations and Future Work

Anix. Anix’s hard revocations offer post-compromise anonymity but require swift action from the affected users so that they can issue revocation certificates and resist impersonation attempts. This may be difficult in practice, e.g., if the user whose device was seized is temporarily detained for interrogation. An interesting direction for future work would include the exploration of anti-forensic mechanisms for automating revocations without user involvement, e.g., by automatically issuing revocation certificates from a secondary device if the user does not periodically cancel a “self-destruct” countdown timer on it.

China is the first country to crack down on mobile ad-hoc file-sharing apps [180, 274], targeting commercial solutions like Apple’s AirDrop, and Bluetooth-based apps. Possible upcoming restrictions envision banning the sale of commercial apps, not affecting apps like Bridgefy, Briar, or Anix. Currently, ad-hoc Bluetooth communications are not restricted,

but future regulations might prompt the development of enhanced mesh protocols that hide the fact that users are using blackout-resistant messaging apps altogether.

Other research directions aligned with our work may also include the study of alternative schemes for the selective linking of pseudonyms and pseudonym revocation, e.g., by adapting recent solutions for other contexts such as signing code commits [168] and VANET scenarios [59], respectively.

Huma. As it stands, Huma’s performance is the most crucial area to improve. Future work on Huma should focus on better utilization of decoy website page storage to increase Huma’s performance even further.

References

- [1] Anix source code and simulator. <https://www.github.com/kamali-sina/anix>.
- [2] The great firewall of china.
- [3] Iran Shuts Down Google, Will Completely Cut Citizens Off the Internet, 2012.
- [4] Community Connectivity: Building the Internet from Scratch - Annual Report of the UN IGF Dynamic Coalition on Community Connectivity, 2016.
- [5] 2019 Internet blackout in Iran, 2023. Page Version ID: 1176557895.
- [6] Egypt Shuts Down Internet Cellphone Services - WSJ. <https://www.wsj.com/articles/SB10001424052748703956604576110453371369740>, 2023. Accessed: 25 November 2023.
- [7] Snowflake, a censorship circumvention system using temporary WebRTC proxies. <https://www.bamssoftware.com/papers/snowflake/snowflake.20231003.e6e1c30d.pdf>, 2023.
- [8] Bramble transport protocol, version 4. <https://code.briarproject.org/briar/briar-spec/-/blob/master/protocols/BTP.md>, 2024.
- [9] Briar. <https://briarproject.org/>, 2024.
- [10] Bridgefy. <https://bridgefy.me/>, 2024.
- [11] Reddit - dive into anything. <https://www.reddit.com/>, 2024.
- [12] Signal. <https://signal.org>, 2024.
- [13] X. happening now. <https://x.com/>, 2024.

- [14] Ittai Abraham, Benny Pinkas, and Avishay Yanai. Blinder—scalable, robust anonymous committed broadcast. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1233–1252, 2020.
- [15] Martin R. Albrecht, Jorge Blasco, Rikke Bjerg Jensen, and Lenka Mareková. Collective information security in Large-Scale urban protests: the case of hong kong. In *Proceedings of the 30th USENIX Security Symposium*, pages 3363–3380, 2021.
- [16] Martin R Albrecht, Jorge Blasco, Rikke Bjerg Jensen, and Lenka Mareková. Mesh messaging in large-scale protests: Breaking bridgefy. In *Topics in Cryptology – CT-RSA 2021*, pages 375–398. Springer, 2021.
- [17] Martin R. Albrecht, Raphael Eikenberg, and Kenneth G. Paterson. Breaking bridgefy, again: Adopting libsignal is not enough. In *Proceedings of the 31st USENIX Security Symposium*, pages 269–286, 2022.
- [18] Nikolaos Alexopoulos, Aggelos Kiayias, Riivo Talviste, and Thomas Zacharias. MCMix: Anonymous Messaging via Secure Multiparty Computation. In *Proceedings of the 26th USENIX Security Symposium*, pages 1217–1234, 2017.
- [19] Alice, Bob, Carol, Jan Beznazwy, and Amir Houmansadr. How china detects and blocks shadowsocks. In *Proceedings of the ACM Internet Measurement Conference*, pages 111–124, 2020.
- [20] Amazon. Enhanced domain protections for amazon cloudfront requests. <https://aws.amazon.com/blogs/security/enhanced-domain-protections-for-amazon-cloudfront-requests/>, 2020. Accessed: 2025-05-27.
- [21] Collin Anderson. The hidden internet of iran: Private address allocations on a national network. *arXiv preprint arXiv:1209.6398*, 2012.
- [22] Collin Anderson. Dimming the internet: Detecting throttling as a mechanism of censorship in iran, 2013.
- [23] Elli Androulaki, Seung Geol Choi, Steven M. Bellovin, and Tal Malkin. Reputation Systems for Anonymous Networks. In *Privacy Enhancing Technologies*, volume 5134, pages 202–218. 2008.
- [24] Sebastian Angel and Srinath Setty. Unobservable communication over fully untrusted infrastructure. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation*, pages 551–569, Savannah, GA, 2016.

- [25] Ars Technica. Google disables domain fronting capability used to evade censors. <https://arstechnica.com/information-technology/2018/04/google-disables-domain-fronting-capability-used-to-evade-censors/>, 2020. Accessed: 2025-05-27.
- [26] Simurgh Aryan, Homa Aryan, and J. Alex Halderman. Internet censorship in Iran : A first look. In *Proceedings of the 3rd USENIX Workshop on Free and Open Communications on the Internet*, 2013.
- [27] Simurgh Aryan, Homa Aryan, and J Alex Halderman. Internet censorship in iran: A first look. In *3rd USENIX Workshop on Free and Open Communications on the Internet (FOCI 13)*, 2013.
- [28] Itzhak Aviv and Uri Ferri. Russian-Ukraine armed conflict: Lessons learned on the digital ecosystem. *International Journal of Critical Infrastructure Protection*, 43:100637, 2023.
- [29] Roger Baig, Ramon Roca, Felix Freitag, and Leandro Navarro. guifi.net, a crowd-sourced network infrastructure held in common. *Computer Networks*, 90:150–165, 2015.
- [30] Ludovic Barman, Italo Dacosta, Mahdi Zamani, Ennan Zhai, Apostolos Pyrgelis, Bryan Ford, Jean-Pierre Hubaux, and Joan Feigenbaum. Prifi: Low-latency anonymity for organizational networks. *Proceedings on Privacy Enhancing Technologies*, 2020(4), 2020.
- [31] Ludovic Barman, Moshe Kol, David Lazar, Yossi Gilad, and Nickolai Zeldovich. Groove: Flexible Metadata-Private messaging. In *Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation*, pages 735–750, Carlsbad, CA, 2022.
- [32] Diogo Barradas, Nuno Santos, Luís Rodrigues, and Vítor Nunes. Poking a hole in the wall: Efficient censorship-resistant internet communications by parasitizing on webrtc. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 35–48, 2020.
- [33] Diogo Barradas, Nuno Santos, and Luís Rodrigues. Effective detection of multimedia protocol tunneling using machine learning. In *Proceedings of the 27th USENIX Security Symposium*, pages 169–185, Baltimore, MD, USA, August 2018.

- [34] Mihir Bellare and Sara K Miner. A forward-secure digital signature scheme. In *Proceedings of the Annual International Cryptology Conference*, pages 431–448, 1999.
- [35] Harel Berger, Micah Sherr, and Adam Aviv. Cadence: A simulator for human movement-based communication protocols. In *Proceedings of the 16th Cyber Security Experimentation and Test Workshop*, CSET '23, page 26–31, 2023.
- [36] John Bethencourt, Jason Franklin, and Mary K Vernon. Mapping internet sensors with probe response attacks. In *USENIX security symposium*, pages 193–208, 2005.
- [37] Alexander Bienstock, Paul Rösler, and Yi Tang. ASMesh: Anonymous and secure messaging in mesh networks using stronger, anonymous double ratchet. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 1–15, 2023.
- [38] Benedikt Birtel and Christian Rossow. Slitheen++: Stealth {TLS-based} decoy routing. In *10th USENIX Workshop on Free and Open Communications on the Internet (FOCI 20)*, 2020.
- [39] Zachary S. Bischof, Kennedy Pitcher, Esteban Carisimo, Amanda Meng, Rafael Bezerra Nunes, Ramakrishna Padmanabhan, Margaret E. Roberts, Alex C. Snoeren, and Alberto Dainotti. Destination unreachable: Characterizing Internet outages and shutdowns. In *Proceedings of the Special Interest Group on Data Communications Conference*, 2023.
- [40] Cecylia Bocovich, Arlo Breault, David Fifield, Xiaokang Wang, et al. Snowflake, a censorship circumvention system using temporary {WebRTC} proxies. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 2635–2652, 2024.
- [41] Cecylia Bocovich and Ian Goldberg. Slitheen: Perfectly imitated decoy routing through traffic replacement. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1702–1714, 2016.
- [42] Cecylia Bocovich and Ian Goldberg. Secure asymmetry and deployability for decoy routing systems. *Proceedings on Privacy Enhancing Technologies*, 2018.
- [43] Bouncy Castle – Open-source cryptographic APIs. <https://www.bouncycastle.org>.
- [44] Nuria Busom, Ronald Petrlic, Francesc Sebé, Christoph Sorge, and Magda Valls. A privacy-preserving reputation system with user rewards. *Journal of Network and Computer Applications*, 80:58–66, 2017.

- [45] Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian Goldberg. A systematic approach to developing and evaluating website fingerprinting defenses. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 227–238, 2014.
- [46] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Annual international cryptology conference*, pages 56–72. Springer, 2004.
- [47] Daniel Camps-Mur, Andres Garcia-Saavedra, and Pablo Serrano. Device-to-device communications with wi-fi direct: overview and experimentation. *IEEE wireless communications*, 20(3):96–104, 2013.
- [48] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.
- [49] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology*, 1:65–75, 1988.
- [50] Chen Chen, Daniele E Asoni, Adrian Perrig, David Barrera, George Danezis, and Carmela Troncoso. Taranet: Traffic-analysis resistant anonymity at the network layer. In *Proceedings of the 2018 IEEE European Symposium on Security and Privacy*, pages 137–152, 2018.
- [51] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd Conference on Knowledge Discovery and Data Mining*, pages 785–794, San Francisco, CA, USA, 2016. ACM.
- [52] Erik Chi, Gaukas Wang, J Alex Halderman, Eric Wustrow, and Jack Wampler. Just add water: Webassembly-based circumvention transports. *Free and Open Communications on the Internet*, 2023.
- [53] Benny Chor, Niv Gilboa, and Moni Naor. *Private information retrieval by keywords*. Technion-IIT, Department of Computer Science, 1997.
- [54] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *Journal of the ACM (JACM)*, 45(6):965–981, 1998.
- [55] Martin Chulov. Syria shuts off internet access across the country. <https://www.theguardian.com/world/2012/nov/29/syria-blocks-internet>, 2012.

- [56] Catalin Cimpanu. Kazakhstan government is now intercepting all https traffic. <https://www.zdnet.com/article/kazakhstan-government-is-now-intercepting-all-https-traffic/>, 2019. Accessed: 2025-06-05.
- [57] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing privacy enhancing technologies: international workshop on design issues in anonymity and unobservability*, pages 46–66. Springer, 2001.
- [58] Katriel Cohn-Gordon, Cas Cremers, and Luke Garratt. On post-compromise security. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, pages 164–178. IEEE, 2016.
- [59] Cláudio Correia, Miguel Correia, and Luís Rodrigues. Using range-revocable pseudonyms to provide backward unlinkability in the edge. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, page 3018–3032, 2023.
- [60] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *Proceedings of the 2015 IEEE Symposium on Security and Privacy*, pages 321–338, 2015.
- [61] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *2015 IEEE Symposium on Security and Privacy*, pages 321–338. IEEE, 2015.
- [62] Henry Corrigan-Gibbs and Bryan Ford. Dissent: accountable anonymous group messaging. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 340–350, 2010.
- [63] Ronald Cramer and Victor Shoup. Signature schemes based on the strong rsa assumption. *ACM Transactions on Information and System Security (TISSEC)*, 3(3):161–185, 2000.
- [64] Alberto Dainotti, Claudio Squarcella, Emile Aben, Kimberly C. Claffy, Marco Chiesa, Michele Russo, and Antonio Pescapé. Analysis of country-wide internet outages caused by censorship. In *Proceedings of the Internet Measurement Conference*, pages 1–18, 2011.
- [65] Alberto Dainotti, Claudio Squarcella, Emile Aben, Kimberly C Claffy, Marco Chiesa, Michele Russo, and Antonio Pescapé. Analysis of country-wide internet outages

- caused by censorship. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 1–18, 2011.
- [66] Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Fast and private computation of cardinality of set intersection and union. In *Proceedings of the International Conference on Cryptology and Network Security*, pages 218–231, 2012.
 - [67] Primavera De Filippi and Félix Tréguer. Expanding the internet commons: The subversive potential of wireless community networks. *Journal of Peer Production, Issue*, (6), 2015.
 - [68] Wladimir De la Cadena, Asya Mitseva, Jens Hiller, Jan Pennekamp, Sebastian Reuter, Julian Filter, Thomas Engel, Klaus Wehrle, and Andriy Panchenko. Traffic-slicer: Fighting website fingerprinting attacks with traffic splitting. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1971–1985, 2020.
 - [69] Ronald J Deibert and Nart Villeneuve. Firewalls and power: An overview of global state censorship of the internet. *Human rights in the digital age*, pages 125–138, 2016.
 - [70] F. Denis, E. Eaton, T. Lepoint, and C. A. Wood. Key Blinding for Signature Schemes. <https://www.ietf.org/archive/id/draft-irtf-cfrg-signature-key-blinding-06.html>, 2024.
 - [71] Claudia Diaz, Harry Halpin, and Aggelos Kiayias. The Nym Network. <https://nymtech.net/nym-whitepaper.pdf>, 2021.
 - [72] Tassos Dimitriou. Decentralized Reputation. In *Proceedings of the 11th ACM Conference on Data and Application Security and Privacy*, pages 119–130, 2021.
 - [73] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the USENIX security symposium*, pages 303–320, 2004.
 - [74] Patrick Dubroy and Ravin Balakrishnan. A study of tabbed browsing among mozilla firefox users. https://www.dgp.toronto.edu/~ravin/papers/chi2010_tabbedbrowsing.pdf, 2010.
 - [75] Kevin P Dyer, Scott E Coull, Thomas Ristenpart, and Thomas Shrimpton. Protocol misidentification made easy with format-transforming encryption. In *Proceedings of*

the 2013 ACM SIGSAC conference on Computer & communications security, pages 61–72, 2013.

- [76] Kevin P Dyer, Scott E Coull, and Thomas Shrimpton. Marionette: A programmable network traffic obfuscation system. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 367–382, 2015.
- [77] Edward Eaton, Tancrède Lepoint, and Christopher A. Wood. Security analysis of signature schemes with key blinding. Cryptology ePrint Archive, Paper 2023/380, 2023.
- [78] Roya Ensafi, David Fifield, Philipp Winter, Nick Feamster, Nicholas Weaver, and Vern Paxson. Examining how the Great Firewall discovers hidden circumvention servers. In *Proceedings of the ACM Internet Measurement Conference*, 2015.
- [79] Roya Ensafi, David Fifield, Philipp Winter, Nick Feamster, Nicholas Weaver, and Vern Paxson. Examining how the great firewall discovers hidden circumvention servers. In *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*, pages 445–458, 2015.
- [80] Saba Eskandarian and Dan Boneh. Clarion: Anonymous communication from multiparty shuffling protocols. *Proceedings of the 29th Network and Distributed System Security Symposium*, 2022.
- [81] Saba Eskandarian, Henry Corrigan-Gibbs, Matei Zaharia, and Dan Boneh. Express: Lowering the cost of metadata-hiding communication with cryptographic privacy. In *Proceedings of the 30th USENIX Security Symposium*, pages 1775–1792, 2021.
- [82] Ellis Fenske and Aaron Johnson. Bytes to schlep? use a fep: Hiding protocol metadata with fully encrypted protocols. *arXiv preprint arXiv:2405.13310*, 2024.
- [83] Dandy Fenz, Dustin Lange, Astrid Rheinländer, Felix Naumann, and Ulf Leser. Efficient similarity search in very large string sets. In *Proceedings of the 24th Scientific and Statistical Database Management*, pages 262–279, 2012.
- [84] David Fifield. *Threat modeling and circumvention of Internet censorship*. PhD thesis, EECS Department, University of California, Berkeley, 2017.
- [85] David Fifield and Mia Gil Epner. Fingerprintability of webrtc. *arXiv preprint arXiv:1605.08805*, 2016.

- [86] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. Blocking-resistant communication through domain fronting. *Proceedings on Privacy Enhancing Technologies*, 2015.
- [87] David Fifield and Lynn Tsai. Censors’ delay in blocking circumvention proxies. In *Free and Open Communications on the Internet*, 2016.
- [88] Gabriel Figueira, Diogo Barradas, and Nuno Santos. Stegozoa: Enhancing WebRTC covert channels with video steganography for internet censorship circumvention. In *Asia CCS*. ACM, 2022.
- [89] Kyle Fredrickson, Ioannis Demertzis, James Hughes, and Darrell Long. Sparta: Practical Anonymity with Long-Term Resistance to Traffic Analysis . In *Proceedings of the 2025 IEEE Symposium on Security and Privacy (SP)*, pages 1457–1473, 2025.
- [90] Sergey Frolov, Fred Douglas, Will Scott, Allison McDonald, Benjamin VanderSloot, Rod Hynes, Adam Kruger, Michalis Kallitsis, David G. Robinson, Steve Schultze, Nikita Borisov, Alex Halderman, and Eric Wustrow. An ISP-scale deployment of TapDance. In *Proceedings of the 7th USENIX Workshop on Free and Open Communications on the Internet*, Vancouver, BC, Canada, August 2017.
- [91] Sergey Frolov, Jack Wampler, Sze Chuen Tan, J Alex Halderman, Nikita Borisov, and Eric Wustrow. Conjure: Summoning proxies from unused address space. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2215–2229, 2019.
- [92] Sergey Frolov, Jack Wampler, and Eric Wustrow. Detecting probe-resistant proxies. In *NDSS*, 2020.
- [93] Sergey Frolov and Eric Wustrow. {HTTPT}: A {Probe-Resistant} proxy. In *10th USENIX Workshop on Free and Open Communications on the Internet (FOCI 20)*, 2020.
- [94] Xinwen Fu, Bryan Graham, Dong Xuan, Riccardo Bettati, and Wei Zhao. Empirical and theoretical evaluation of active probing attacks and their countermeasures. In *International Workshop on Information Hiding*, pages 266–281. Springer, 2004.
- [95] Philip Garrison, Esther Han Beol Jang, Michael A. Lithgow, and Nicolás Andrés Pace. The Network Is an Excuse: Hardware Maintenance Supporting Community. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW2), 2021.

- [96] John Geddes, Max Schuchard, and Nicholas Hopper. Cover your acks: Pitfalls of covert channel censorship circumvention. In *Proceedings of the 20th ACM Conference on Computer and Communications Security*, pages 361–372, Berlin, Germany, November 2013.
- [97] Shahabeddin Geravand and Mahmood Ahmadi. Bloom filter applications in network security: A state-of-the-art survey. *Computer Networks*, 57(18):4047–4064, 2013.
- [98] Edgardo Gerck et al. Overview of Certification Systems: x. 509, CA, PGP and SKIP. *The Black Hat Briefings*, 99, 1997.
- [99] David Goldschlag, Michael Reed, and Paul Syverson. Onion routing. *Communications of the ACM*, 42(2):39–41, 1999.
- [100] Devashish Gosain, Anshika Agarwal, Sambuddho Chakravarty, and Hrishikesh B Acharya. The devil’s in the details: Placing decoy routers in the internet. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pages 577–589, 2017.
- [101] Margarita Grinko, Sarvin Qalandar, Dave Randall, and Volker Wulf. Nationalizing the internet to break a protest movement: Internet shutdown and counter-appropriation in iran of late 2019. *Proc. ACM Hum.-Comput. Interact.*, 6(CSCW2), November 2022.
- [102] Margarita Grinko, Sarvin Qalandar, Dave Randall, and Volker Wulf. Nationalizing the internet to break a protest movement: Internet shutdown and counter-appropriation in iran of late 2019. *Proceedings of the ACM on human-computer interaction*, 6(CSCW2):1–21, 2022.
- [103] Rachid Guerraoui, Anne-Marie Kermarrec, Anastasiia Kucherenko, Rafael Pinot, and Sasha Voitovich. On the inherent anonymity of gossiping, 2023. arXiv:2308.02477 [cs.DC].
- [104] Stan Gurtler and Ian Goldberg. SoK: Privacy-Preserving Reputation Systems. *Proceedings on Privacy Enhancing Technologies*, 2021(1):107–127, 2021.
- [105] Gus. Tor in russia: A call for more webtunnel bridges. <https://blog.torproject.org/call-for-webtunnel-bridges/>, 2024. The Tor Project, November 28, 2024.

- [106] Zygmunt J Haas, Joseph Y Halpern, and Li Li. Gossip-based ad hoc routing. *IEEE/ACM Transactions on networking*, 14(3):479–491, 2006.
- [107] Bridger Hahn, Rishab Nithyanand, Phillipa Gill, and Rob Johnson. Games without frontiers: Investigating video games as a covert channel. In *2016 IEEE European Symposium on Security and Privacy*, pages 63–77, Saarbrücken, Germany, 2016. IEEE.
- [108] Shaddi Hasan, Yahel Ben-David, Giulia Fanti, Eric Brewer, and Scott Shenker. Building dissent networks: Towards effective countermeasures against large-scale communications blackouts. In *Free and Open Communications on the Internet*, 2013.
- [109] Chaya Hiruncharoenvate, Zhiyuan Lin, and Eric Gilbert. Algorithmically bypassing censorship on Sina Weibo with nondeterministic homophone substitutions. In *Proceedings of the 9th International Conference on Web and Social Media*, Oxford, UK, 2015. AAAI.
- [110] Nguyen Phong Hoang, Jakub Dalek, Masashi Crete-Nishihata, Nicolas Christin, Vinod Yegneswaran, Michalis Polychronakis, and Nick Feamster. GFWeb: Measuring the Great Firewall’s Web censorship at scale. In *USENIX Security Symposium*, 2024.
- [111] Nguyen Phong Hoang, Panagiotis Kintis, Manos Antonakakis, and Michalis Polychronakis. An empirical study of the i2p anonymity network and its censorship resistance. In *Proceedings of the internet measurement conference 2018*, pages 379–392, 2018.
- [112] Nguyen Phong Hoang, Arian Akhavan Niaki, Jakub Dalek, Jeffrey Knockel, Pellaeon Lin, Bill Marczak, Masashi Crete-Nishihata, Phillipa Gill, and Michalis Polychronakis. How great is the great firewall? measuring china’s {DNS} censorship. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3381–3398, 2021.
- [113] John Holowczak and Amir Houmansadr. Cachebrowser: Bypassing chinese censorship without proxies using cached content. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 70–83, 2015.
- [114] Amir Houmansadr, Chad Brubaker, and Vitaly Shmatikov. The parrot is dead: Observing unobservable network communications. In *2013 IEEE Symposium on Security and Privacy*, pages 65–79. IEEE, 2013.

- [115] Amir Houmansadr, Giang T.K. Nguyen, Matthew Caesar, and Nikita Borisov. Cirripede: Circumvention infrastructure using router redirection with plausible deniability. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, pages 187–200, Chicago, IL, USA, October 2011.
- [116] Amir Houmansadr, Thomas J. Riedl, Nikita Borisov, and Andrew C. Singer. I want my voice to be heard: IP over voice-over-IP for unobservable censorship circumvention. In *Proceedings of the 20th Network and Distributed System Security Symposium*, San Diego, CA, USA, February 2013.
- [117] Amir Houmansadr, Edmund L Wong, and Vitaly Shmatikov. No direction home: The true cost of routing around decoys. In *NDSS*. Citeseer, 2014.
- [118] DataCebo Inc. Sdv — synthetic data vault. <https://docs.sdv.dev/sdv>, 2025.
- [119] Nicola J Bidwell. Wireless in the weather-world and community networks made to last. In *Proceedings of the 16th Participatory Design Conference 2020*, page 126–136, 2020.
- [120] Peipei Jiang, Qian Wang, Jianhao Cheng, Cong Wang, Lei Xu, Xinyu Wang, Yihao Wu, Xiaoyuan Li, and Kui Ren. Boomerang:{Metadata-Private} messaging under hardware trust. In *Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation*, pages 877–899, 2023.
- [121] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1:36–63, 2001.
- [122] Simon Josefsson and Ilari Liusvaara. Edwards-curve digital signature algorithm (eddsa). Technical report, 2017.
- [123] Marc Juarez, Sadia Afroz, Gunes Acar, Claudia Diaz, and Rachel Greenstadt. A critical evaluation of website fingerprinting attacks. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 263–274, 2014.
- [124] Saqib A Kakvi, Keith M Martin, Colin Putman, and Elizabeth A Quaglia. Sok: anonymous credentials. In *International Conference on Research in Security Standardisation*, pages 129–151. Springer, 2023.
- [125] Sina Kamali and Diogo Barradas. Anix: Anonymous blackout-resistant microblogging with message endorsing. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 1381–1399. IEEE, 2025.

- [126] Michael Kan. GitHub unblocked in china after former google head slams its censorship | computerworld.
- [127] Syed Karim, Aaron Rogers, and Edward Birrane. Bridging the Information Divide: Offering Global Access to Digital Content with a Disruptive CubeSat Constellation. In *Proceedings of the 28th Annual AIAA/USU Conference on Small Satellites*, 2014.
- [128] J Karlin, D Ellard, AW Jackson, CE Jones, G Lauer, DP Mankins, and TW Strayer. Decoy routing: Toward unblockable Internet communication. In *Proceedings of the 1st USENIX Workshop on Free and Open Communications on the Internet*, San Francisco, CA, USA, August 2011.
- [129] Josh Karlin, Daniel Ellard, Alden W Jackson, Christine E Jones, Greg Lauer, David P Mankins, and W Timothy Strayer. Decoy routing: Toward unblockable internet communication. In *USENIX workshop on free and open communications on the Internet (FOCI 11)*, 2011.
- [130] Divyank Katira, Gurshabad Grover, Kushagra Singh, and Varun Bansal. Censor-Watch: On the implementation of online censorship in India. In *Free and Open Communications on the Internet*, 2023.
- [131] Darya Kaviani, Deevashwer Rathee, Bhargav Annem, and Raluca Ada Popa. Myco: Unlocking Polylogarithmic Accesses in Metadata-Private Messaging . In *Proceedings of the 2025 IEEE Symposium on Security and Privacy*, pages 4419–4437, 2025.
- [132] Daphne Keller. *Towards an Internet Free of Censorship II: Perspectives in Latin America*. Facultad de Derecho, Centro de Estudios en Libertad de Expresión y Acceso a la Información, 2017.
- [133] Sheharbano Khattak, Tariq Elahi, Laurent Simon, Colleen M Swanson, Steven J Murdoch, and Ian Goldberg. Sok: Making sense of censorship resistance systems. In *Proceedings on Privacy Enhancing Technologies*, volume 2016, pages 37–61, Darmstadt, Germany, 2016.
- [134] Sheharbano Khattak, Tariq Elahi, Laurent Simon, Colleen M. Swanson, Steven J. Murdoch, and Ian Goldberg. SoK: Making sense of censorship resistance systems. *Privacy Enhancing Technologies*, 2016(4):37–61, 2016.
- [135] Jeffrey Knockel, Masashi Crete-Nishihata, Jason Q. Ng, Adam Senft, and Jedidiah R. Crandall. Every rose has its thorn: Censorship and surveillance on social video

- platforms in China. In *Proceedings of the 5th USENIX Workshop on Free and Open Communications on the Internet*, 2015.
- [136] Jeffrey Knockel, Lotus Ruan, and Masashi Crete-Nishihata. Measuring decentralization of Chinese keyword censorship via mobile games. In *Free and Open Communications on the Internet*, 2017.
 - [137] John Koetsier. Ukrainians prepping for internet loss by getting apps for offline, private, mesh communications. <https://www.forbes.com/sites/johnkoetsier/2022/02/25/ukrainians-prepping-for-internet-loss-by-getting-apps-for-offline-private-mesh-communications/?sh=710a40b7791d>, 2022.
 - [138] Patrick Tser Jern Kon, Aniket Gattani, Dhiraj Saharia, Tianyu Cao, Diogo Barradas, Ang Chen, Micah Sherr, and Benjamin E Ujcich. Netshuffle: Circumventing censorship with shuffle proxies at the edge. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 3497–3514. IEEE, 2024.
 - [139] Patrick Tser Jern Kon, Sina Kamali, Jinyu Pei, Diogo Barradas, Ang Chen, Micah Sherr, and Moti Yung. {SpotProxy}: Rediscovering the cloud for censorship circumvention. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 2653–2670, 2024.
 - [140] Hugo Krawczyk. Cryptographic extraction and key derivation: The hkdf scheme. In *Annual Cryptology Conference*, pages 631–648. Springer, 2010.
 - [141] Christiane Kuhn, Martin Beck, Stefan Schiffner, Eduard Jorswieck, and Thorsten Strufe. On privacy notions in anonymous communication. *arXiv preprint arXiv:1812.05638*, 2018.
 - [142] Christiane Kuhn, Martin Beck, Stefan SCHIFFNER, Eduard Jorswieck, and Thorsten Strufe. On privacy notions in anonymous communication. *Proceedings on Privacy Enhancing Technologies*, 2019(2), 2019.
 - [143] Juhi Kulshrestha, Motahhare Eslami, Johnnatan Messias, Muhammad Bilal Zafar, Saptarshi Ghosh, and Krishna P. Gummadi. Dataset: Search and recommendation systems: A study of the impact of algorithm transparency on user behavior, 2021.
 - [144] Juhi Kulshrestha, Marcos Oliveira, Orkut Karaçalık, Denis Bonnay, and Claudia Wagner. Web routineness and limits of predictability: Investigating demographic and behavioral differences using web tracking data. *Proceedings of the International AAAI Conference on Web and Social Media*, 15(1):327–338, May 2021.

- [145] Juhi Kulshrestha, Marcos Oliveira, Orkut Karaçalik, Denis Bonnay, and Claudia Wagner. Web routineness and limits of predictability: Investigating demographic and behavioral differences using web tracking data. *Proceedings of the International AAAI Conference on Web and Social Media*, 15(1):327–338, May 2021.
- [146] Albert Kwon, Henry Corrigan-Gibbs, Srinivas Devadas, and Bryan Ford. Atom: Horizontally scaling strong anonymity. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 406–422, 2017.
- [147] Albert Kwon, David Lu, and Srinivas Devadas. XRD: Scalable messaging system with cryptographic privacy. In *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation*, pages 759–776, 2020.
- [148] Albert Hyukjae Kwon, David Lazar, Srinivas Devadas, and Bryan Ford. Riffle: An efficient communication system with strong anonymity. *Proceedings on Privacy Enhancing Technologies*, 2016(2):115–134, 2016.
- [149] Simon Langowski, Sacha Servan-Schreiber, and Srinivas Devadas. Trellis: Robust and scalable metadata-private anonymous broadcast. *traffic*, 9(45):82.
- [150] David Lazar, Yossi Gilad, and Nickolai Zeldovich. Karaoke: Distributed private messaging immune to passive traffic analysis. In *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation*, pages 711–725, 2018.
- [151] David Lazar and Nickolai Zeldovich. Alpenhorn: Bootstrapping secure communication without leaking metadata. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation*, pages 571–586, 2016.
- [152] Adam Lerner, Giulia Fanti, Yahel Ben-David, Jesus Garcia, Paul Schmitt, and Barath Raghavan. Rangzen: Anonymously Getting the Word Out in a Blackout, 2016. arXiv:1612.03371 [cs].
- [153] Shuai Li, Mike Schliep, and Nick Hopper. Facet: Streaming over videoconferencing for censorship circumvention. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 163–172, Scottsdale, AZ, USA, 2014.
- [154] Po-Ching Lin, Ping-Chung Li, et al. Inferring openflow rules by active probing in software-defined networks. In *2017 19th International Conference on Advanced Communication Technology (ICACT)*, pages 415–420. IEEE, 2017.

- [155] Anna Litvinenko. Re-defining borders online: Russia’s strategic narrative on internet sovereignty. *Media and Communication*, 9:5–15, 2021.
- [156] Chao Liu, Ryen W. White, and Susan Dumais. Understanding web browsing behaviors through weibull analysis of dwell time. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2010.
- [157] Yue Liu, David R. Bild, David Adrian, Gulshan Singh, Robert P. Dick, Dan S. Wallach, and Z. Morley Mao. Performance and energy consumption analysis of a delay-tolerant network for censorship-resistant communication. In *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 257–266, 2015.
- [158] Anna Harbluk Lorimer, Lindsey Tulloch, Cecylia Bocovich, and Ian Goldberg. Ostracopithecus: Overt user simulation for censorship circumvention. In *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society*, pages 137–150, 2021.
- [159] Redis Ltd. Redis. <https://redis.io/>, 2025.
- [160] Donghang Lu, Thomas Yurek, Samarth Kulshreshtha, Rahul Govind, Aniket Kate, and Andrew Miller. Honeybadgermpc and asynchromix: Practical asynchronous MPC and its application to anonymous communication. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 887–903, 2019.
- [161] Li Lu, Jinsong Han, Yunhao Liu, Lei Hu, Jin-Peng Huai, Lionel Ni, and Jian Ma. Pseudo trust: Zero-knowledge authentication in anonymous p2ps. *IEEE Transactions on Parallel and Distributed Systems*, 19(10):1325–1337, 2008.
- [162] Kyle MacMillan, Jordan Holland, and Prateek Mittal. Evaluating snowflake as an indistinguishable censorship circumvention tool. *arXiv preprint arXiv:2008.03254*, 2020.
- [163] Eleanor Marchant and Nicole Stremlau. A spectrum of shutdowns: Reframing internet shutdowns from africa. *International Journal of Communication*, 14, 2020.
- [164] V Gayoso Martínez, L Hernández Encinas, et al. A comparison of the standardized versions of ecies. In *2010 Sixth International Conference on Information Assurance and Security*, pages 1–4. IEEE, 2010.

- [165] Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.
- [166] Alexander Master and Christina Garman. A worldwide view of nation-state internet censorship. *Free and Open Communications on the Internet*, 2023.
- [167] Jon McLachlan, Andrew Tran, Nicholas Hopper, and Yongdae Kim. Scalable onion routing with torsk. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 590–599, 2009.
- [168] Kelsey Merrill, Zachary Newman, Santiago Torres-Arias, and Karen R. Sollins. Speranza: Usable, privacy-friendly software signing. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, page 3388–3402, 2023.
- [169] O RABIN Michael and MO Rabin. How to exchange secrets by oblivious transfer. Technical report, Technical report, Aiken Computation Laboratory. Harvard University, 1981.
- [170] Robbie Mitchell. Tracking Internet Shutdowns in 2023. <https://pulse.internetsociety.org/blog/tracking-internet-shutdowns-in-2023>.
- [171] Yilin Mo and Bruno Sinopoli. Secure control against replay attacks. In *2009 47th annual Allerton conference on communication, control, and computing (Allerton)*, pages 911–918. IEEE, 2009.
- [172] Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. Skypemorph: Protocol obfuscation for tor bridges. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 97–108, 2012.
- [173] Soo-Jin Moon, Milind Srivastava, Yves Bieri, Ruben Martins, and Vyas Sekar. Pryde: A modular generalizable workflow for uncovering evasion attacks against stateful firewall deployments. In *Symposium on Security & Privacy*. IEEE, 2024.
- [174] Paul Murley, Zane Ma, Joshua Mason, Michael Bailey, and Amin Kharraz. Websocket adoption and the landscape of the real-time web. In *Proceedings of the Web Conference 2021*, pages 1192–1203, 2021.
- [175] Heather Murphy. Ominous Text Message Sent to Protesters in Kiev Sends Chills Around the Internet. <https://archive.nytimes.com/thelede.blogs.nytimes.com/2014/01/22/ominous-text-message-sent-to-protesters-in-kyiv-sends-chills-around-the-internet/>, 2014.

- [176] Steven Lee Myers and Paul Mozur. China is waging a disinformation war against hong kong protesters. <https://www.nytimes.com/2019/08/13/world/asia/hong-kong-protests-china.html>, 2019.
- [177] Milad Nasr, Sadegh Farhang, Amir Houmansadr, and Jens Grossklags. Enemy at the gateways: Censorship-resilient proxy distribution using game theory. In *Network and Distributed System Security*. The Internet Society, 2019.
- [178] Milad Nasr and Amir Houmansadr. Game of decoys: Optimal decoy routing through game theory. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1727–1738, 2016.
- [179] Milad Nasr, Hadi Zolfaghari, and Amir Houmansadr. The waterfall of liberty: Decoy routing circumvention that resists routing attacks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2037–2052, 2017.
- [180] Kelly Ng. Chinese censors take aim at AirDrop and Bluetooth. <https://www.bbc.com/news/world-asia-china-65830185>, 2023.
- [181] Arian Akhavan Niaki, Shinyoung Cho, Zachary Weinberg, Nguyen Phong Hoang, Abbas Razaghpanah, Nicolas Christin, and Phillipa Gill. Iclab: A global, longitudinal internet censorship measurement platform. In *Proceedings of the 41st IEEE Symposium on Security and Privacy*, 2020.
- [182] Shadowsocks org. Shadowsocks. <https://shadowsocks.org>, 2025.
- [183] Encode OSS. Django rest framework. <https://www.django-rest-framework.org/>, 2025.
- [184] Ramakrishna Padmanabhan, , Arturo Filastò, Maria Xynou, Ram Sundara Raman, Kennedy Middleton, Mingwei Zhang, Doug Madory, Molly Roberts, and Alberto Dainotti. A multi-perspective view of Internet censorship in Myanmar. In *Free and Open Communications on the Internet*, 2021.
- [185] Pallet. Flask framework. <https://flask.palletsprojects.com/en/stable/>, 2025.
- [186] John H. Parmelee and Shannon L. Bichard. *Politics and the Twitter revolution : how tweets influence the relationship between political leaders and the public*. Lexington studies in political communication. Lexington Books, 2013.

- [187] Trevor Perrin and Moxie Marlinspike. The double ratchet algorithm. *GitHub wiki*, 112, 2016.
- [188] Neil Perry, Bruce Spang, Saba Eskandarian, and Dan Boneh. Strong Anonymity for Mesh Messaging, 2022. arXiv:2207.04145 [cs].
- [189] Ania M. Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The loopix anonymity system. In *Proceedings of the USENIX Security Symposium*, 2017.
- [190] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Proceedings of the 26th Network and Distributed Systems Security Symposium*, 2019.
- [191] Fanny Potkin and Jessie Pang. Offline message app downloaded over million times after myanmar coup. <https://www.reuters.com/article/idUSKBN2A22H0/>, 2021.
- [192] Amogh Pradeep, Hira Javaid, Ryan Williams, Antoine Rault, David Choffnes, Stevens Le Blond, and Bryan Ford. Moby: A Blackout-Resistant Anonymity Network for Mobile Devices. *Proceedings on Privacy Enhancing Technologies*, 2022(3):247–267, 2022.
- [193] Amogh Pradeep, Hira Javaid, Ryan Williams, Antoine Rault, David Choffnes, Stevens Le Blond, and Bryan Alexander Ford. Moby: A blackout-resistant anonymity network for mobile devices. *Proceedings on Privacy Enhancing Technologies*, 2022(3):247–267, 2022.
- [194] PyCryptodome. Pycryptodome. <https://www.pycryptodome.org/>, 2025.
- [195] Ram Sundara Raman, Leonid Evdokimov, Eric Wurstrow, J Alex Halderman, and Roya Ensafi. Investigating large scale https interception in kazakhstan. In *Proceedings of the ACM Internet Measurement Conference*, pages 125–132, 2020.
- [196] Reethika Ramesh, Ram Sundara Raman, Matthew Bernhard, Victor Ongkowijaya, Leonid Evdokimov, Anne Edmundson, Steven Sprecher, Muhammad Ikram, and Roya Ensafi. Decentralized control: A case study of russia. In *Proceedings of the 27th Network and Distributed Systems Security Symposium*, 2020.
- [197] RFC 8032 - Edwards-Curve Digital Signature Algorithm (EdDSA). <https://www.rfc-editor.org/rfc/rfc8032>.

- [198] Alexander Ronzhyn. The use of facebook and twitter during the 2013-2014 protests in ukraine. In *Proceedings of the European Conference on Social Media*, pages 442–449, 2014.
- [199] Marc B Rosen, James Parker, and Alex J Malozemoff. Balboa: Bobbing and weaving around network censorship. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3399–3413, 2021.
- [200] Sajin Sasy and Ian Goldberg. Sok: Metadata-protecting communication systems. *Proceedings on Privacy Enhancing Technologies*, 2024.
- [201] Sophie Schmieg. Reconstructing public keys from signatures . <https://keymaterial.net/2024/06/15/reconstructing-public-keys-from-signatures/>, 2024.
- [202] Srinath Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. In *Annual International Cryptology Conference*, pages 704–737. Springer, 2020.
- [203] Stefaan Seys and Bart Preneel. Arm: Anonymous routing protocol for mobile ad hoc networks. *International Journal of Wireless and Mobile Computing*, 3(3):145–155, 2009.
- [204] Peter Shadbolt. Firechat in hong kong: How an app tapped its way into the protests. <https://www.cnn.com/2014/10/16/tech/mobile/tomorrow-transformed-firechat/index.html>, 2014.
- [205] Adrian Shahbaz, Allie Funk, and Kian Vesteinsson. Countering an authoritarian overhaul of the internet (freedom on the net). <https://freedomhouse.org/report/freedom-net/2022/countering-authoritarian-overhaul-internet>, 2022.
- [206] Piyush Kumar Sharma, Devashish Gosain, and Sambuddho Chakravarty. Camoufler: Accessing the censored web by utilizing instant messaging channels. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pages 147–161, 2021.
- [207] Piyush Kumar Sharma, Rishi Sharma, Kartikey Singh, Mukulika Maity, and Sambuddho Chakravarty. Dolphin: a cellular voice based internet shutdown resistance system. *Proceedings on Privacy Enhancing Technologies*, pages 589–607, 2023.
- [208] shelikhoo and ggus. Hiding in plain sight: Introducing webtunnel. <https://blog.torproject.org/introducing-webtunnel-evading-censorship-by-hiding-in-plain-sight/>, 2024. Accessed: 2025-05-27.

- [209] Meng Shen, Kexin Ji, Zhenbo Gao, Qi Li, Liehuang Zhu, and Ke Xu. Subverting website fingerprinting defenses with robust traffic representation. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 607–624, Anaheim, CA, August 2023.
- [210] Meng Shen, Kexin Ji, Zhenbo Gao, Qi Li, Liehuang Zhu, and Ke Xu. Subverting website fingerprinting defenses with robust traffic representation. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 607–624, 2023.
- [211] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 1928–1943, 2018.
- [212] Monika Skowron, Artur Janicki, and Wojciech Mazurczyk. Traffic fingerprinting attacks on internet of things using machine learning. *IEEE Access*, 8:20386–20400, 2020.
- [213] Travis Smith, Vasilios Pappademetriou, William Gaylord, and Cooper Van Kampen. Vibe: Meshing inexpensive routers. In Kohei Arai, editor, *Advances in Information and Communication*, pages 578–587. Springer Nature Switzerland, 2023.
- [214] Ilona Stadnik. Russia: An independent and sovereign internet? In *Power and authority in internet governance*, pages 147–167. Routledge, 2021.
- [215] Al Jazeera Staff. After internet blackout, iranians take stock. <https://www.aljazeera.com/economy/2019/11/27/after-internet-blackout-iranians-take-stock>, 2019. Al Jazeera, November 27, 2019.
- [216] Statista. Average time spent per day on select social media platforms in the United States in 2023. <https://www.statista.com/statistics/1301075/us-daily-time-spent-social-media-platforms/>.
- [217] Zhen Sun and Vitaly Shmatikov. Telepath: A minecraft-based covert communication system. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 2223–2237. IEEE, 2023.
- [218] Zhen Sun and Vitaly Shmatikov. Differential degradation vulnerabilities in censorship circumvention systems. *arXiv preprint arXiv:2409.06247*, 2024.

- [219] Ram Sundara Raman, Prerana Shenoy, Katharina Kohls, and Roya Ensafi. Censored planet: An internet-wide, longitudinal censorship observatory. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 49–66, 2020.
- [220] Qiuqing Tai. China’s media censorship: A dynamic and diversified regime. *Journal of East Asian Studies*, 14(2):185–210, 2014.
- [221] The Tor Project. Tor bridges. <https://tb-manual.torproject.org/bridges/>, 2024. Accessed: 2025-06-05.
- [222] Tor Project. lyrebird - the obfourscator. <https://gitlab.torproject.org/tpo/anti-censorship/pluggable-transport/lyrebird>, 2012. Accessed: 2025-05-27.
- [223] Elisa Tsai, Deepak Kumar, Ram Sundara Raman, Gavin Li, Yael Eiger, and Roya Ensafi. Certainty: Detecting dns manipulation using tls certificates. In *Privacy Enhancing Technologies Symposium (PETS)*, 2023.
- [224] Michael Carl Tschantz, Sadia Afroz, Anonymous, and Vern Paxson. SoK: Towards grounding censorship circumvention in empiricism. In *Symposium on Security & Privacy*. IEEE, 2016.
- [225] Michael Carl Tschantz, Sadia Afroz, Vern Paxson, et al. Sok: Towards grounding censorship circumvention in empiricism. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 914–933. IEEE, 2016.
- [226] Zeynep Tufekci. After the protests. <https://www.nytimes.com/2014/03/20/opinion/after-the-protests.html>, 2014.
- [227] Lindsey Tulloch and Ian Goldberg. Lox: Protecting the social graph in bridge distribution. *Proceedings on Privacy Enhancing Technologies*, 2023.
- [228] James M Turner. The keyed-hash message authentication code (hmac). *Federal Information Processing Standards Publication*, 198(1):1–13, 2008.
- [229] Nirvan Tyagi, Yossi Gilad, Derek Leung, Matei Zaharia, and Nickolai Zeldovich. Stadium: A distributed metadata-private messaging system. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 423–440, 2017.
- [230] Nirvan Tyagi, Yossi Gilad, Derek Leung, Matei Zaharia, and Nickolai Zeldovich. Stadium: A distributed metadata-private messaging system. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 423–440, 2017.

- [231] Zeya Umayya, Dhruv Malik, Devashish Gosain, and Piyush Kumar Sharma. Ptpperf: On the performance evaluation of tor pluggable transports. In *Proceedings of the 2023 ACM on Internet Measurement Conference*, page 501–525, 2023.
- [232] Project V. V2ray — project v. <https://www.v2ray.com/en/>, 2025.
- [233] Adithya Vadapalli, Kyle Storrier, and Ryan Henry. Sabre: Sender-anonymous messaging with fast audits. In *Proceedings of the 2022 IEEE Symposium on Security and Privacy*, pages 1953–1970, 2022.
- [234] Amin Vahdat, David Becker, et al. Epidemic routing for partially connected ad hoc networks. *Tech. Report CS-200006, Duke University*, 2000.
- [235] Jelle Van Den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152, 2015.
- [236] Jelle Van Den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152, 2015.
- [237] Benjamin VanderSloot, Sergey Frolov, Jack Wampler, Sze Chuen Tan, Irv Simpson, Michalis Kallitsis, J Alex Halderman, Nikita Borisov, and Eric Wustrow. Running refraction networking for real. *Proceedings on Privacy Enhancing Technologies*, 4:321–335, 2020.
- [238] Karl Vick. Inside the Clandestine Efforts to Smuggle Starlink Internet Into Iran. <https://time.com/6249365/iran-elon-musk-starlink-protests/>, 2023.
- [239] Afonso Vilalonga, João S Resende, and Henrique Domingos. Torkameleon: Improving tor’s censorship resistance with k-anonymization and media-based covert channels. In *2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1490–1495. IEEE, 2023.
- [240] Christian von der Weth and Manfred Hauswirth. DOBBS: Towards a comprehensive dataset to study the browsing behavior of online users. In *Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, 2013.
- [241] Ryan Wails, Rob Jansen, Aaron Johnson, and Micah Sherr. Proteus: Programmable protocols for censorship circumvention. *Free and Open Communications on the Internet*, 2023.

- [242] Ryan Wails, Rob Jansen, Aaron Johnson, and Micah Sherr. Censorship evasion with unidentified protocol generation. In *Proceedings of the USENIX Security Symposium*, 2025.
- [243] Ryan Wails, Andrew Stange, Eliana Troper, Aylin Caliskan, Roger Dingledine, Rob Jansen, and Micah Sherr. Learning to behave: Improving covert channel security with behavior-based designs. *Proceedings on Privacy Enhancing Technologies*, 2022.
- [244] Ryan Wails, Andrew Stange, Eliana Troper, Aylin Caliskan, Roger Dingledine, Rob Jansen, and Micah Sherr. Learning to behave: Improving covert channel security with behavior-based designs. *Proceedings on Privacy Enhancing Technologies*, 2022.
- [245] Jane Wakefield. Russia 'successfully tests' its unplugged internet. <https://www.bbc.com/news/technology-50902496>, 2019. BBC News, December 24, 2019.
- [246] Andy Wang. Cloak: An anti-censorship tool via pluggable transport. <https://github.com/cbeuw/Cloak>, 2017. Accessed: 2025-06-04.
- [247] Liang Wang, Kevin Dyer, Aditya Akella, Thomas Ristenpart, and Thomas Shrimpton. Seeing through network-protocol obfuscation. In *Proceedings of the 22nd ACM Conference on Computer and Communications Security*, pages 57–69, Denver, CO, USA, October 2015.
- [248] Qiyang Wang, Xun Gong, Giang T.K. Nguyen, Amir Houmansadr, and Nikita Borisov. CensorSpoofer: Asymmetric communication using IP spoofing for censorship-resistant web browsing. In *Proceedings of the 19th ACM Conference on Computer and Communications Security*, pages 121–132, Raleigh, NC, USA, October 2012.
- [249] Qiyang Wang, Zi Lin, Nikita Borisov, and Nicholas Hopper. rbridge: User reputation based tor bridge distribution with privacy preservation. In *NDSS*, 2013.
- [250] Duncan J Watts and Steven H Strogatz. Collective dynamics of ‘small-world’ networks. *Nature*, 393(6684):440–442, 1998.
- [251] Minghui Wei. Domain shadowing: Leveraging content delivery networks for robust {Blocking-Resistant} communications. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3327–3343, 2021.
- [252] Zachary Weinberg, Diogo Barradas, and Nicolas Christin. Chinese wall or swiss cheese? keyword filtering in the great firewall of china. In *Proceedings of the Web Conference*, page 472–483, 2021.

- [253] Zachary Weinberg, Jeffrey Wang, Vinod Yegneswaran, Linda Briesemeister, Steven Cheung, Frank Wang, and Dan Boneh. Stegotorus: a camouflage proxy for the tor anonymity system. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 109–120, 2012.
- [254] Devin Windelspecht. What journalists need to know to report on protests in iran. <https://ijnet.org/en/story/what-journalists-need-know-report-protests-iran>, 2022.
- [255] Stefan Wojcik and Adam Hughes. Sizing up twitter users. *PEW research center*, 24:1–23, 2019.
- [256] David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. Dissent in numbers: Making strong anonymity scale. In *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation*, pages 179–182, 2012.
- [257] Charles V Wright, Scott E Coull, and Fabian Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *NDSS*, volume 9, 2009.
- [258] Mingshi Wu, Jackson Sippe, Danesh Sivakumar, Jack Burg, Peter Anderson, Xiaokang Wang, Kevin Bock, Amir Houmansadr, Dave Levin, and Eric Wustrow. How the great firewall of china detects and blocks fully encrypted traffic. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 2653–2670, 2023.
- [259] Mingshi Wu, Jackson Sippe, Danesh Sivakumar, Jack Burg, Peter Anderson, Xiaokang Wang, Kevin Bock, Amir Houmansadr, Dave Levin, and Eric Wustrow. How the great firewall of china detects and blocks fully encrypted traffic. In *Proceedings of the 32nd USENIX Security Symposium*, pages 2653–2670, 2023.
- [260] Mingshi Wu, Jackson Sippe, Danesh Sivakumar, Jack Burg, Peter Anderson, Xiaokang Wang, Kevin Bock, Amir Houmansadr, Dave Levin, and Eric Wustrow. How the Great Firewall of China detects and blocks fully encrypted traffic. In *USENIX Security Symposium*, 2023.
- [261] Mingshi Wu, Ali Zohaib, Zakir Durumeric, Amir Houmansadr, and Eric Wustrow. A Wall Behind A Wall: Emerging Regional Censorship in China . In *Proceedings of the 2025 IEEE Symposium on Security and Privacy*, pages 1363–1380, 2025.
- [262] Eric Wustrow, Scott Wolchok, Ian Goldberg, and J. Halderman. Telex: Anticensorship in the network infrastructure. In *Proceedings of the 20th USENIX Security Symposium*, pages 459–474, San Francisco, CA, USA, August 2011.

- [263] Bin Xiao, Wei Chen, Yanxiang He, and EH-M Sha. An active detecting method against syn flooding attack. In *11th International Conference on Parallel and Distributed Systems (ICPADS'05)*, volume 1, pages 709–715. IEEE, 2005.
- [264] Ruohan Xiong and Jeffrey Knockel. An efficient method to determine which combination of keywords triggered automatic filtering of a message. In *Proceedings of the 9th USENIX Workshop on Free and Open Communications on the Internet*, Santa Clara, CA, USA, 2019.
- [265] Xueyang Xu, Z Morley Mao, and J Alex Halderman. Internet censorship in china: Where does the filtering occur? In *Proceedings of the 12th International Conference on Passive and Active Network Measurement*, pages 133–142, Vienna, Austria, 2011.
- [266] Diwen Xue, Anna Ablove, Reethika Ramesh, Grace Kwak Danciu, and Roya Ensafi. Bridging barriers: A survey of challenges and priorities in the censorship circumvention landscape. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 2671–2688, 2024.
- [267] Diwen Xue, Anna Ablove, Reethika Ramesh, Grace Kwak Danciu, and Roya Ensafi. Bridging barriers: A survey of challenges and priorities in the censorship circumvention landscape. In *Proceedings of the 33rd USENIX Security Symposium*, 2024.
- [268] Diwen Xue, Michalis Kallitsis, Amir Houmansadr, and Roya Ensafi. Fingerprinting obfuscated proxy traffic with encapsulated {TLS} handshakes. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 2689–2706, 2024.
- [269] Diwen Xue, Benjamin Mixon-Baca, ValdikSS, Anna Ablove, Beau Kujath, Jedidiah R Crandall, and Roya Ensafi. Tspu: Russia’s decentralized censorship system. In *Proceedings of the 22nd ACM Internet Measurement Conference*, pages 179–194, 2022.
- [270] Diwen Xue, Reethika Ramesh, ValdikSS, Leonid Evdokimov, Andrey Viktorov, Arham Jain, Eric Wustrow, Simone Basso, and Roya Ensafi. Throttling Twitter: an emerging censorship technique in Russia. In *Internet Measurement Conference*, 2021.
- [271] Diwen Xue, Reethika Ramesh, ValdikSS, Leonid Evdokimov, Andrey Viktorov, Arham Jain, Eric Wustrow, Simone Basso, and Roya Ensafi. Throttling twitter: An emerging censorship technique in russia. In *ACM Internet Measurement Conference (IMC)*, 2021.

- [272] Mehdi Yahyanejad. Methods and systems for digital data transmission through satellite TV channels, U.S. Patent 10 448 116, 2019.
- [273] Altug Yalcintas and Naseraddin Alizadeh. Digital protectionism and national planning in the age of the internet: the case of iran. *Journal of Institutional Economics*, 16(4):519–536, 2020.
- [274] Wang Yun. Is Apple’s AirDrop safe to use in China? <https://www.rfa.org/english/news/china/china-apple-airdrop-01152024112633.html>.
- [275] Bassam Zantout, Ramzi Haraty, et al. I2P data communication system. In *Proceedings of the 1st ACM SIGCOMM Workshop on Information-Centric Networking*, pages 401–409, 2011.
- [276] Ennan Zhai, David Isaac Wolinsky, Ruichuan Chen, Ewa Syta, Chao Teng, and Bryan Ford. AnonRep: Towards Tracking-Resistant Anonymous Reputation. 2016.
- [277] Wenxuan Zhou, Amir Houmansadr, Matthew Caesar, and Nikita Borisov. Sweet: Serving the web by exploiting email tunnels. In *Proceedings of the 6th Workshop on Hot Topics in Privacy Enhancing Technologies*, Bloomington, IN, USA, 2013.
- [278] Wanzheng Zhu, Hongyu Gong, Rohan Bansal, Zachary Weinberg, Nicolas Christin, Giulia Fanti, and Suma Bhat. Self-supervised euphemism detection and identification for content moderation. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 229–246. IEEE, 2021.
- [279] Danford Zirugo and Admire Mare. Assessing twitter’s revolutionary potential in an authoritarian regime. *Digital Technologies, Elections and Campaigns in Africa*, 2023.
- [280] Hadi Zolfaghari and Amir Houmansadr. Practical censorship evasion leveraging content delivery networks. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1715–1726, 2016.