Inadvertent Censorship-Resistance

Cooper Glavin
Georgetown University
Washington, DC

Micah Sherr
Georgetown University
Washington, DC

ABSTRACT

There is an active arms-race between privacy advocates and nation-state censors, with the former developing and refining *censorship resistance systems* (CRSes) to bypass censorship while the latter look for new ways to detect and block their use. Improving and developing new CRSes is a critical undertaking towards ensuring an open and accessible Internet.

This short paper, however, takes a different and complimentary tack. Instead of exclusively developing systems that are explicitly designed to bypass censorship, we advocate for the development of non-CRS communication platforms that are generally useful outside the context of censorship-resistance yet have the property that they are also difficult to block. Our intention is to avoid the CRS "classification problem" (*can the censor detect that an observed flow is an instance of a CRS protocol?*) and instead promote the widespread adoption of a communication platform that offers desirable performance and security properties with censorship-resistance as a side-effect. We outline desirable traits of a communication platform within this space, present a case study of how libp2p nearly meets our goals, open a discussion about the limitations of this approach, and outline future steps towards achieving inadvertent censorship-resistance.

KEYWORDS

censorship, censorship resistance

1 INTRODUCTION

The Internet is the foremost medium for education, political expression, and economic development. Having control over the Internet landscape thus provides nation-states with political, social, and economic benefits. Many nation-states are adopting increasingly sophisticated censorship systems [15, 27, 32] to exert this control, forcing privacy advocates to play reactively and develop new censorship-resistant systems (CRSes) that allow unfettered access to the Internet.

Current CRS approaches generally fall into four categories: mimicry [26] (pretending to be an allowed protocol), tunneling (embedding a covert channel within an actual allowed overt channel) [17, 36], random (randomizing traffic flows to thwart protocol classification), and refraction networking [16, 24, 42] (redirection of traffic via in-network routers). An important invariant among all approaches is that a *covert* communication channel is hidden within an innocous-looking (to the censor) *overt* channel. A censor's ability to detect and block a CRS reduces to a classification problem: can it distinguish non-obfuscated flows from those which contain a covert channel [37, 38, 41]? This has led to an active arms-race between privacy advocates who develop new methods of CRSes and censors who seek to detect and block them.

While we advocate for the continued development of new privacy-enhancing technologies that permit free access to the Internet, in this short paper, we propose a radically different and complementary approach. We argue for development in technologies that are *not* developed primarily for censorship-circumvention. Rather, we focus on communication platforms that provide strong performance, usability, and security benefits, while bringing censorship-resistance as a side-effect. Similar to the position paper by Tan and Sherr [33], we argue that to maximize collateral damage and make it untenable for a censor to block a protocol, that protocol should not be used primarily for censorship-resistance.

There is a subtle yet important distinction between what we term in this paper *inadvertent censorship-resistance* and the development of technologies that focus exclusively on censorship-resistance. In most prior work, the goal is to establish traffic flows that carry covert information but appear innocuous to a censor. When this desired indistinguishability fails to be achieved, censors can identify the use of CRSes and prevent their use [37, 38, 41]. In inadvertent censorship-resistance, our goal is instead to develop commonly used protocols that are used mostly to access allowed resources but, due to their security and robustness properties, are difficult for a censor to selectively block when they are used to access disallowed content. We take inspiration from the development and growth of HTTPS, which was not developed for censorship-resistance, but clearly makes a censor's job more difficult since it can no longer observe content.[1] Because HTTPS is so ubiquitous and relatively few sites are actually disallowed by censors [39], blocking HTTPS *en masse* is rarely done, presumably because it carries enormous collateral damage.

This paper outlines the properties of a communication platform that, if widely adopted, could lead to inadvertent censorship-resistance. It is vital that such a platform bring advantages like speed, security, and durability in order for it to gain popularity. A solution in this space would leverage desirable traits for communication, creating a situation in which the opportunity cost of banning its use outright is too high for a censor (similar to HTTPS). In particular, we focus on communication platforms that achieve desirable traits for network communication such as end-to-end encryption, decentralization, and the ability to route around network failures via network overlays. These traits are important for a communication platform to gain widespread adoption, bringing utility beyond existing alternatives.

We are looking to enable the development of popular applications on top of such communication platforms that use decentralization and network-fault tolerance to make discrimination via destination more difficult. By design, our vision of inadvertent censorship-resistance does *not* attempt to conceal the applications

[1] This is an oversimplification, since TLS headers and extensions (notably, Server Name Indication (SNI) [12]) can leak important meta-data. Still, we argue that end-to-end encryption of content makes it more difficult for a censor to perform content-based (rather than site-based) blocking.

being used by the communicating parties. Our threat model allows a censor to determine that our communication framework is being used and which particular application is in use (e.g., a file transfer application vs. a video streaming application). Because the communication platform itself is (inadvertently) not amenable to censorship, the applications that make use of it should *not* be used primarily for censorship-resistance, since such dedicated CRSes would be unnecessary. Hence, blocking a popular application that uses the communication platform would (hopefully) incur too high collateral damage since it is not used primarily to access forbidden content.

In what follows, we describe the desirable communication properties of a communication platform that could provide inadvertent censorship-resistance. Perhaps surprisingly, we find that one already-deployed platform, libp2p [29], meets many of our design goals. We describe how its increased adoption (leading to a higher cost of blocking it *in toto*) could lead to communication that is more robust against censorship, as well as discussing its shortcomings and future steps that would be needed to better achieve inadvertent censorship-resistance.

2 BACKGROUND AND RELATED WORK

As with previous work [25, 34], we assume a threat model in which a *censor* controls a *sphere of influence* in which it can observe all traffic. The censor may choose to block or allow individual packets or flows into or out of its sphere of influence, but must abide by standard cryptographic assumptions—in particular, it cannot obtain plaintext without knowledge of the corresponding decryption key. *Censored users* (or, in what follows, simply *users*) reside within the censor's sphere of influence and are subject to censorship. The censor may also operate outside of its sphere of influence, for example, by operating a VPS node on a foreign cloud. However, the censor cannot observe traffic outside of its sphere of influence unless it traverses nodes under its control.

As noted above, existing CRSes generally (but not always) apply one of four high-level approaches: mimicry, tunneling, appearing as random traffic, or refraction networking. Mimicry is now generally dismissed as inherently weak due to the difficulty of perfectly mimicking the particularities of implementations of the overt protocol [21]. Tunneling avoids these challenges by layering the covert channel within an actual implementation of the overt protocol, and is the basis for widely deployed censorship circumvention protocols such as Snowflake [14] that is used by Tor [11]. However, Wails et al. [36] show that behavioral differences between how the covert and overt protocols are commonly used can leak information. Shadowsocks [31], Scramblesuit [40], and obfs4 [4] attempt to fall into the "long tail" of traffic flows by morphing covert channels into streams that look entirely random. However, Wang et al. [38], Wu et al. [41], and most recently Wails et al. [37] demonstrate that censors can apply simple detection rules to detect these high-entropy flows with high accuracy. Finally, refraction network is an exciting approach, but requires in-network participation, which to date has been scant, and can be expensive to operate [35].

The idea of inadvertent censorship-resistance was originally proposed by Tan and Sherr in a short position paper [33]. Tan and Sherr advocated for achieving censorship-resistance as a "characteristic of a widely fielded and general-purpose communication platform" [33], and suggested the use of an encrypted key-value store as a possible instantiation of such a platform. We also promote the development of difficult-to-censor protocols that are not developed for censorship resistance *per se*, and extend Tan and Sherr's position paper in three important respects: (1) we describe a set of properties that a communication platform should exhibit to achieve inadvertent censorship-resistance, (2) we perform a case study of an existing platform, libp2p, that we believe could achieve our goals if it gains more popularity, and (3) we discuss the limitations and challenges of inadvertent censorship-resistance.

3 INADVERTENT CENSORSHIP-RESISTANT PLATFORMS

A CRS that is used solely for its censorship circumvention is inherently perilous: if its flows can be identified, the cost of barring access to the system is low since it serves no purpose other than evasion. In contrast, if a communication system has widespread adoption due to its utility in business or commerce, then the cost of preventing its use may be too high for the censor, even if that system is sometimes (but relatively rarely) used to access content that would otherwise be blocked.

In this section, we enumerate the characteristics that a communication platform should exhibit for it to be both (1) attractive to developers and thus likely to gain widespread adoption, and (2) difficult to block as a consequence of its performance- and security-driven design goals (i.e., rather than having censorship-resistance as a standalone goal).

An inadvertent censorship-resistant communication platform should have the following properties:

End-to-end confidentiality and authenticity. End-to-end (e2e) security is a standard desirable property of modern communication systems, and debates about law enforcement access to encryption notwithstanding [1], encryption has become the norm on the Internet [19]. To avoid dependence on a centralized PKI (see below), the communication platform should use a self-authenticating addressing scheme (i.e., addresses encapsulate nodes' public keys) to provide strong e2e security.

Organizational decentralization and Openness. Companies may be reluctant to adopt technologies that rely entirely on a single external entity, due to concerns about cost, the longevity of the service provider, and being subject to a provider's (potentially changing) policies. An open communication platform with decentralized or federated governance offers more stability (since no one provider is relied upon) and may better address the needs of the platform's users.

Scalability and Robustness. The communication platform should have the capability of growing with its userbase, and should be capable of routing around network failures to enable highly reliable communication channels.

Reachability. The platform should incorporate techniques such as NAT piercing and overlay routing [3] to ensure that any users—regardless of network configuration—are able to communicate with

minimal burden. Nodes should be identified separate from their IP addresses, allowing multiple potential modes of communication.

Elasticity. While a system must be able to scale and handle network stress, it should also be able to scale up and down elastically. Network usage is rarely linear, and a platform should be able to accommodate rapid fluctuations while still being efficient with resource usage.

We emphasize that the above list focuses on utility, performance, and governance properties that we believe would be attractive to application developers, hopefully leading to widespread adoption. Purposefully, we do not specify anonymity or censorship-resistance properties as primary design goals, as our hope is that an open communication platform that achieves the above will also, by construction, be inadvertently censorship-resistant.

In particular, two of our desirable properties can often be incompatible with surveillance and censorship: *e2e encryption* thwarts a censor's ability to directly observe content, forcing it to use more error-prone inference techniques (e.g., website fingerprinting [8, 9]); and *reachability* makes blocking by destination address more difficult if the communication platform supports overlay (i.e., multihop application-layer) message delivery as a means to bypass network failures. In a nutshell, our hypothesis is that as communication platforms become more secure, resilient, scalable, and decentralized, they may also become less amenable to supporting censorship. Of course, a nation-state censor may opt to block the platform in its entirety, but it then risks "falling behind" as more applications and users world-wide switch to the platform due to its utility and performance advantages.

We admittedly take an optimistic stance, but are buoyed by the recent increase in interest in developing robust and scalable peer-to-peer communication platforms, some of which meet the properties outlined above. In the next section, we perform a case study of one platform, libp2p [29], that is gaining maturity. libp2p brings unique features such as strong reachability that make it appealing to developers, while also achieving some censorship-resistance properties as a side-effect of its design.

4 CASE STUDY: LIBP2P

libp2p is a framework for developing peer-to-peer (p2p) applications. It provides a modular network stack that allows the application programmer to pick the tools that best match their needs. Exhibiting many of the desirable traits outlined in § 3, libp2p already has existing applications built on it such as InterPlanetary File System (IPFS) [5], a p2p content delivery tool, and Berty [6], a p2p messaging app. As an added benefit, these applications maintain a consistent user-base, with the IPFS network reporting approximately 39,000 online servers and 258,000 unique clients observed from 6 August 2023 to 12 November 2023 [28].

libp2p communication is composed of bidirectional channels with modular support for different transports, encryption schemes, multiplexers, and NAT traversal. As a p2p protocol, libp2p is decentralized, with the only centralized infrastructure coming from bootstrap nodes to allow new nodes to populate their hash tables; however, this is not required and alternative discovery protocols that avoid bootstrap nodes such as mDNS and random-walk are also implemented. libp2p is extensible and its modular design allows new extensions to be built into the networking stack.

Allowing nodes to be identified by peerID-IP pairs, libp2p uses a multi-addressing scheme that also identifies the transport to be used. When a node joins the network, it creates a list of multi-addresses through which it is reachable, allowing it to have many different methods of being reached (e.g., via QUIC or TCP/TLS). libp2p's multi-addressing scheme provides robustness against network failures: nodes can be reached via multiple channels so even if one channel fails, a node is still typically reachable. Additionally, libp2p supports relaying—that is, forwarding messages via an intermediary node on the p2p network—to route around network failures.

While traditional client-server applications work well with NATs (since in most cases, it is a client behind the NAT that initiates a connection to the not-behind-a-NAT server), p2p communication is often hampered by reachability challenges since both communicating parties may be behind NATs. libp2p implements a NAT hole-punching protocol for enabling p2p communication even between NATted hosts, allowing most users to connect to the network and run a fully-functioning node regardless of network configuration.[2] libp2p allows new nodes to discover their public IP, identify their NAT configuration, and then automatically create reservations for a relay system that allows the node to upgrade to a direct connection with a peer. Seemann et al. report that this works for most clients, having an 86% success rate for TCP and a 93% success rate using QUIC [30].

Towards inadvertent censorship-resistance with libp2p. libp2p requires encrypted connections between peers. Its multi-addressing scheme and modular transport mechanism provide flexibility as to which specific encrypted transport (e.g., TLS, WebRTC, QUIC, etc.) is used. The use of common e2e-encrypted transports makes it more difficult for a censor to perform content-based blocking, while also providing some protections against easily identifying the use of libp2p.[3]

libp2p's decentralized architecture protects against destination-based blocking. When direct communication between peers is not achievable (for instance, due to a censorship event), libp2p supports relaying the connection via another peer. Since communication is encrypted between peers, this effectively constitutes a proxy service, where an intermediary node located outside the censor's sphere of influence can act as a middle-man. This does not provide as robust anonymity properties as networks designed specifically for anonymity (e.g., Tor [11] and Nym [10]) since the intermediary learns the network locations of both the initiator and destination peers. Still, the large number of potential intermediary peers makes it difficult for a censor to prevent such relaying. Additionally, since relaying is a "standard" feature of libp2p to both route around network failures and enable p2p networking when a peer is located

[2]Which clients can receiving incoming connections is fairly complex and is dependent on their particular NAT configuration (e.g., symmetric vs. one-to-one, etc.).

[3]Disguising the use of the platform is *not* one of our design goals as we are expressly reliant on the popularity of the platform to achieve inadvertent censorship resistance. However, the use of common transports (e.g., WebRTC) may provide some benefit before the platform gains ubiquity since it may frustrate a censor's ability to detect and block it.

behind a NAT, its use is not necessarily indicative of an attempt to counter censorship.

Shortcomings. Although libp2p offers many properties of an inadvertent censorship-resistant system, it has several important limitations. The framework needs additional time and resources to gain widespread adoption by developers. Although the platform currently offers API support for multiple programming languages, not all features are available for all languages. One feature that is crucial for application development is increased support for browser-based connectivity, specifically WebTransport [2] and WebRTC implementations across a variety of languages.

Additionally, libp2p still largely remains coupled to bootstrap nodes to populate DHT tables—this infrastructure can lead to potential points of failure as applications scale.

Censors have historically shown willingness to accept collateral damage to ban censorship-evasion approaches [23, 41]. Consequently, libp2p (and more generally, inadvertent censorship resistant systems) is entirely dependent on a useful application being developed that gains enough traction to be allowed past censorship systems. Until it gains ubiquity, blocking the libp2p network in its entirety is likely feasible without incurring too high collateral damage.

5 DISCUSSION AND FUTURE STEPS

Blocking based on content or destination is difficult in an inadvertent censorship resistant communication platform due to the use of peer-to-peer encryption and support for multi-hop routing. However, censors are not limited to censoring based on IP addresses and plaintext message content, and can apply myriad other techniques to perform blocking. In particular, the properties we outline in § 3 omit traffic padding or other techniques that might mitigate traffic fingerprinting attacks. Despite both performance and privacy improvements in padding techniques [18, 22], such defenses still incur substantial communication overhead. Since they are primarily aimed at defeating website fingerprinting in the context of Tor or some other anonymity service, it is unlikely that adopting padding techniques makes sense for a general-purpose communication platform. In summary, platforms that provide inadvertent censorship resistance are likely still to be vulnerable to traffic fingerprinting, which could lead to content-based filtering.

However, given that few deployed CRSes and anonymity systems use padding, we suspect the accuracy of traffic fingerprinting is far too low to avoid significant over-blocking, especially when the base rate of accessing disallowed content is low (as we would expect it to be in an inadvertent censorship resistance communication platform). In short, while an inadvertent censorship resistant communication platform may be vulnerable to traffic fingerprinting attacks, censors (at least today) tend not to use such methods, likely because it is too imprecise given the low base rates of attempted censorship circumvention.

It is worth re-highlighting that a key requirement for inadvertent censorship resistance is that the communication platform be popular and used predominantly for reasons not related to censorship resistance. Inadvertent censorship resistance is entirely reliant on the development and use of supported applications. While the platform could be appealing in its own right, until an application developed with it gains widespread adoption, censors can simply block the platform in its entirety.

Moreover, we envision it will be trivial for a censor to identify applications using the communication platform. For example, nodes in libp2p freely report which applications they support. Attempting to hide censorship-resistant applications amongst permitted applications becomes extremely difficult and is contrary to our goals. Instead, we rely on applications being developed that allow the user to access both allowed and (by virtue of the platform's robustness properties) forbidden content, forcing the censor to identify which streams under the supported application should be blocked.

We believe there is a place for both censorship-resistance systems, whose explicit purpose is to evade censorship, *and* inadvertent censorship resistance communication platforms that circumvent censorship via a side effect. By virtue of having a single "disallowed" purpose—that is, to avoid censorship—a censor bears little cost to blocking a CRS entirely, assuming it it can accurately detect its use. However, since CRSes directly target censorship, they are well equipped to engage in the censorship arms-race and develop new techniques for evading censors. In contrast, communication platforms that exhibit inadvertent censorship resistance are more inflexible and rigid, and cannot respond quickly if censors learn how to distinguish communication to disallowed content from communication to allowed content. However, as the communication platform becomes increasingly popular for reasons unrelated to censorship evasion, the base rate of accesses to disallowed content will decrease. This will likely make it more difficult for a censor that must perform traffic fingerprinting to main high precision.

With that in mind, we posit that inadvertent censorship resistance is an important direction that is worth further study. It is difficult to know how a nation-state censor would respond to a communication platform that clearly is not aimed at censorship resistance, but nevertheless is difficult to surveil. That China blocked encrypted SNI early in its design and deployment processes [7] and blocks most encrypted DNS (DoH/DoT) traffic [20] provides some indication that protocols designed explicitly to hinder censorship will be quickly blocked.[4] On the other hand, protocols that provide security (e.g., HTTPS) and performance benefits (e.g., QUIC) are mostly (but not always [13]) allowed, likely due to their ubiquity and high collateral cost of blocking.

How to best develop a communication platform that provides inadvertent censorship resistance is an open question. Having censorship-resistance as a design goal makes such a system non-inadvertent and may lead to it being mostly used as a censorship-resistance technology. We believe that communication platforms that organically offer censorship-resistance, as a side-effect of its design, are more likely to be successful, both as a popular mechanism for communicating and as a means to support unfettered access to information online. We argue that supporting the development of existing efforts such as libp2p should be a priority for privacy advocates. Such platforms have the *capability* of achieving inadvertent censorship resistance, but as of yet, lack the critical mass both in terms of applications and users that would make it a platform that is untenable for a censor to block.

[4] CRSes are more difficult to block because they attempt to appear as legitimate or unknown communication channels.

REFERENCES

[1] Harold Abelson, Ross Anderson, Steven M Bellovin, Josh Benaloh, Matt Blaze, Whitfield" Whit" Diffie, John Gilmore, Matthew Green, Susan Landau, Peter G Neumann, et al. 2015. Keys under doormats. *Commun. ACM* 58, 10 (2015), 24–26.

[2] Bernard Adoba, Nidhi Jaju, and Victor Vasiliev. 2023. *WebTransport.* Working Draft. World Wide Web Consortium (W3C).

[3] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. 2001. Resilient overlay networks. In *ACM Symposium on Operating Systems Principles (SOSP).*

[4] Yawning Angel. 2023. obfs4: The obfourscator. https://gitlab.com/yawning/obfs4.

[5] Juan Benet. 2014. IPFS - Content Addressed, Versioned, P2P File System. *arXiv preprint arXiv:1407.3561* (2014). Available at https://arxiv.org/abs/1407.3561..

[6] Berty NGO. 2023. Berty: Unstoppable P2P Communication. https://berty.tech/.

[7] Kevin Bock, iyouport, Anonymous, Louis-Henri Merino, David Fifield, Amir Houmansadr, and Dave Levin. 2020. *Exposing and Circumventing China's Censorship of ESNI.* Technical Report. Great Firewall Report. Available at https://gfw.report/blog/gfw_esni_blocking/en/.

[8] Giovanni Cherubin, Rob Jansen, and Carmela Troncoso. 2022. Online Website Fingerprinting: Evaluating Website Fingerprinting Attacks on Tor in the Real World. In *USENIX Security Symposium (USENIX Security).*

[9] Weiqi Cui, Tao Chen, Christian Fields, Julianna Chen, Anthony Sierra, and Eric Chan-Tin. 2019. Revisiting Assumptions for Website Fingerprinting Attacks. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS).*

[10] Claudia Diaz, Harry Halpin, and Aggelos Kiayias. 2021. The Nym Network. https://lirias.kuleuven.be/3751283?limo=0.

[11] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium (USENIX).*

[12] D. Eastlake. 2011. *Transport Layer Security (TLS) Extensions: Extension Definitions.* RFC 6066. Internet Engineering Task Force (IETF).

[13] Kathrin Elmenhorst. 2022. *A Quick Look at QUIC Censorship.* Technical Report. Open Technology Fund (blog post). Available at https://www.opentech.fund/news/a-quick-look-at-quic-censorship/..

[14] David Fifield. 2017. *Threat modeling and circumvention of Internet censorship.* Ph.D. University of California, Berkeley.

[15] Arturo Filasto and Jacob Appelbaum. 2012. OONI: Open Observatory of Network Interference. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI).*

[16] Sergey Frolov, Jack Wampler, Sze Chuen Tan, J. Alex Halderman, Nikita Borisov, and Eric Wustrow. 2019. Conjure: Summoning Proxies from Unused Address Space. In *ACM SIGSAC Conference on Computer and Communications Security (CCS).*

[17] Sergey Frolov and Eric Wustrow. 2020. HTTPT: A Probe-Resistant Proxy. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI 20).*

[18] Jiajun Gong and Tao Wang. 2020. Zero-delay Lightweight Defenses against Website Fingerprinting. In *USENIX Security Symposium (USENIX Security).*

[19] Google. 2023. HTTPS Encryption on the Web. https://transparencyreport.google.com/https/overview?hl=en.

[20] Nguyen Phong Hoang, Michalis Polychronakis, and Phillipa Gill. 2022. Measuring the Accessibility of Domain Name Encryption and Its Impact on Internet Filtering. In *International Conference on Passive and Active Network Measurement (PAM).*

[21] Amir Houmansadr, Chad Brubaker, and Vitaly Shmatikov. 2013. The Parrot is Dead: Observing Unobservable Network Communications. In *IEEE Symposium on Security and Privacy (Oakland).*

[22] Marc Juarez, Mohsen Imani, Mike Perry, Claudia Diaz, and Matthew Wright. 2016. In *Computer Security (ESORICS)*, Ioannis Askoxylakis, Sotiris Ioannidis, Sokratis Katsikas, and Catherine Meadows (Eds.).

[23] Simin Kargar and Keith McManamen. 2018. Censorship and collateral damage: Analyzing the Telegram ban in Iran. *Berkman Klein Center Research Publication* 2018-4 (2018).

[24] Josh Karlin, Daniel Ellard, Alden W. Jackson, Christine E. Jones, Greg Lauer, David P. Mankins, and W. Timothy Strayer. 2011. Decoy Routing: Toward Unblockable Internet Communication. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI).*

[25] Sheharbano Khattak, Tariq Elahi, Laurent Simon, Colleen M. Swanson, Steven J. Murdoch, and Ian Goldberg. 2016. SoK: Making Sense of Censorship Resistance Systems. *Proceedings on Privacy Enhancing Technologies (PoPETS)* 2016, 4 (July 2016), 37–61.

[26] Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. 2012. SkypeMorph: Protocol Obfuscation for Tor Bridges. In *ACM Conference on Computer and Communications Security (CCS).*

[27] Arian Akhavan Niaki, Shinyoung Cho, Zachary Weinberg, Nguyen Phong Hoang, Abbas Razaghpanah, Nicolas Christin, and Phillipa Gill. 2020. ICLab: A Global, Longitudinal Internet Censorship Measurement Platform. In *IEEE Symposium on Security and Privacy (SP).*

[28] Protocol Benchmarking & Optimization Team (ProbeLab). 2023. IPFS DHT. https://probelab.io/ipfsdht/#ipfs-servers-vs-clients-plot.

[29] Protocol Labs. 2023. libp2p. https://libp2p.io/.

[30] Marten Seemann, Max Inden, and Dimitris Vyzovitis. 2022. Decentralized Hole Punching. In *IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW).*

[31] Shadowsocks Contributors. 2023. Shadowsocks: A fast tunnel proxy that helps you bypass firewalls. https://shadowsocks.org/.

[32] Ram Sundara Raman, Prerana Shenoy, Katharina Kohls, and Roya Ensafi. 2020. Censored Planet: An Internet-wide, Longitudinal Censorship Observatory. In *ACM SIGSAC Conference on Computer and Communications Security (CCS).*

[33] Henry Tan and Micah Sherr. 2014. Censorship Resistance as a Side-Effect. In *International Workshop on Security Protocols.*

[34] Michael Carl Tschantz, Sadia Afroz, Anonymous, and Vern Paxson. 2016. SoK: Towards Grounding Censorship Circumvention in Empiricism. In *IEEE Symposium on Security and Privacy (Oakland).*

[35] Benjamin VanderSloot, Sergey Frolov, Jack Wampler, Sze Chuen Tan, Irv Simpson, Michalis Kallitsis, J. Alex Halderman, Nikita Borisov, and Eric Wustrow. 2020. Running Refraction Networking for Real. *Proceedings on Privacy Enhancing Technologies* 2020, 4 (Aug. 2020). https://doi.org/10.2478/popets-2020-0075

[36] Ryan Wails, Andrew Stange, Eliana Troper, Aylin Caliskan, Roger Dingledine, Rob Jansen, and Micah Sherr. 2022. Learning to Behave: Improving Covert Channel Security with Behavior-Based Designs. *Proceedings on Privacy Enhancing Technologies* 2022, 3 (July 2022), 179–199.

[37] Ryan Wails, George Arnold Sullivan, Micah Sherr, and Rob Jansen. 2024. On Precisely Detecting Censorship Circumvention in Real-World Networks. In *Network and Distributed System Security Symposium (NDSS).*

[38] Liang Wang, Kevin P Dyer, Aditya Akella, Thomas Ristenpart, and Thomas Shrimpton. 2015. Seeing through Network-Protocol Obfuscation. In *ACM Conference on Computer and Communications Security (CCS).*

[39] Zachary Weinberg, Mahmood Sharif, Janos Szurdi, and Nicolas Christin. 2017. Topics of Controversy: An Empirical Analysis of Web Censorship Lists. *Proceedings on Privacy Enhancing Technologies (PoPETS)* 1 (July 2017), 42–61.

[40] Philipp Winter, Tobias Pulls, and Juergen Fuss. 2013. ScrambleSuit: A Polymorphic Network Protocol to Circumvent Censorship. In *ACM Workshop on Privacy in the Electronic Society (WPES).*

[41] Mingshi Wu, Jackson Sippe, Danesh Sivakumar, Jack Burg, Peter Anderson, Xiaokang Wang, Kevin Bock, Amir Houmansadr, Dave Levin, and Eric Wustrow. 2023. How the Great Firewall of China Detects and Blocks Fully Encrypted Traffic. In *USENIX Security.*

[42] Eric Wustrow, Scott Wolchok, Ian Goldberg, and J. Alex Halderman. 2011. Telex: Anticensorship in the Network Infrastructure. In *USENIX Security Symposium (USENIX).*