Towards more Effective Censorship Resistance Systems

by

Mohammad Tariq Elahi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2015

© Mohammad Tariq Elahi 2015

Some rights reserved.

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

The content in Chapter 2 was co-authored with Colleen Swanson, who provided the discussion of the global adversary and the failed limited sphere of influence assumption. John Doucette provided the framework and preliminary analysis of the simple censor cases, Steven Murdoch provided the real world problem motivation and the simulator and its description, and Hadi Hosseini provided a portion of the related work discussion in Chapter 3. George Danezis provided a sketch and core code for the two *PrivEx* variants in Chapter 4. Kevin Bauer provided the Tor simulation patch and scripts while Mashael AlSabah provided the related work discussion in Chapter 5. The rest of this thesis contains original content and was authored under the supervision of Ian Goldberg.

Abstract

Internet censorship resistance systems (CRSs) have so far been designed in an ad-hoc manner. The fundamentals are unclear and the foundations are shaky. Censors are, more and more, able to take advantage of this situation. Future censorship resistance systems ought to be built from strong theoretical underpinnings and be based on empirical evidence.

Our approach is based on systematizing the CRS field and its players. Informed by this systematization we develop frameworks that have broad scope, from which we gain general insight as well as answers to specific questions. We develop theoretical and simulation-based analysis tools 1) for learning how to manipulate censor behavior using game-theoretic tactics, 2) for learning about CRS-client activity levels on CRS networks, and finally 3) for evaluating security parameters in CRS designs.

We learn that there are gaps in the CRS designer's arsenal: certain censor attacks go unmitigated and the dynamics of the censorship arms race are not modeled. Our game-theoretic analysis highlights how managing the base rate of CRS traffic can cause stable equilibriums where the censor allows some amount of CRS communication to occur. We design and deploy a privacy-preserving data gathering tool, and use it to collect statistics to help answer questions about the prevalence of CRS-related traffic in actual CRS communication networks. Finally, our security evaluation of a popular CRS exposes suboptimal settings, which have since been optimized according to our recommendations.

All of these contributions help support the thesis that more formal and empirically driven CRS designs can have better outcomes than the current state of the art.

Acknowledgements

الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِين

These too few and inadequate words on this page can not convey the gratitude that I have for the many mentors and supporters who have made possible this work, nor can the sum of their influence and impact on myself and where I now stand be bound between these covers; and yet, to thank them I must try.

Foremost is Ian Goldberg, my supervisor. I am truly indebted to him for his generosity; from the freedom to take my time to explore and pick my topic, to the copious funding that allowed me to focus on research, to all the time that I stole away by walking into his always-open office with questions, and to the always collegial manner in which he shared his knowledge and helped me work towards answers.

I am grateful to Nick Hopper for agreeing to be my external examiner and for spearheading the many areas of Internet censorship resistance that make it an interesting and active field. I am grateful also to Srinivasan Keshav and Bernard Wong for their invaluable critique and feedback of my research proposal and along with Mahesh Tripunitara for agreeing to be on my examining committee.

I would like to thank my many co-authors, especially Mashael Alsabah, Kevin Bauer, George Danezis, Roger Dingledine, John Doucette, Hadi Hoseinni, Steven Murdoch, and Colleen Swanson, for the great fun we have had solving hard problems and fighting the good fight. I would also like to call out my CrySP lab mates Erinn Atwater, Cecylia Bocovic, Navid Esfahani, Sarah Harvey, Kevin Henry, Ryan Henry, Atif Khan, Hassan Khan, Nikolas Unger, Jalaj Upadhyay, and certainly not least, Tao Wang, for their gracious mental agility, unconditional emotional support, and overall inclusiveness which made the CrySP lab a home away from home and the Vortex a never-ending ride.

Finally, and with great affection, I want to thank my family—Ammi, Baba, Sanobar, Bilal, and Sundus—for your unyielding faith in me as I embarked on this uncharted journey. My dearest wife Mahvish, I will never forget the sacrifices you have made, the endless support you bolstered me up with, and your wit and humor when I was feeling overwhelmed; you have made all the difference. Little Sophia, the wheels on the bus do go round and round, and may they always turn forever more.

In Memoriam

Mehboob Elahi
(1951–1998)

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Internet is a tool that impacts the lives of hundreds of millions of people around the world. It allows the fluid exchange of information and ideas from disparate corners of the globe, linking individuals together economically, socially, and politically. The ease with which information can be disseminated has been a boon to successfully creating social change. For example, history of the past few years shows that the events of the Arab Spring were in part spurred by the ability of revolutionaries to organize and mobilize the population at large through the use of social networking tools such as Facebook [BFJ+12] and Twitter [EW11]. Additionally, news of unfolding events from the Arab Spring being leaked from within—again through the Internet—engaged the rest of the world, bringing attention to the unfolding events, and applying pressure on the ruling elite.

Indeed, so successful is the Internet for dissemination and organization that oppressive regimes regularly curtail or outright censor it. These regimes are not alone, as many governments, Internet service providers, and corporations exert varying levels of control within their spheres of influence. While there may be legitimate reasons for controlling the spread of information, such as privacy concerns, national security, corporate confidentiality, and public safety, there are many questionable reasons, including the chilling of speech, governmental overreach, and corporate misdeed.

These questionable reasons give rise to the *circumventor* whose aim is to push back by overcoming the censor's best efforts. To this end the circumventor produces, or uses, techniques and technologies to circumvent or resist the censorship apparatus. The current situation is one where widespread and sophisticated networking equipment operated by a corrupt government allows it to enforce unjust and oppressive policies on their citizens. Indeed, the field of censorship resistance concerns itself with bringing balance to the power differential that now exists between the oppressor regimes and the oppressed citizenry.

The battle between the two sides—censor and circumventor—is a hotly contested struggle to gain the upper hand, at least given the current Internet architecture. As the censor learns how to be more effective, the circumventor learns how to overcome, which in turn teaches the censor. This cycle repeats until one side reaches the limits of the resources they are willing and able to invest and the costs they are willing to bear, at which point a tentative equilibrium is reached. The resulting balance, however, may later shift due to technological advances, changing political policies, or changes in resource limits.

At its core, the purpose of censorship resistance is to enable freedom of communication. The ideal *censorship resistance system (CRS)* is one which, even in the presence of a powerful censor, provides a method of communication for users everywhere, enabling connections to arbitrary destinations, and in a manner that does not expose users to the censor's (future) retribution.

It is true that CRSs can be used by malicious parties such as organized criminals. However, the power dynamics are not the same as between the citizenry and censor as we noted earlier. An organized malicious entity has more protections available to it than the average citizen, such as weapons, safe houses, and liquid assets. The CRS certainly can be leveraged, but it is not necessary since the malicious entity can just as successfully harness alternative communication methods. Indeed, these communication methods have existed before the advent of CRS designs, such as burner cell phones, satellite phones, and underground tunnels infiltrating borders. The malicious actor can more easily achieve its goals without CRSs than can the citizen who has no other recourse than to craft his own solution—assuming he is capable.

1.1 Motivation

The CRS community has made great advances in providing circumvention solutions; however, we do not yet know how to evaluate the systems and compare them with one another in terms of difficulty of detection and blocking, performance, and user security, among other questions.

We identify the following concrete problem areas, which we then address in the remainder of this work.

The CRS field is not systematic. We find that there is a great need for the systematization of knowledge of the CRS field. This need is apparent when we consider the lack of common frameworks for designing and evaluating CRS systems. It is clear that great effort

is being made in the pursuit of better CRS systems, yet we are unable to pinpoint what it is about these designs that makes them good, how they interact with each other, and also how they fare under different censorship scenarios. Often intrinsic properties (those that the system can control through design) and extrinsic properties (those that emerge from the environment and are not controlled by the design) cloud matters when we wish to evaluate CRS design choices. A formal framework would allow us to separate the intrinsic properties that are critical for effective CRSs and the extrinsic properties that prevail in the operating environment. These intrinsic properties define the *ideal* CRS that the CRS community can measure themselves against and ultimately aim for.

The next two problems are sides of the same coin. Together they provide the evidence and support for the systematic designs, analysis, and deployment addressed above.

Censors are a mystery. On the one side—while there are a number of useful results in the literature that identify the techniques and equipment being utilized to conduct censorship—there is still a paucity of information when it comes to the censor's beliefs and preferences. This information is of great value to help in the analysis of the more general problem of applying strategy to CRS design and deployment. Taking this strategic view affords the opportunity to escape an escalating arms race where the circumventor tries to outfox the censor.

CRS users are a mystery. The other side of the problem is that we do not have robust models of CRS user behavior. Accurate models can provide valuable input to CRS design leading to 1) apt designs that enhance performance and the user experience and 2) better strategically savvy design choices and deployments. It is a challenge to collect useful user information due to the risks associated with the collection of client-usage data, which may compromise users' privacy and security. Any solution needs to address these concerns for not only the users but also the operators of CRSs.

In this work we make progress on all of these problems to make meaningful contributions to the CRS field and support our thesis:

Thesis statement. Historical and contemporary CRS design is driven by novelty, experience, or favorable environmental conditions with little empirical or formal evidence to support design decisions. As a remedy, we can produce a set of frameworks towards a more principled and evidence-based approach to CRS design.

1.2 Contributions

The scope of censorship resistance research is broad. We narrow our scope and provide contributions to four specific areas, which are listed below.

- *Systematization of Knowledge.* We present a model of the censor and censorship and a taxonomy of censorship resistance in Chapter 2. These models allows us to analyze different censorship threats and the applicable circumvention strategies. This analysis leads us to identify gaps in the censorship resistance literature that we selectively address.

- *Game-Theoretic Analysis.* We provide a detailed analysis of the censor as a rational actor and produce a framework to model censorship as a strategic game with the CRS and censor acting to increase their utility in Chapter 3. We present discussions of various censorship scenarios. We then apply our game-theoretic framework to a real-world problem of CRS deployment that maximizes the outcome for the censorship resistor.

- *Privacy-preserving CRS Data Collection.* In our game-theoretic framework we notice the need for empirical data, which is presently difficult to come by. One of the major concerns is that data collection on CRS networks adds risk to its users. In Chapter 4, we present a privacy-preserving data collection framework for use with CRS networks and design a system that leverages it. We also present the results of a real-world deployment of this system.

- *Analysis and Prevention of a Client-Linking Attack in a CRS.* Any deployed CRS must resist attempts by the censor to scuttle the protection it provides. In Chapter 5, we investigate the design of Tor, a commonly used CRS, to analyze its resistance to active attacks by a censor aiming to identify its users. We find that the deployed design is suboptimal and propose recommendations to enhance user privacy and security. Our proposals have already been accepted by the Tor community and have been rolled out into the deployed Tor network.

Chapter 2

A Systematization of Internet Censorship and Resistance

We first present general background about Internet censorship and resistance to help orient and motivate the discussions in the following chapters. We also systematize the literature to provide a coherent framework to help our discussion and evaluation of censorship and resistance use cases, properties, and strategies.

Generally speaking, the Internet is composed of many networks—operated and maintained by various entities large and small—connected together to allow communication across different parts of it. Cooperation is the lynch pin that allows everything to work more or less seamlessly. In particular, the Internet was not designed to prevent an active adversary from reducing network connectivity and curtailing communications between parts of it. In this work, we identify censors as adversaries that actively prevent free and open access for users in their sway to information and opportunities for collaboration, communication, and commerce with outside entities.

Overlay networks utilize pre-existing network connectivity between users on the Internet. They leverage this connectivity to establish logical networks, which have customized data transmission mechanisms, naming and addressing schemes, and network topologies and behaviors. These logical networks add additional functionality that is different from the underlying network they overlay. It is important to note that the overlay network must interface with the underlying network to realize its functionality; however, from the perspective of overlay network users, data, and observers, the interface is invisible and the overlay network is the only reality. Examples of overlay networks existing on the Internet today are the BitTorrent file-sharing network and the Tor anonymous communication network.

The censor typically utilizes components commonly found in network deployments, such as firewalls and intrusion detection systems (IDS).

A firewall, labeled "Censorship Apparatus" in Figure 2.1, is a common tool used to enforce access control to certain parts of the Internet based on some set of rules that implement a policy. The most significant firewalls (for censorship) are those operated by nation states, *i.e.* the censors, that control Internet access, *i.e.* censorship, for their populations.

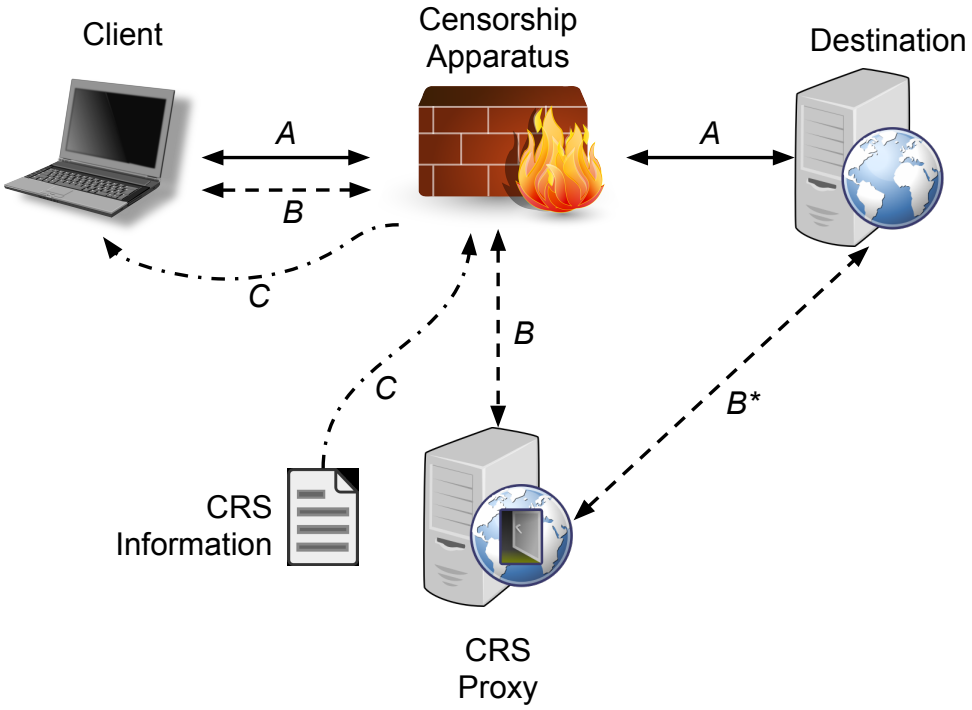


Figure 2.1: A general censorship arrangement with a simple CRS deployment.

There are many types of firewalls; some can only filter based on IP addresses, while others can inspect and filter based on the contents of individual packets, while still others are able to keep state across many packets and reconstruct what is occurring at the application level and filter on that. We go into further details of firewall capabilities in Section 2.4. Intrusion detection systems are related to firewalls, and are typically deployed behind the firewall on the internal network to detect unwanted network entity *behavior* or traffic *patterns*. These usually work in concert with the firewall to make it react to threats.

We assume that the censorship apparatus is composed of both firewall and IDS functionality and do not differentiate between them for the remainder of this work, using the generic term "firewall" to refer to both.

A network proxy, labeled "CRS Proxy" in Figure 2.1, is a tool that can be used to bypass firewall restrictions. It provides an indirect method of accessing a restricted part of the Internet. The only condition is that the proxy itself be accessible by users behind the firewall.

A *channel*, at an abstract level, is a network path that allows the flow of information from an origin to a destination on the network. More concretely, a channel is constructed with networking software, which provides mechanisms for the establishment, maintenance, and utilization of the channel. While there are a large number and variety of network software and mechanisms, we are only concerned with their functionality in general and refer to them in abstract as a "channel mechanism".

Overt channels do not try to hide their existence and the censor can detect them passing through their point of presence on the network and are allowed since they are not targets of censorship. Often, with a few exceptions, the channel between the client and CRS proxy is an overt channel. *Covert* channels attempt to hide from the censor's detection. Generally, they do this by using the overt channel as a cover for their covert traffic. The idea is that the covert traffic can hide within the overt cover traffic and escape detection. As an example, a possible overt channel is encrypted email traffic that in reality is being used as a cover for HTTP requests designed to look like encrypted email traffic. To the censor both types of traffic look the same and hence, in the ideal situation, are not blocked by the censor's firewall.

A common work flow for CRS utilization is depicted in Figure 2.1: The client desires to connect to a censored destination, which in the absence of censorship would have been done via channel A, but that channel is blocked by the censor. To overcome this obstacle, a CRS is employed, which the client utilizes through channels B and B^*, which have been created via a difficult-to-block mechanism utilizing overt and covert traffic to and from the CRS proxy. In most cases the client will need to learn about, and how to communicate with, the CRS proxy and so CRS information is learned through a channel C, which is also difficult to block like channel B but has shortcomings, *e.g.* very high latency, making it unsuitable as a data channel.

2.1 CRS Users and Use Cases

Users of CRSs include whistleblowers, content publishers, citizens wishing to organize, and many more, and any one CRS may be employed simultaneously for multiple use cases. Each use case may require different properties of the CRS. For example; a whistleblower needs to have their identity protected and be able to get their message to an outsider without raising suspicions that this is occurring; a content publisher wants that no amount of coercion, of themselves or the hosts of their content, could impact the availability of their content; and citizens wishing to organize would want unblockable access to their self-organization tool.

By examining the literature and the use cases therein, we extract common CRS security properties, keeping in mind that not every CRS design aims to achieve every property. We describe these security properties next.

An **unblockable data channel** is one that the censor, having identified it, is unable to block due to extenuating circumstances, most usually the threat of collateral damage. If the censor is unable, or unwilling, to block the data channel then the CRS activity can proceed unmolested.

An **undetectable data channel** is one that the censor is unable to identify as belonging to CRS activity. This is a useful for cases where the censor *could* block the channel if they knew it were CRS related.

An **anonymous publisher** is one whose identity is hidden from CRS-participating entities and observers on the Internet in general. This is to prevent the chilling of speech that the threat of retribution by the censor would cause on the publisher.

An **unlinkable client-destination path** is one where an observer, within the CRS network or outside it, cannot tell which destination a particular client is communicating with. This prevents the censor, or any other observer, from learning whether the client is communicating with known undesirable, and censored, destinations.

An **incoercible publisher** is one for whom the censor's threats of force are not credible due either to the fact that the publisher is outside the censor's reach or to the fact that the CRS is designed in such a way that makes it impossible to comply with the censor's demands.

An **incoercible covert destination/storage** is one for whom the censor's threats of force are not credible for the same reasons as those for the incoercible publisher above.

A **deniable client participant** is one whose participation in the CRS is blurred with innocuous activities to create reasonable doubt in the censor's mind and give the client plausible reasons for their, secretly CRS related, activities. This is to mitigate the threat of censor retribution and the chilling of speech.

A **deniable forwarder participant** is, like the client participant above, one whose participation is covered in reasonable doubt and this helps to prevent the censor from blocking it or meting out punishment.

A **deniable covert destination/storage participant** is, like the forwarder participant above, one whose participation is covered in reasonable doubt for the same reasons.

Along with the requirements for security, CRS clients' usage is impacted by what is possible to accomplish given the characteristics of the communication channel. Like the security properties above, we again refer to the literature to extract common channel characteristics below that help us compare and contrast CRS designs.

1. **Latency** is the delay introduced by the CRS in carrying out a user action, *e.g.* sending a message, or receiving a file. The two types of CRS are low-latency and high-latency with the former being useful where delays would impact usage, *e.g.* web browsing, and the latter for uses that are not impacted by delays, *e.g.* leaking a document to a whistleblowing website.

2. **Throughput** is the amount of bandwidth that the CRS needs to harness for both its own functionality and to enable the use cases it was designed for. There are two types: high-throughput and low-throughput. An example high-throughput CRS is Tor where it consumes the bandwidth dedicated by numerous volunteer routers that form the network, while Collage is a low-throughput CRS since it can only harness bandwidth on the same order of magnitude as the size of image files that are posted on photo-sharing websites.

3. **Goodput** is the useful bandwidth available, of the total throughput bandwidth above minus the CRS's operational overhead, for CRS user activity. A high-goodput CRS means that the CRS is high-throughput and has low operational overhead resulting

in high amounts of bandwidth available for user activities. A low-goodput CRS means that either the CRS is low-throughput to begin with or that the CRS is high-throughput but has high overhead resulting in little bandwidth for user activities. In the CRS examples above, Tor is a high-goodput CRS and is useful for large data transfers, while Collage is low-goodput since it uses steganography, that adds a large overhead but is still useful for smaller data-footprint activities.

4. **Reachability** is the range, or extent, of the Internet—which is composed of people, web pages, and service platforms—that a CRS user can communicate and/or interact with. The range can be general or limited; general being the same as unfettered access to the Internet and limited being a curtailed set of niche destinations, audiences, and/or content. The former is useful where the user wants transparent access to the Internet such as a user who wishes to instant message with friends, watch streaming movies, and post to social networking websites. The latter is useful where the user only requires a limited part of the Internet such as communicating amongst a tight-knit group of dissidents who only require access to one bulletin board system.

5. **Interactivity** of the CRS channel tells us whether bidirectional communication is possible between CRS users or whether it is only unidirectional. A CRS that is bidirectional means that it can be used by a user to both send and receive data, whereas a unidirectional CRS can only transmit or receive data during any given session of use of the system. A unidirectional CRS is useful where the user only wishes to post to a message board but will not read the board themselves or when they only wish to access information but will not be adding anything themselves. A bidirectional CRS is useful for times where users needs to interact with, and react to, other users and there is a need for a back and forth within the same session of use of the CRS.

These channel characteristics, being either high/low, general/limited, or bi-/unidirectional, define the *quality of service*[1] (QoS) that a CRS user can expect. The channel QoS dictates the user experience and the kinds of activities that are possible on that channel and hence the use cases that it can support. We will see later in Section 2.8 how CRS design goals are driven by user requirements for security and QoS.

[1]Usually quality of service is concerned with the performance of a given channel (*i.e.* latency, throughput, and goodput); we add CRS-specific channel characteristics (reachability and interactivity) since these are also pertinent in the CRS setting.

2.2 Internet Censorship

Censors vary widely with respect to their motivation, sphere of influence, and technical sophistication. A wide range of entities, from individuals to corporations to state-level actors, may function as a censor. Therefore, a CRS should be precise as to the type and capabilities of the censor(s) that it is designed to circumvent.

In particular, the extent to which a censor can effectively attack a given CRS is a consequence of that censor's resources and constraints. We will focus on the technical resources, capabilities, and goals of the censor, which are informed by their *sphere of influence* (SoI), their *sphere of visiblity* (SoV), and their set of *economic constraints*.

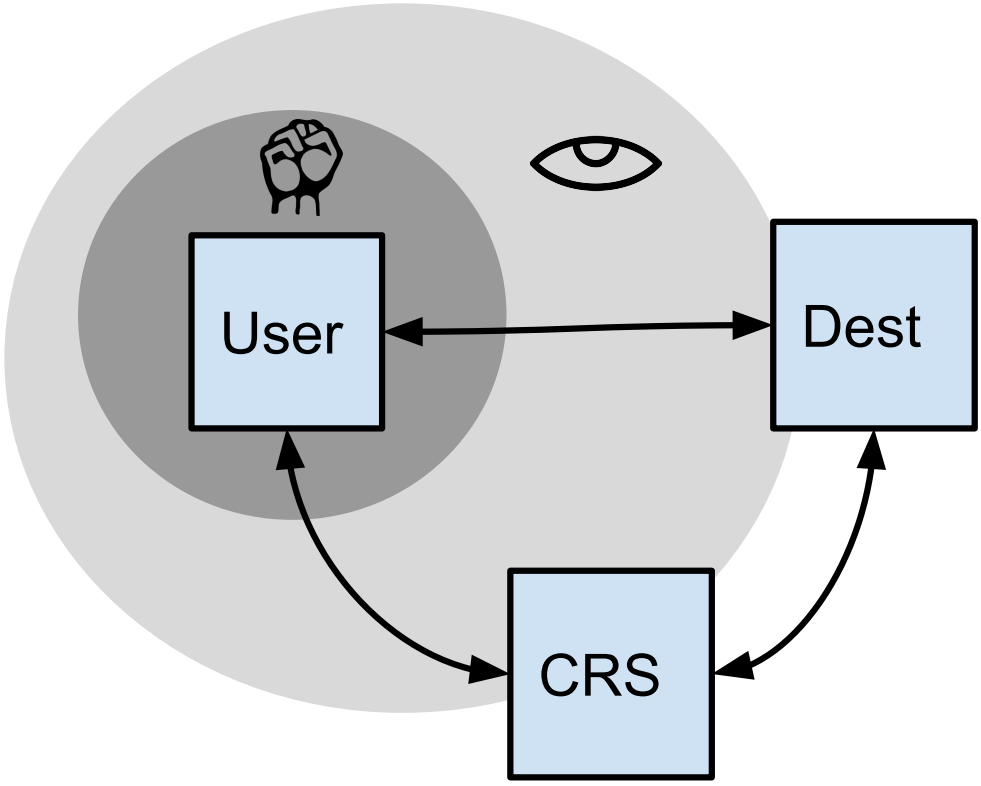


Figure 2.2: A simplified SoI (dark gray area with fist) and SoV (light gray area with eye) arrangement. The arrows depict the communication channels among the entities. The channel between the user and destination is within both the SoI and SoV, but the destination itself is outside the SoI but may be within the SoV.

Sphere of Influence (SoI) is the degree of *active* control the censor has over the flow of information and behavior of individuals and/or larger entities. Similarly, *Sphere of Visibility* (SoV) is the degree of *passive* visibility a censor has over the flow of information on its networks and those of other operators. The success or failure of a CRS depends in large part on the assumptions its designers make about the SoI and SoV of the censor.

Figure 2.2 depicts a simplified, yet prevalent, example SoI and SoV. The user is within the SoI and SoV of the censor as is their direct traffic to any destination on the Internet, including to the CRS. Of course, unapproved destinations are blocked, leaving the CRS the only means of access to those destinations. The CRS itself is not hidden from the censor, but it is outside the censor's SoI. This setup depicts the usual CRS scenario. Note that the censor *could* influence, *e.g.* block, the direct traffic between the user and CRS but, with the right CRS design in place, it is confounded about which traffic to actually interfere with without impacting non-CRS traffic. We will discuss in-depth details of the various CRS strategies in Section 2.6 that allow this censor confounding to happen. The censor can always block all traffic, to the CRS as well as to the general Internet, but we will examine in Chapter 3 how this course of action is not typically in the censor's best interests.

This is the essential challenge of CRS design: to allow users to access censored content and destinations while within the influence and visibility of the censor and to keep the censor ignorant of it despite its best efforts.

We specify the limitations to both SoI and SoV due to constraints that are *physical*, *political*, and *economic*. Limitations due to geography are an example of physical constraints, while relevant legal doctrine, or international agreements and understandings, which may limit the types of behavior in which a censor is legally allowed or willing to engage are examples of political limitations. Economic constraints assume the censor operates within some specified budget.

2.2.1 A Model of Censorship Apparatus

Recall that the aim of the censor is to distinguish between acceptable and unacceptable traffic. The censor first deems some traffic as unacceptable, and when the population starts to use CRSs, the censor then prohibits the *CRS traffic itself*. We focus on the latter, yet allow that the former can also be aptly accommodated, in the discussion that follows. With this in mind we model the censorship apparatus next.

At an abstract level the censorship apparatus is composed of a classifier and cost function that both feed in to a decision function as depicted in Figure 2.3. This simplification

allows us to distill censorship activity to three distinct categories or phases: Exposure, Detection, and Response. We then leverage this model in our censorship resistance taxonomy to follow in Section 2.6.

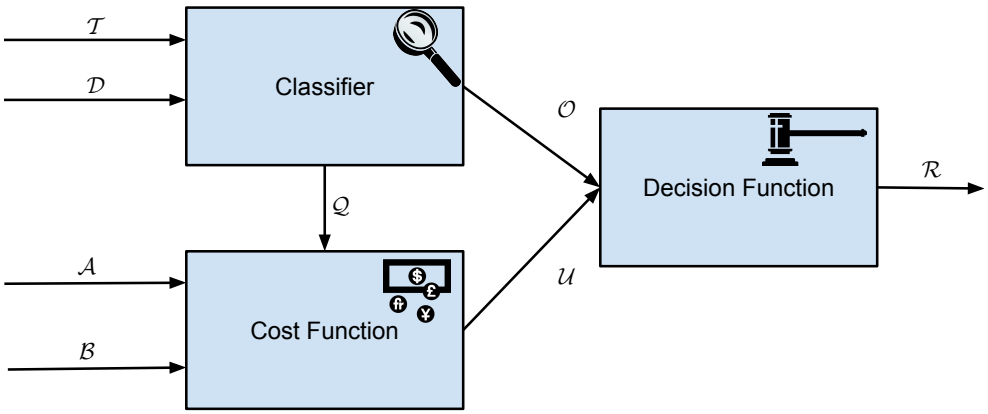


Figure 2.3: An abstraction of the censorship apparatus and workflow. $\mathcal{T}$ is the data (network traffic), $\mathcal{D}$ are the distinguishers to be used for detection by the Classifier and $\mathcal{Q}$ denotes the error rates of the chosen distinguishers (*i.e.* FNR and FPR); $\mathcal{O}$ is the output (*i.e.* flagged traffic) of the classifier. $\mathcal{A}$ and $\mathcal{B}$ are the censor's tolerances for collateral damage (FPR) and information leakage (FNR) respectively and $\mathcal{U}$ is the resulting utility function parameterized by $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{Q}$. $\mathcal{R}$ is the censor's response based on the input $\mathcal{O}$ and $\mathcal{U}$ into the Decision Function.

In the first phase the censor *exposes distinguishers* ($\mathcal{D}$) that can, within an acceptable margin of error ($\mathcal{Q}$), identify prohibited network activity. This will be one of the inputs of the Classifier, along with the network traffic ($\mathcal{T}$), once it is deployed. As an example, a censor may learn that a particular packet length is dominant for most, but not all, of a particular CRS's traffic, *e.g.* in Tor packets of length 586 bytes are prevalent. There is also a small amount of non-CRS traffic that also has that particular packet length which the classifier will confuse with CRS activity. If the error rates are acceptable to the censor, the packet length of 586 bytes will be used as a distinguisher by the classifier.

In our model, we define the error rates as the *false negative rate* (FNR), also known as information leakage where CRS traffic is mislabeled as being legitimate, thus allowing the CRS user to access prohibited information, and *false positive rate* (FPR), also known as collateral damage where legitimate traffic is mislabeled as being CRS related, thus

causing the censor to block legitimate traffic. We assume that the censor would prefer a lower FNR and FPR whenever it has the opportunity. This is a reasonable assumption since increasing these errors would not increase the censor's utility in any realistic manner. However, accuracy alone does not explain the observed behavior of censors, nor is it the only input in to the censor's decision making. We explore some of the missing aspects of the censor's decision function in Chapter 3.

In the second phase, the Classifier tries to *detect* all instances of the distinguisher using a classification mechanism.[2] In our example above, the classifier will flag network flows where packets of length 586 bytes occur above a censor-defined threshold.

Finally, the censor will *respond* to the output ($\mathcal{O}$) of the Classifier—along with input from the Cost Function ($\mathcal{U}$) which takes in to account the censor's tolerance for errors ($\mathcal{A}$ and $\mathcal{B}$)—with a follow-up action ($\mathcal{R}$) to limit or stop the prohibited activity. In our example, the response could be to disconnect a flow, *e.g.* by sending RST packets to the server to break the connection when packets of 586 bytes occurred too often, rendering the CRS ineffective. The response would depend on the actual mechanism employed by the CRS and how susceptible it is to different forms of interference. The censor needs to balance the response to ensure that non-CRS activity does not become collateral damage.

It is clear that distinguishers play a vital role. We define them as being composed of *feature* and *value* pairs. A feature is just an attribute, such as an IP address, protocol signature, or a packet size distribution. A feature has an associated value which can be a singleton, a list, or a range. In general, values are drawn in the form of a *distribution* from all possible values that feature can take. Where this distribution is sufficiently unique to the CRS the feature-value pairing is dubbed a distinguisher and can be used by the censor to detect prohibited activities. In our example above, the feature was packet length since there was a peak in the distribution at 586 bytes. Now the packet length as the feature with value of 586 bytes forms a distinguisher the censor can utilize to identify the CRS.

The primary source of distinguishers is network traffic within the censor's SoV. We can map these distinguishers to the network, transport, and application layers of the network stack. Distinguishers can be extracted from the feature-value pairs within headers and payloads of protocols at the various network layers.

Some concrete examples include the source and destination addresses in IP headers at the network layer, source and destination ports and the sequence number in the TCP

[2]We gloss over the details of the actual mechanism for the purpose of clarity of description and discussion. In reality the mechanism might be deterministic or stochastic, real-time or off-line, and have thresholds for determining when to tag something as being CRS related, as well as a host of other parameters. We abstract away these details without loss of accuracy or generality.

header at the transport layer, and the TLS record content type in the TLS header in the application layer. Furthermore, the payload of the packets, if unencrypted, can reveal forbidden keywords. The censor may act on these distinguishers without fear of collateral damage since there is little uncertainty about the veracity of the information.

The censor also learns from the explicit distinguishers above about the expected behavior of the traffic. A certain profile of feature-value pairs implies particular, and known, traffic behavioral characteristics that the censor can use to detect anomalous, and potentially CRS, traffic. For instance, a VoIP channel usually has data flowing in both directions but rarely at the same time, as both parties are usually communicating interactively and take turns to speak. A CRS that uses this channel for bulk download would only have data flowing predominantly in one direction, with some data flowing back for data transmission control, which would be the distinguisher the censor can use to detect the CRS usage.

To augment these explicit distinguishers, the censor can collect statistics about the traffic to learn additional distinguishers such as packet length and timing distributions.

Distinguishers are high- or low-quality depending on if they admit low or high error rates respectively. Furthermore, they can be low- or high-hanging depending on if little or large amounts of resources are required to field them respectively. For the censor, high-quality low-hanging distinguishers are ideal, whereas for the circumventor denying any kind of distinguishers is ideal, but driving the censor to depend on only high-hanging and low-quality ones is preferred if the ideal situation is not possible.

2.2.2 General Censor Threat Model

We now define our threat model based on the network-level capabilities of a censor. A censor has complete visibility (SoV) of network traffic flows within its SoI. It may have additional visibility outside of its SoI, such as information gained from collusion with third parties. In particular, the censor may be a *passive* adversary with respect to some portion of the network, only able to observe channel content, such as when an ISP grants access to past network traffic flows.

The censor may be *active* with respect to some portion of the network (within the SoI), with the ability to *modify*, *delete*, *inject*, and *delay* channel content. Furthermore, the censor may have the ability to modify channel *characteristics* as well as content, such as controlling the timing patterns of traffic. The censor may have additional insight into the CRS system, that may be gleaned by being an active CRS participant or by devoting resources in a volunteer-based CRS.

2.2.3 General Censor Goals

With these capabilities in mind, we define the technical goals of a censor as follows:

1. *Detection and Identification* of CRS traffic. The censor wants the ability to categorize individual traffic flows as being part of a CRS. In particular, the censor may wish to expose the identity of the person using the CRS, or the type of censored material accessed.

2. *Disruption* of CRS traffic. The censor wishes to have the ability to disrupt any CRS-related traffic of its choosing in a way that renders the CRS ineffective.

2.3 Censorship Resistance

Censorship resistance is the act of overcoming the threats of the censor's SoI and SoV on censored traffic—including the traffic of the CRS itself, and user and operator security—while achieving an acceptable level of performance so as to be useful. There are a large number of CRS designs that exist; we extract from them a common set of CRS components, attack surfaces, design goals, and strategies, which we present in the remainder of the chapter. Table 2.1 provides the structure and summary of the discussion that follows.

2.3.1 Censorship Resistance Components

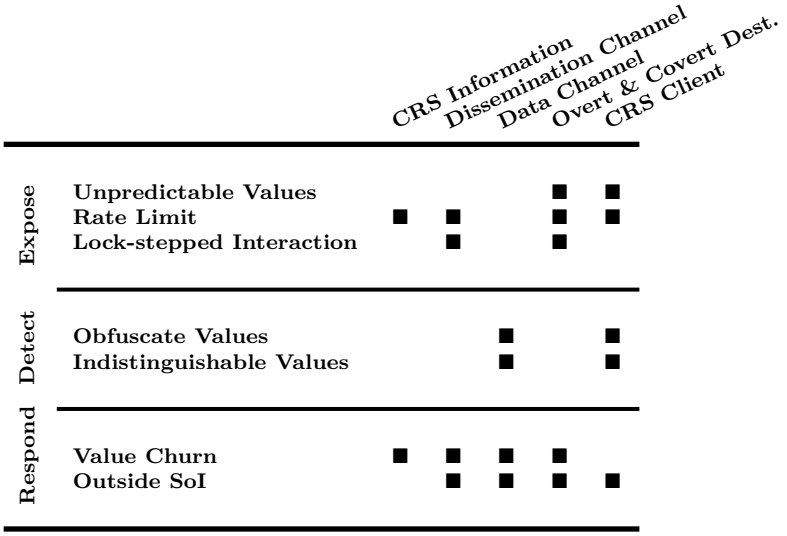
We now describe the CRS components in detail (see Figure 2.4).

Out-of-band CRS Information. Certain CRSs require secret information to *bootstrap* the data channel. This may include setup or operational information such as symmertric keys, network dead-drop locations, and any other information that is necessary for the client to learn how to participate in the CRS.

Out-of-Band Channel. A channel outside the censor's SoI and SoV is termed an *out-of-band channel*, and the censor can neither view nor affect it. This channel is used to communicate out-of-band CRS-operational secrets that are assumed to somehow arrive with absolute security versus the censor.

The key is that there is a limitation inherent to the channel that does not allow it to be used as an ongoing data or dissemination channel, which would also obviate the need

Table 2.1: Strategies employed to defend against censorship threats to CRS components. CRS strategies, to be outlined in Section 2.6 are on the left, divided into the censorship phases described in Section 2.2.1. The CRS components, outlined in this section, are along the top. A solid black square indicates that this strategy has been applied to this CRS component.



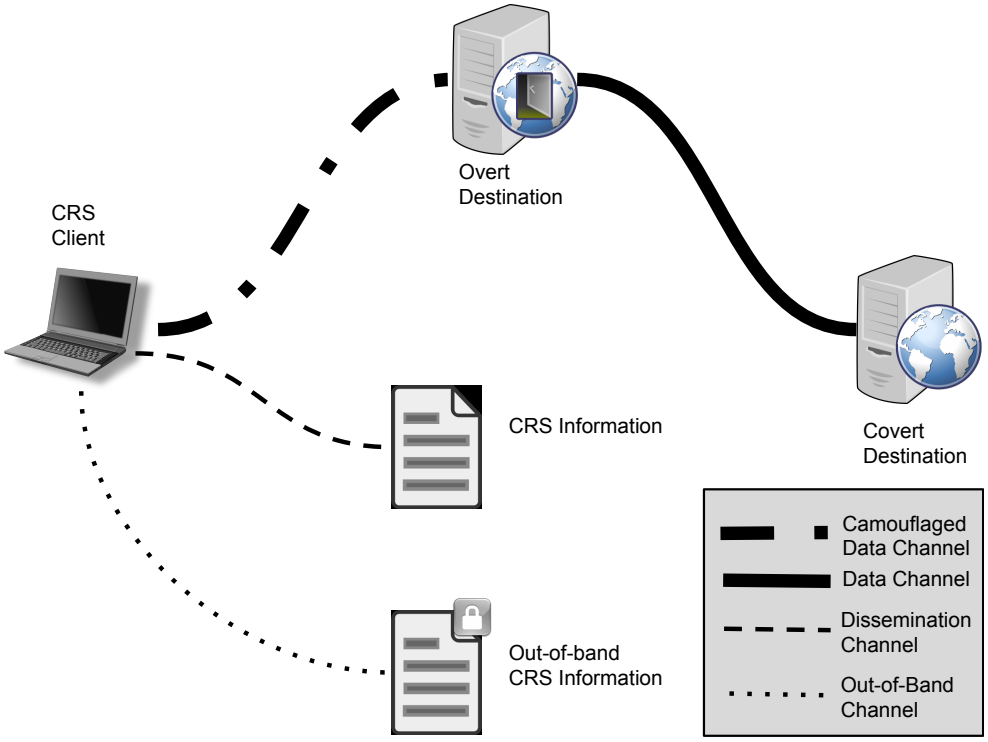


Figure 2.4: High level components of a CRS.

to use a CRS in the first place. An example of such a channel is a person from outside the SoV bringing addresses of CRS proxies on a USB stick through the airport and hand delivering it to the client.

CRS Information. In order to participate in the CRS, a client must discover information about the CRS, which can include addressing, naming, and routing information as well as information to help with security needs such as authentication and encryption. In general this is any information that the client needs to participate in the CRS.

Often this information is general and public, such as domain name mappings to IP addresses, routing information, and other general details that allow Internet communications. However, it is also often the case that along with this public information, CRS-specific information is required, such as addressing information of CRS proxy servers, locations of censored content, and other CRS-specific details. While Internet information is available to all, under certain designs CRS-specific information could be restricted to prevent the censor from learning it or from tampering with it.

Dissemination Channel. CRS information is requested and served over a dissemination channel. This channel can take many forms, such as ordinary email to a publicly available address, a CRS-client issued lookup request to a directory serving a file, or an anonymized encrypted connection to a hidden CRS information store, to name a few. Generally, the throughput and network characteristics of the dissemination channel are limited as compared to the data channel as depicted in the figure by the relative thinness of the line as compared to the data channel line. The dissemination channel is within the censor SoV, and often SoI.

Data Channel. There are two types of data channels: one that is between the client and overt destination, and another that is between the overt and covert destinations. Not every CRS uses a channel of the second type. In all events, this is the data-carrier channel transporting censored data to and from the client and the overt destination, and where needed to the covert destination.

The client-to-overt-destination data channel may be camouflaged if the censor uses traffic analysis to detect CRS activity. The overt-to-covert-destination data channel, if one exists, may also have the same properties, but it may also not be camouflaged at all since it is outside the censor SoV and SoI. The two channels may use different network mechanisms and have very different security and performance properties.

Overt Destination. This is the destination that the censor can observe the client ostensibly communicating with over the client-to-overt-destination data channel.

The overt destination can play a few different types of roles depending on the CRS:

1. Forwarder: When the client wants to communicate with or gain access to censored destinations that are external to the CRS, the overt destination acts as a proxy forwarding traffic to and from the client and external destination.

2. Dummy Destination: Ostensibly, the client is communicating with the overt dummy destination but in reality she is communicating with a third covert destination. As an example, this can be achieved by means of networking equipment that diverts the CRS traffic enroute to the dummy destination to the covert destination without exposing it.

3. Covert Destination/Data store: When the covert page, person, or platform (see next entry below) the client wants to communicate with is actually available at the overt destination, the overt destination simply serves the content directly to clients using the CRS.

Overt destinations are usually aware of CRS activity since they are playing an active role in the data-communication channel and when they are acting as covert destinations. Indeed, there most likely needs to be an installation of special software to enable CRS functionality.

However, there are exceptions. For example, where the CRS design leverages already-extant overt destination functionality, the overt destination may be unaware of CRS activity since it does not deviate from its regular operation. The CRS activity is a side effect, albeit intentional due to the CRS mechanisms that were employed. A real-world example is Google's AppEngine, that allows websites to be hosted on Google's cloud storage and using Google's IP addresses. A CRS that hosts content, for example instructions on how to circumvent firewalls, on AppEngine is leveraging normal functionality for CRS uses and does so without Google's explicit acceptance or knowledge. [FLH+15] Another exception is a dummy destination, which also does not even have to be a CRS participant or be aware that it is being invoked by the client for the purpose of fooling the censor. A real-world example is where, in response to a request from a client, a CRS server hosting content can send censored content to the client and fool the censor as to the origin of the traffic by spoofing the sender address with a dummy address. [Hsu00, WGN+12] The host at the dummy address does not need to be involved in this deception.

There are variations to the basic roles above that may come in to play and we will point them out where appropriate.

Covert Destination. This is the covert page, person, or platform the CRS client ultimately wants to connect to. Page refers to any covert information that can be read. Person is any entity that the client can interact with. Finally, platform refers to an application or service that the client can utilize or leverage. This definition allows for use cases that require interactivity as well as read and publish mechanics.

Some concrete examples include being able to read a dissident blog (page), a video call with a friend (person), and tweeting about the location of the next anti-government rally on Twitter (platform).

CRS Client. This is special CRS client software, or non-CRS software with CRS-enabling modifications, that can utilize CRS information, channels, and destinations to give the desired CRS behavior to the client.

Depending on the censorship scenario, a client may need to first obtain this software through the out-of-band or dissemination channels.

We now describe the work flow with respect to the components of a CRS just described:

1. As a preliminary step the user may use an out-of-band channel to gather bootstrapping information that facilitates access to and use of the CRS. This step is not mandatory and only required by some systems.

2. A CRS user learns (enough) CRS information, through the dissemination channel, to allow them to connect with an overt destination that will act as a CRS proxy,[3] usually using CRS client software. In the case of the dummy destination, the actual CRS proxy is the network equipment on the path to the dummy site.

3. The client connects to the CRS proxy over the data channel, which may be camouflaged to prevent detection.

4. The CRS proxy services the client's requests, either fulfilling them itself, from local resources, or forwarding them on to one or more covert destinations.

2.4 Attack Surfaces

Each of the CRS components can be targeted to attack CRS security and functionality, and ultimately clients; we therefore refer to these components as *attack surfaces*: the exposed pieces of the CRS able to be attacked. We will look at a number of general attacks on each surface, summarized in Table 2.2, to gain insight about the particular issues and challenges inherent to these CRS components.

2.4.1 CRS Information

Harvest: The censor can readily harvest CRS information that is public by observation alone, and where it is restricted to clients or CRS operators it can do so by actively infiltrating, or compromising, the CRS dissemination process. This is exactly the process that some governments wishing to block access to the Tor network have done, such as China [Lew09].

[3] Recall that the overt destination is not blocked and is either not suspected by the censor as participating in a CRS, or is a high-value site that would cause too much collateral damage to block, or is an innocent dummy destination.

Table 2.2: Attack surfaces and the general attack vectors that are relevant to each.

Attack Surface	General Attack Vector
CRS Information	Harvest Poison
Disseminiation Channel	Deny Masquerade
Data Channel	Detect Deny Disrupt & Degrade
Overt/Covert Destination	Comb Coerce Crush Curtail Corrupt
CRS Client	Compromise & Deny Coerce Link

Poison: The censor can corrupt, or modify, public information it is in control of as well as the information it can inject into the CRS where it is possible. Examples include DNS poisoning [BCK12] and routing table changes that isolate traffic to known, trusted, and censored paths [SGTH12]. The censor needs to be careful about corrupting information [ICA12] needed for Internet operations since it can impact non-CRS use. [McP08]

2.4.2 Dissemination channel

Deny: The dissemination channel mechanism may be blocked, in the same manner as data channels (see below), thus denying legitimate clients access to CRS information.

Masquerade: Where the CRS depends on secrecy of CRS information the censor can pretend to be a legitimate client and use the dissemination channel to harvest information. China has twice, in 2009 and again in 2010, overwhelmed Tor's bridge distribution strategies by simply pretending to be enough legitimate users from different subnets on the Internet. [Din11c]

2.4.3 Data channel

Detect: From header information, which is in the clear, the censor can tell if the traffic is going to a known CRS overt destination. Also, if it is in the clear, the content (or payload) can provide evidence that the packet is CRS-related. Where this information is unavailable, or obfuscated, traffic analysis based on packet statistics and behavioral signatures can be used. The censor would compare these clues against known CRS statistics and behaviors to see how well they match. [Din11a] Where the clues match known non-CRS statistics and behaviors, the censor would look for discrepancies that would give away the pretense.

Deny: The censor can drop all the identified CRS traffic at the firewall or another point of presence, which would effectively sever the CRS data channel. For instance the Chinese firewall, circa 2004, sent RST packets to both ends of a network flow whenever it detected a Tor connection thus effectively severing the connection. [CMW06]

Disrupt/Degrade: Alternatively, especially when it is not confident, the censor can manipulate the traffic by injecting spurious packets, dropping key portions of the traffic, or modifying the contents of packets. [Lew12] Similarly, again when it is not confident, the censor can also manipulate the characteristics of the underlying network link the data channel is utilizing and introduce delays, low connection timeout values [Tor13], and other constraints. This can be used where the CRS data channel is brittle and not resilient to errors. This is especially useful where the overt traffic is more robust to the disruptions and degradations the censor introduces than the CRS traffic. [GSH13, LSH14] The censor can leverage this discrepancy between CRS and non-CRS data channel error handling to target only the CRS while leaving the non-CRS data effectively unmolested. These tactics can guard against collateral damage that would otherwise be incurred if the censor employed more drastic measures like disconnecting a network flow.

2.4.4 Overt and Covert Destinations

Comb: Where the censor is unable to efficiently harvest through the dissemination channel, they may comb the Internet by probing destinations for telltale signs of CRS presence. For instance China has been very aggressive in probing for bridges by watching outbound TLS connections to U.S. IP ranges and then following up with Tor connection requests to verify the existence of a bridge. After this confirmation step, all TLS connections are dropped by the Chinese firewall for users within China connecting to that bridge. [Wil12]

Coerce: Within its SoI the censor can use legal, or extralegal, coercion to shut down an overt destination. Indeed, history shows that this has been successful in the past, for example where a popular anonymous email service was pressured by the U.S. government to reveal users' private information [Fis05].

Crush: For destinations outside the censor's SoI where force or legal pressure cannot be applied, the censor can mount a network attack. An example of this was when China launched a distributed denial of service attack (DDoS), which overloads the destination so that it can not service legitimate requests, against Github to target censorship resistance content hosted there. [Goo15] Other attacks based on infiltration of resource pools can cause similar denial of service attacks.

Curtail: A censor can easily block the IP addresses of overt destinations hosting "undesirable" content, with firewall filter rules and DNS poisoning or injection attacks against CRS information. Such methods are widespread and deployed as a matter of course by censoring regimes. Aryan *et al.* [AAH13], Nabi [Nab13], and Clayton *et al.* [CMW06] provide evidence of this for Iran, Pakistan, and China respectively. The censor can also enact a "whitelist" that only allows access to approved destinations.

Corrupt: The censor could set up malicious resources, such as proxy nodes or fraudulent documents that counteract the CRS's goals, to attract unwary CRS clients and negatively impact their CRS usage. For instance, adversarial guard relays are known to exist on the Tor network, and they can be used to compromise Tor's client-destination unlinkability property. See Chapter 5 for an in-depth analysis of this form of attack.

2.4.5 CRS Client

Link: The censor could try to implicate clients for using a CRS, since such systems may be regarded as having only illegitimate uses. The censor can do this by identifying client connections to known CRS-participating overt destinations. The censor may further be interested in access to covert destinations and/or content and can do this by attracting clients to use or access censor-controlled CRS resources. Like the corruption attack above, Chapter 5 provides an investigation of this attack vector as well.

Coerce: The censor can pressure publishers into retracting their publications, and/or chilling their speech. China regularly regulates the online utterances of its citizens, using an army of thousands of workers who monitor all forms of public communication

and identify dissidents [ZPP+13], and uses force, such as imprisonment, to achieve a chilling of free speech [Rif15].

Compromise & Deny: The censor can install monitoring software on the client computer. This could be publicly announced and sanctioned by an authority, such as in the case of Green Dam Youth Escort [WYH09]. The censor could impose rules that demand clients install only compromised versions of communication software such as TOM-Skype [mar12]. Finally, the censor can also install malware through covert means or by tricking the unsuspecting client. The censor can block at the client level by disallowing unapproved software from being installed on the operating system, disrupting functionality of Internet searches by returning pruned results, and displaying warnings to the user to dissuade them from attempting to seek or distribute censored content through the client or even to use it.

2.5 Desired CRS Design Goals

We began this chapter by examining users and their use cases and extracted the relevant security and performance requirements in Section 2.1. We modeled the censor and the censorship apparatus and partitioned censorship activity into three phases in Section 2.2. We then investigated general censorship resistance design and drew out the common CRS components in Section 2.3, as well as the threats to each in Section 2.4. We are now in a position to consolidate what we have learned into a coherent set of CRS functionality, security, and performance design goals.

At the very general level a CRS has the following functionality goals:

1. *CRS Information Discovery*: The user must be able to learn proxy and/or content network locations, as well as auxiliary information required to participate in the CRS.

2. *CRS Data Transport*: The CRS data channel must be able to punch through the censor's technological barriers and connect users with destinations (pages, people, platforms) from which they would otherwise be blocked.

3. *CRS Bootstrapping and Availability*: The CRS should be designed to deploy on the Internet of today and maintain consistent availability. It should not depend on features and functionality that have not yet become, and are not likely to become in the near term, a reality on the Internet.

A CRS must ensure the security of the above three functions as well as that of CRS users and operators. We now consolidate the required CRS-security properties, that we have extracted from the literature on attack surfaces and vectors, and real-world examples. They are organized by the three phases of censorship from our model in Section 2.2.1:

- **Exposure Reduction**
  1. *Prevent CRS information harvesting* that is possible by masquerading as a CRS client and combing the Internet address space.
  2. *Prevent CRS participant identification* that is possible by masquerading as a CRS client or overt/covert destination.
  3. *Prevent active probing* by the censor who is trying to distinguish whether CRS infrastructure or content is present at a particular network location.
- **Detection Prevention**
  1. *Provide an undetectable data channel* that camouflages traffic patterns, behaviors, and inconsistencies detectable by the censor.
- **Response Mitigation**
  1. *Provide an unblockable data channel* that operates in a curtailed networking environment that cannot be blocked without collateral damage to non-CRS traffic.
  2. *Provide user and operator anonymity and deniability* to prevent chilling effects and self-censorship as well as to mitigate coercion.
  3. *Provide resilience* against the censor actively participating in, compromising parts of, removing parts of, and adding malicious resources to the CRS.

Finally we address the desired performance properties, discussed in Section 2.1, and note that these are driven by specific use cases. However, it is clear that if all the properties are maximized, *i.e.* provide the highest level of performance, then all use cases will be covered.

Therefore, the desired *ideal* CRS is one that provides all of the security properties and has **low latency**, **high throughput** and **goodput**, has **general reachability**, and allows **bidirectional communication**. However, in reality we see that there are few CRS designs that provide all of these properties; often there is a trade-off between security and performance. We shall see evidence of this later in Table 2.4 where we summarize security and performance properties of real-world CRS designs. Thus, the types of use cases is driven by the particular security and performance properties that the CRS achieves.

2.6 Censorship Resistance Strategies

Our aim has been to better understand how a CRS and censorship apparatus may interact. We distilled the censorship apparatus to its main parameters and functions (in Section 2.2.1) and we have just consolidated security goals common to all CRSs (in Section 2.5) above. We can now draw out the essence of this interaction and cast it in terms of the interaction between error rates inherent in the detection phase and strategies adopted by the CRS to apply *upward pressure* on them. This way the many CRS designs, with many a varied implementation detail, can be categorized by the manner in which they apply this pressure.

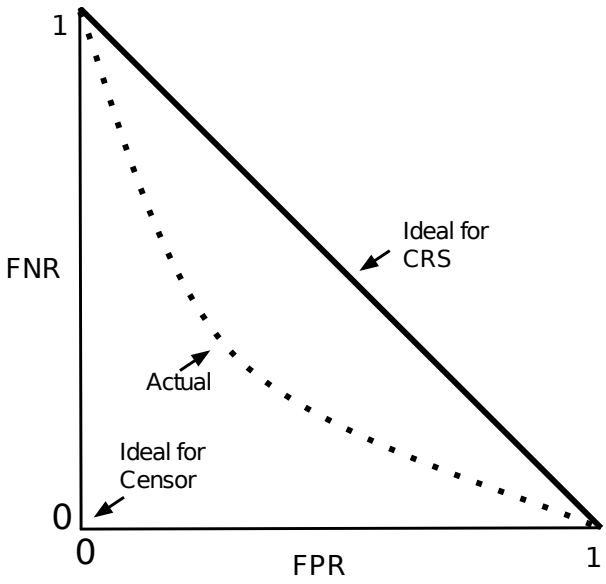


Figure 2.5: Tying strategies to the censor error model. The diagonal line indicates the ideal CRS amount of FNR and FPR and the bottom left corner is the ideal censor amount of FNR and FPR. The dotted curve is the actual censor error rate. The censor puts leftward and downward pressure on this curve while CRS strategies apply or alleviate pressure by moving the curve upwards or rightwards.

Let us clarify what we mean by upward pressure. Consider Figure 2.5 as a graph of the error rates (FNR vs. FPR) of a generic censorship apparatus, specifically the classifier. The axes have been normalized with the extreme ends representing maximum error rate and the origin representing no errors. This graph bears similarity to the more common receiver operating characteristic (ROC) curve; the only change has been to replace the true

positive rate (TPR) with its compliment, FNR. The diagonal line joining the two extreme ends of the axes represents the scenario where the censor is doing no better than random guessing and hence not gaining any benefit from fielding a classifier and the distinguishers it leverages. This is the ideal scenario for the CRS designer. At the other extreme, the curve that hugs the two axes, not drawn for sake of reducing clutter, represents the best outcome for the censor, where they incur no errors at all. The reality is the curve labeled "Actual" that lies somewhere between the two extreme ideal scenarios. With the choice of classification mechanism and distinguishers, the censor chooses where on the curve they wish to be (*i.e.* the level of error they can tolerate) given that they are able to absorb the cost of fielding such a classifier.

CRS strategies aim to push the curve closer to the diagonal (upwards and to the right), the ideal CRS situation, and this is what is intended by upward pressure. Of course, with the censor applying new techniques to move the curve down and the CRS moving it back up, the net effect of the CRS could be to keep the curve in the same place.

When speaking of error rates of detection mechanisms it is important to keep in mind how prevalent CRS activity is as compared to non-CRS activity on a given attack surface. This is the basic idea behind the *base rate fallacy*. The censor, and CRS designer, need to keep in mind that while the apparatus may have a seemingly low error rate, the base rate of CRS activity might be a few orders of magnitude lower and thus the apparatus would nonetheless cause almost all reported positives to be false positives.

We next discuss existing CRS strategies in turn, grouped by the relevant censorship phase and describing how each strategy impacts the censor's error rates and achieves the security goals we outlined in Section 2.5.

2.6.1 Exposure Phase

For designs where CRS information is sensitive, CR strategies that target this phase aim to prevent or slow the censor from learning high-quality distinguishers. The main threat is to the dissemination channel and the CRS information itself. The main challenge is that legitimate users learn CRS information easily while the censor cannot harvest efficiently. In this phase, secret information has already been transmitted through an out-of-band channel, and the CRS client is now ready to engage with the CRS proper.

Unpredictable Values (UV)

An unpredictable selection of distinguisher values, chosen from the possible value space that includes non-CRS elements, ensures that the censor cannot implement an *a priori*

block of those values. Since the censor has yet to discover distinguisher values, there is a window of opportunity where misclassification can occur, thereby temporarily maintaining the FNR. Note that there is no protection from the censor learning that a certain value is CRS-related, and only that the censor does not *yet* know of this relationship.

One method of generating unpredictable values is to recruit CRS participants outside the SoI, who are in control of, or own, particular values of a feature, *e.g.* IP addresses, and utilize them in a seemingly, to the censor, random manner. Another method—where the value space is not restricted—is to adjust the distribution of CRS values across the value space so that there is no pattern to be observed. For example, the CRS could adjust the distribution of packet lengths by randomly padding the packets so that the new distribution has no similarity to the original. In either method, the values can either be used as-is, meaning that if a censor learns them then it knows the true identity and/or location, or they are indirect values, meaning that the true value is obfuscated by a level of indirection and the censor does not learn the true identity and/or location. A simple example of this latter case is a person with a pseudonym in a messaging CRS that provides sender and recipient anonymity. To talk to that person we address our message to their pseudonym, but we do not learn their true name, or network address, or location in real life, assuming that the messaging system is not compromised.

Some CRSs are useful for introducing unpredictable forwarder IP address values for CRS usage. FlashProxy [FHE+12] runs Javascript proxy code within the browser of any website client who visits a CRS-supporting website and makes them a piece of the CRS. It is irrelevant whether or not this CRS-supporting site is blocked by the censor since these temporary forwarders are created *on the client side* by anyone outside the SoI who visits the site. DEFIANCE [LMP+12] utilizes a scheme for hopping among a large pool of IPs, provisioned from cooperating ISPs. Tor [DMS04a] widely recruits volunteers to operate Tor bridge [DM06a] nodes, relays whose addressing information is not shared like ordinary Tor relays, and Psiphon [Psi], a single-hop CRS, purchases IP addresses to be used as forwarder addresses.

Other CRSs are useful for protecting the content location (storage) and covert destination information. Utilizing the Remailer concept based on Chaum's ideas [Cha81], Goldberg and Wagner's Rewebber proposal [GW98] utilizes multi-hop routing for encrypted publication on servers whose identities are hidden behind a pseudonymous name space. This makes it difficult for the censor to block covert destinations since it is difficult to track down the servers they are hosted on. Even if that were easy, it is impossible for the server to know what content it hosts due to encryption and so it is therefore impossible to target specific content. Similarly, Publius [WRC00] and Tangler [WM01] both use this same trick of encrypting all stored content for plausible server deniability properties. The difference

is that in Publius the encryption key is shared among servers using a *k-of-n* threshold secret scheme so that some lost, *i.e.* censored, shares are acceptable. Tangler on the other hand invokes the concept of collateral damage by enforcing, through encryption, that any single document depends on other documents and to remove it would mean the removal of these other documents also. Where the censor has vested interests in certain documents remaining available, this technique provides good protection. Freenet [CSWH00] introduced the use of peer-to-peer (P2P) overlay networking to provide censorship resistant storage of documents. Search and retrieval operations cause redundant copies of documents to be made thus ensuring their long-term availability. The decentralized topology ensures that the censor cannot target the true location(s) of documents and also leverages the fact that most servers will be beyond the censor's influence. Building on top of this, Serjantov [Ser02] introduces anonymous access and the concept of "forwarders" as a defence behind which the "storers" of documents can hide and thus evade the censor. Although not P2P in nature, Tor Hidden Services [DMS04a] also leverage anonymous access to hide the true location and identity of servers from not only the users but also from the network nodes.

Rate Limited (RL)

For the censor who attempts to harvest CRS information by masquerading as a legitimate CRS participant, we can limit the rate at which the censor learns feature-value pairs by adding mechanisms to slow down the dissemination process. This strategy works in concert with the one above by enhancing the security of distinguisher values that are yet unknown to the censor.

The four main methods are proofs of work, partitioning the value-space over time slices, partitioning the value-space over CRS participant attribute(s), and trust-based. Proofs of work include captchas and other puzzles that require the participant to spend some of their resources to gain information. They become onerous when wide-scale harvesting by a resource-bound censor is attempted. Value-space partitions over time slices ensure that even if harvesting is efficient, the censor can only learn values according to the CRS's timetable. Similarly, partitioning over the participant attribute(s) ensures that participants learn disjoint sets over the value-space and hence no one participant, who is restricted in resources (*e.g.* limited IP addresses or low connectivity in a social graph), could potentially learn all CRS information. These mechanisms aim to ensure that there exist windows of opportunity where misclassification can occur and again maintain the FNR for as long as possible.

Tor bridge addresses are made available through the dissemination channel using a release schedule that only reveals a few addresses at a time. The idea is that, despite the censor's efforts, a timed release schedule introduces windows of availability where the censor has not yet harvested and blocked the address.

Feamster *et al.* [FBW+03] utilize *key space hopping* as a means of partitioning the proxy address space in a way that is dependent on attributes of the user, in this case their IP address. This is also leveraged for Tor bridge address distribution. Thus, the adversary needs to masquerade as many users in order to collect the values from the entire address space. To further hinder the censor, we can employ proof of work and life schemes, such as solving puzzles and captchas. Köpsell and Hillig [KH04] limit the rate of effective harvesting of proxy addresses by the censor using a puzzle-based challenge that is presented when requesting a proxy address. This requires the censor to expend effort to obtain proxy addresses, and thus makes both automated and human-driven harvesting more expensive and time consuming.

Proximax [LM11] assigns trust by measuring the safety of distribution channels. Each channel (a user) is assigned a subset of proxy addresses, which are polled for availability. Each channel is expected to redistribute the available addresses to other as-yet-untrusted users. If the proxies remain active then that channel is to be trusted further, but if they become unavailable, or unreachable, then the channel is not to be trusted and further proxy information is not distributed through it. rBridge [WLBH13] utilizes a similar notion where reputation is associated with distribution channels and reputation scores are used for address dissemination. UnBlock [SCL+12] utilizes friend relationships in the Google+ social graph to help clients within the censor's SoI use their contacts outside the SoI as forwarders.

Lock-stepped Interactions (LI)

A censor can actively probe a suspected CRS proxies to confirm CRS participation. To mitigate this threat a CRS can introduce a sequence of interactions to the client-proxy protocol that hides the true nature of the CRS proxy. The aim is to prevent efficient probing by the censor. This strategy is similar to rate limiting with a key difference that this strategy focuses on mitigating attacks that involve censor-CRS interaction over the data channel whereas earlier only the dissemination channel was considered.

The main method is to introduce an authentication mechanism during data channel establishment between the client and overt destination. The properties of the mechanism are similar to those of proofs-of-work mentioned earlier, except the CRS can monitor connections and at first feign non-CRS behavior to confound the censor's probes, *e.g.*

seeing if a particular port is accepting connections. Only after a successful controlled interaction does the CRS expose itself. The assumption is that the censor needs to invest resources to conduct the interaction, and thus makes harvesting less effective, which again enables windows of opportunity during which the FNR can be temporarily maintained.

The CRS can require every connecting client to authenticate to the overt destination. BridgeSPA [SJP+11], ScrambleSuit [WPF13], and DEFIANCE [LMP+12] all require clients to present authentication tokens in order to receive service from the overt destination. These tokens can be distributed through the discovery protocol, albeit in a rate-limiting fashion such that the censor cannot harvest all tokens.

2.6.2 Detection Phase

We arrive at the next phase with the assumption that the censor has failed, fully or partially, to prevent CRS clients from learning CRS information that allows them now to proceed to actually using the CRS. The censor may leverage whatever they have learned about the CRS from the previous phase by inspecting network traffic or from side-channels and is now going to attempt to stop CRS usage.

Strategies here take advantage of the censor's resource limitations by obfuscating low-hanging high-quality distinguishers from the data channel, *e.g.* publicly known IP addresses of CRS-participating overt destinations. The aim is to induce a higher cost to successfully processing traffic on the data channel using the remaining, high-quality but higher-hanging, distinguishers. This is an effective strategy in situations where real-time processing, *i.e.* at network line speed, is required to prevent contemporaneous CRS traffic, or where off-line processing would not yield information useful for future data channel detection. Distinguisher effectiveness is determined by the error rates, FNR and FPR, being low enough to be practical for the censor to utilize. For this phase of censorship, the aim for CRS strategies is to primarily make the data channel undetectable, but some also target other CRS components. The effects of the strategies here boil down to "not looking like a CRS" and "looking like a non-CRS".

An example of this is found in Tor where the 586-byte fingerprint is removed from the network flows by morphing the packet lengths into a non-Tor-like distribution [MLDG12, WPF13] and hence "not looking like a CRS (Tor)". This effectively denies the censor this low-hanging high-quality distinguisher, and forces them to look to hopefully higher-hanging distinguishers.

Obfuscated Values (OV)

This strategy aims to obfuscate distinguishers that uniquely identify the CRS. On the data channel the two main low-hanging distinguishers are network addresses of covert destinations, and the contents of the payload. These can be dealt with using encryption and/or stenographic techniques. The potential FNR is maintained as long as the censor resources are bounded and insufficient for attacking the encryption and steganographic techniques and for leveraging higher-hanging distinguishers efficiently.

Hiding the fact the client is communicating with a covert destination can be provided by manipulating the path the data channel takes. Instead of directly connecting with it, the client instead connects with an intermediary node (the overt destination) and from there is forwarded to the actual desired, covert, destination. Tor [DMS04a] is a popular anonymous communication system that has increasingly been used for censorship resistance since it bounces the data channel over three hops in a manner that ensures that no one node, or observer, can link both client and destination together. In a similar, but restricted variation, Web MIXes [BFK00] utilize cascades—nodes in predetermined and fixed chains—and client- and server-side proxies to provide unlinkable traffic between the user and host. More recently, Decoy Routing [KEJ+11], Telex [WWGH11], Cirripede [HNCB11], and TapDance [WSH14] leverage details of encrypted Internet protocol traffic to send signals to routers en-route to allowed overt destinations to divert the flow of the traffic to censored covert destinations. The only conditions are that 1) the router is on the path to overt destinations that would cause large collateral damage to the censor if they became unreachable and 2) there be no alternative routes that avoid the router. The main differences between the systems are in the encryption schemes used. Decoy routing is based on symmetric encryption while Telex, Cirripede, TapDance are all based on asymmetric encryption. Decoy routing sends a sentinel, a symmetric key the router and user share, along with the TLS client hello message, whereas Telex tags a TLS random nonce using public key steganography and Cirripede does something similar but with the initial sequence number found in the TCP header. TapDance tags the TLS ciphertext itself to achieve the same result. The trick is that the censor is unable to tell normal encrypted traffic from decoy routed traffic but the router on the path is able to do so with little effort. OSS [FNB13] leverages *online scanning services* that allow HTTP redirects, to read data from the covert destination, and use them as a forwarder by embedding data in the URL of the HTTP redirects. Finally, DenaLi [NFS14] uses errors that are pervasive in wifi traffic as a steganographic channel. The key insight is that any node connected to a WiFi access point can be a potential client and destination, and hence the covert destination can remain hidden in plain sight.

Obfuscating CRS traffic requires managing its network-level characteristics. Wiley's Dust [Wil11] removes the fingerprint of Tor's plaintext TLS handshake traffic by encrypting it. It overcomes the censor's ability to fingerprint string patterns, packet lengths, and timing information of a Tor communication stream. ScrambleSuit [WPF13] removes the unique packet size feature of 586 bytes through data padding to match generally occurring packet size distributions, but not any specific protocol or service. Obfs3 [Din12] removes the distinguishing ciphersuite information in the TLS handshake messages by encrypting the entire TLS session.

Indistinguishable Values (IV)

As an extension to the previous strategy, we can deny all high-quality distinguishers (low- or high-hanging) by ensuring that they are indistinguishable from non-CRS distinguishers. The net effect is to make the CRS look like another target non-CRS. There are two assumptions here: one is that the censor must chose to either allow CRS traffic unmolested or ham-fistedly disrupt both CRS and non-CRS traffic since they are impossible to disentangle; the second is that certain distinguishers admit values that can be used for both CRS and non-CRS use simultaneously.

There are two main methods; the first step for each is to identify a non-CRS target distribution for the high-quality distinguisher we wish to remove. In practice, the aim is to ensure that the values of a feature of CRS traffic come from the same distribution as non-CRS values. To generalize, we say that the CRS must conform to the same value distribution as non-CRS traffic for all features. Any deviations from the distribution limits the effectiveness of the strategy. Then, the first variant of the strategy tries to utilize the non-CRS value distributions by adopting them in its design in an attempt at mimicry. The second variant, instead of adopting the values, embeds the non-CRS itself as one of the CRS components, and utilizes it for CRS purposes. The distinction is that in the former the onus is on the CRS design and implementation of all aspects of the data channel to conform to the non-CRS distribution, with the assumption that this is achievable goal. In the latter the trust is shifted to the non-CRS data channel with the assumption that introduction of CRS traffic does not cause a deviation in any non-CRS value distribution. A variation of the latter, where the same distribution is impossible or difficult to obtain, is to alternate between the CRS and non-CRS utilizing the value space, or some subset of it. As long as the censor is unable to tell when the value is CRS and when non-CRS the effect is the same as above.

The following examples adopt the traffic on Tor's data channel (which is low-latency and encrypted) to look like particular other non-CRS targets. SkypeMorph [MLDG12] adopts

Skype's communication protocols and shapes the distribution of traffic characteristics of Tor communications to look like that of a Skype video call. Format transforming encryption (FTE) [DCRS13] transforms Tor traffic to mimic other types of traffic, *e.g.*, HTTP or Javascript. StegoTorus [WWY+12] uses stegonographic techniques to embed Tor traffic within the payload of popular traffic, such as HTTP or JavaScript, to bypass censors. For higher-latency traffic, Collage [BFV10], which extends the basic ideas of Infranet [FBH+02], and Message in a Bottle [IKV12] utilize stegonographic techniques but embed CRS traffic within images that are posted to popular image hosting platforms.

Large Internet companies such as Google, Amazon, and CloudFlare provide cloud-based platforms for hosting services and content. Circumvention systems like GoAgent [goa11] and meek [FLH+15] use the fact that these platforms host multiple services and all are accessible from a common external address. From the censor's perspective the CRS client is connecting to the common platform address, but within the encrypted traffic is the address of the true covert destination or location of content, which the platform knows about. This mechanism is known as *fronting* and is common, and has legitimate uses in shared hosting scenarios. In a similar fashion, CloudTransport uses the storage platform S3 from Amazon as a read and write buffer in the cloud, in order to access the CRS operating on a server elsewhere. From the censor's perspective, the client is only accessing Amazon and not the CRS. Freewave [HRBS12] made use of Skype *super nodes*, which are network nodes that route calls between clients, as proxies to hide the true destination. It also encodes data as audio, in the manner of a acoustic modem, over the Skype audio channel to hide the true nature of the traffic. Facet [LSH14] also utilizes Skype but embeds YouTube videos into the video stream thereby allowing CRS clients to view censored YouTube videos. Recent CRS designs, such as Rook [VK15] and Castle [HNGJ15], leverage online gaming platforms to make use of in-game chat features or game data manipulation as the CRS data channel. In theory, from the censor's point of view, the CRS and the legitimate non-CRS service are indistinguishable with respect to the data channel. This is however quite difficult to achieve in reality and CRSs do trip up on certain aspects of indistinguishability. [GSH13, HBS13]

2.6.3 Response Phase

Designing strategies for this phase is done with the mindset that they should prevent an unwanted response from the censor. Generally, the censor's *response* to the output of the detection phase is to block, interfere with, or allow monitored activities on the attack surfaces; the censor may also decide to devote additional resources analyzing the CRS to

learn more effective distinguishers.[4] As stated in Section 2.5, the first goal of CRS strategies in this phase is to ensure that exposure and detection of data channel distinguishers does not lead to the data channel being blocked. The second goal is to provide plausible client and operator deniability given the exposures and detection the censor has conducted. We note that there are synergies between strategies in this phase and some from earlier phases that are exploited for further CRS effectiveness.

Churned Values (CV)

To mitigate failures at the exposure and detection phases the CRS introduces churn as a defense against the censor acting upon exposed distinguisher values. The idea is that a distinguisher can lose its effectiveness if the values gleaned during the exposure phase are no longer valid, and/or in continual flux. A CRS design can introduce churn by using values only for a short lengths of time, or using them once. The censor needs to continuously employ resources at the exposure phase for yet-unknown and currently employed CRS values that maintain the FNR. The assumption is that the censor would prefer to find a stable and consistent distinguisher that can be employed for a long time, such that the costs of the exposure phase are amortized over the time the CRS is in play.

The usual method adopted by the circumventor is to provision a large pool of values and then unpredictably utilize them and then expire them within short time frames. An added CRS element that manages the transitions is also required. The contrast between rate limiting and value churn strategies is that in the former the censor's knowledge is cumulative, whereas in the latter the censor's knowledge is quickly outdated and acting on it would come at a cost.

To ensure access to the CRS, FlashProxy's forwarders are temporary and *last only for as long as a website client is viewing the website*. DEFIANCE's pool of IPs only serve as CRS forwarders for *brief periods of time*. We note that value churn in the case of FlashProxy is dictated by the website visitors, and hence extrinsic to the CRS, whereas DEFIANCE manages the churn rate itself, through the dissemination channel. Churn also occurs naturally but we do not consider it a factor since it is neither assured that it will be quick enough, nor can there be control of how values are reused.

[4]These actions are contingent upon the knowledge the censor has about the accuracy of the detection mechanism, together with the costs of false negatives and false positives. These are costs that economic game theory may better explain, which we focus on in Chapter 3.

Outside the Scope of Influence (OSoI)

CRSs can leverage inherent limitations to the extent of the censor's scope of influence, as outlined in Figure 2.2, and place critical components that are susceptible to attack beyond it. In this way the censor can know what to attack, but is unable to do anything to it; contrast this with the indistinguishable values strategy from the previous phase, in which the censor may only know that the non-CRS is being used for some CRS activity but cannot distinguish CRS from non-CRS uses. Indeed, so central is this strategy that, with a few exceptions, all CRSs employ it to some extent.

The main manner most CRSs employ this strategy is to place or utilize overt destinations and CRS information outside the SoI, *e.g.* national or jurisdictional boundaries, to which CRS clients are able to establish data and dissemination channels respectively.

To ensure availability of content and the CRS, Anderson's Eternity Service [And96], one of the earliest CRSs, leverages distribution of data across a large number of overt destinations deployed in diverse jurisdictions, so that at least some servers will be outside any given censor's scope of influence. Back's Usenet Eternity [Bac97] is a 1997 implementation of this proposal. Indeed, many of the CRSs we have come across in the literature employ the OSoI strategy. We can make one distinction, however, which is that some CRSs take advantage of existing *CRS-agnostic* third-party infrastructure or services instead of deploying their own *CRS-specific* infrastructure. Cloud providers like Google, Amazon, and CloudFlare, or VoIP providers like Skype, or content hosts like YouTube and Flickr, are all CRS-agnostic platforms that have been leveraged for their OSoI properties.

2.7 Attack Mitigation and Remaining Gaps

Let us now consider how each CRS component is protected by the strategies we have identified. For each strategy we distill the main techniques used to realize them. Ideally, there should be mitigations for all of the different censor attacks. We note the gaps and discuss the properties of potential strategies that could fill them. Refer to Table 2.3 for a handy reference for the discussion that follows.

2.7.1 CRS Information and the Dissemination Channel

We consider the CRS information and its dissemination channel together since they logically depend on each other. The primary target of the rate-limiting strategy is to combat

Table 2.3: A drill-down of Table 2.1, showing techniques employed to defend against censorship threats to CRS components. The CRS components we described in Section 2.3.1 and the attacks on each, as discussed in Section 2.4, are organized along the top. The rows are grouped by the phases of censorship described in Section 2.2.1. Each group is organized by the strategies that apply to that phase and the techniques that realize them, both discussed in Section 2.6. A solid black square indicates that this technique has been applied to this attack vector.

			CRS Component															
			CRS Info.		Diss. Chan.		Data Channel			Overt/Covert Destination					CRS Client		
Phase (Strategy)	**Technique**	**Attack**	Harvest	Poison	Deny	Masquerade	Detect	Deny	Disrupt & Degrade	Comb	Coerce	Crush	Curtail	Corrupt	Compromise & Deny	Coerce	Link
Expose (UV)	Actual Value												■				
	Indirect Value									■	■		■				■
Expose (RL)	Time		■										■				
	Value-Space		■										■				
	Proof-of-Work		■			■							■				
	Trust		■			■							■				■
Expose (LI)	Client Authentication					■				■							
Detect (OV)	Traffic Patterns						■										■
	Covert Destination/Content						■										■
Detect (IV)	Adopt						■										■
	Co-opt						■										■
Respond (OSoI) (CV)	Resource-Driven		■		■			■		■			■				
	CRS-Driven		■		■			■		■			■				
Respond (OSoI)	CRS-Specific							■			■					■	
	CRS-Agnostic				■			■			■			■		■	

the censor—masquerading as a CRS participant—from efficiently harvesting CRS information. However, there are no strategies that mitigate the threat of the censor poisoning CRS information. As noted earlier, it is not without risk for the censor to manipulate public information used by both CRS and non-CRS activity, so there is at least an implicit level of defense. Denial of the channel could be mitigated by the same strategies that secure the data channel, where unpredictable and indistinguishable values make detection difficult.

2.7.2 Data Channel

Obfuscated and indistinguishable values certainly make it difficult to detect the data channel. Channel denial is mitigated by the success of these strategies as well as the strategy of placing overt and covert destinations outside the scope of influence. However, there are limits to the success that depend on how well these strategies are implemented. Transforming traffic to look like another protocol, such as HTTP, generally has only limited success [HRBS12, GSH13]. For instance, in the case of mimicry to obfuscate known distinguishers, the censor only has to find one disparity, whereas a CRS must perfectly imitate the chosen cover protocol in order to succeed. Cover protocols are generally complex, with behavior dependent on their particular use cases. An imitator has the task of not only making the protocol look correct, *i.e.*, matching explicit values, but also ensure it behaves according to expected norms, *i.e.*, matching implicit values. Common protocol-level disparities are a result of incomplete or incorrect cover protocol implementation, such as failure to handle errors in a consistent manner. Both SkypeMorph and CensorSpoofer suffer from systematic errors stemming from incomplete imitation of the cover protocol [HBS13, GSH13, LSH14].

Even if CRS traffic is tunneled over the cover protocol, to avoid the problems inherent to mimicry, the censor may be able to take advantage of *channel usage inconsistencies* and *content inconsistencies* [LSH14]. A CRS may rely on channel characteristics in a different manner from the cover protocol. If the overt protocol is more robust to network degradation, for example, the censor can manipulate the network to disrupt CRS traffic, but not legitimate cover protocol traffic. Iran conducted such an attack on Tor by limiting the duration of TLS connections to two minutes. [Tor13] Legitimate connections, to text-based websites, were not affected by this since the website has loaded within that time period. On the other hand, Tor traffic is interrupted, as Tor TLS connections are longer lived than two minutes and would need to be reestablished often. In a similar vein, Iran also throttled TLS connections to 2 kilobits per second rates, making browsing and other activities difficult [Lew11]. These attempts are to block CRSs that do not perfectly match the use cases of the popular protocols they tunnel through, which has the effect of making the usage

of the CRS onerous and thus discouraging it. Other examples include SWEET [ZHCB13] and Freewave [HRBS12]. SWEET uses email as the carrier for web traffic. Since email is a high-latency tolerant protocol and web traffic is not this disparity can be exploited by the censor—who can simply delay all emails leaving the SoI—to impact only the CRS and not the carrier protocol. Similarly, since streaming audio/video is loss tolerant, the censor can disrupt CRSs like Freewave by dropping enough packets to disrupt Freewave transmissions, while leaving actual Skype traffic within the threshold of acceptable level of performance.

Even if the data channel is encrypted, the content of CRS communications may be distinguishable from the content of the cover protocol. For example, Li *et al.* [LSH14] and Geddes *et al.* [GSH13] show that Freewave may be detected using an n-gram based classifier on packet lengths. This attack, however, is contingent on the censor accurately modeling normal content. This may be easier for special-purpose protocols, such as VoIP, which is used only for video or voice traffic, than for general-purpose protocols, such as TLS, which is used for multiple types of data. Achieving indistinguishability of CRS traffic from legitimate, popular cover protocols has been a natural focus in the research literature. In light of difficulties ensuring consistency at the protocol, channel, and content levels, Li *et al.* propose that CRSs not only use popular, unblocked cover protocols to tunnel traffic, but also to match CRS content with that of the chosen cover protocol. Their proposal, Facet, is a prime example of a system that attempts to do this, by building a data channel that sends video traffic over Skype, and ensuring the video traffic approximately matches the expected traffic pattern for a video chat call. It remains to be seen if further discrepancies may be discovered with this approach.

2.7.3 Overt and Covert Destinations

Rate-limiting strategies help to mitigate the impact of the censor curtailing access to overt destinations. The censor combing for CRS-related values is potentially mitigated by the lock-stepped interaction strategy. Coercion resistance is provided by, again, placing CRS-participating components outside the SoI. Hiding the true location or nature of covert content, using the strategy of unpredictable values, also mitigates the threat of the censor coercing content from being taken down.

Decoy routing CRSs [KEJ+11, WWGH11, HNCB11, WSH14] are an interesting attempt at obfuscation that has attracted research attention. They work by placing CRS *stations* within the network infrastructure itself, such as at routers at participating ISPs outside the censor's SoI. Users signal their intent to use the CRS by steganographically tagging a seemingly innocent connection to a *decoy destination* (or dummy destination in

our terminology), which can be any site not blocked by the censor and that has a CRS station on the network path between the client and itself . The user may then request her real, blocked destination, which will be served by the CRS station. The main difference between these systems lies in their ability to handle asymmetric flows, a feature of Decoy Routing, Cirripede, and TapDance, and their reliance on an inline network-flow-blocking element, which is necessary in all of these systems except TapDance. All of these systems rely on strategic deployment, making it impossible for the censor to "route around" cooperating ISPs without significant collateral damage. If this assumption is invalid, however, the censor can avoid or otherwise blackhole the route. The tools for this are already present in networking equipment; the question is if an alternative route exists for desirable hosts on the other side of the proxy. Thomson *et al.* [SGTH12] investigate the feasibility of this attack while Houmansadr *et al.* [HWS14] evaluate the costs associated with it. This is an open research problem, with game-theoretic analysis as an avenue for further pursuit.

However, gaps exist. The first is that most CRSs do not yet attempt to prevent crushing tactics (such as a DDoS) against CRS-participating destinations or storage. The CRS implicitly depends on the leveraged platform or their network connectivity provider to prevent such a scenario. In general, preventing an DDoS attack does not yet have a robust solution, outside of over-provisioning of bandwidth and IP addresses. The second gap is that corrupted content and CRS participants are not prevented or mitigated by any strategy. Both of these areas are avenues for future research.

2.7.4 CRS Client

The linking attack is the main vector that is mitigated in the exposure and detection phases. Indirect values, trust, obfuscated and/or indistinguishable traffic patterns and destinations are the main techniques employed. We have already discussed the relevant examples from the literature in the discussion about overt/covert destinations and the dissemination channel.

In the response phase, publisher coercion may be mitigated by CRSs that do not allow the deletion or modification of published content. Alternatively, plausible deniability may deflect suspicion, as seen in DenaLi [NFS14], where errors in broadcasted messages on the local WiFi network are used to hide steganographic messages and CRS participation.

2.7.5 Mitigation Summary and Trends

In general, there is almost complete coverage across all the phases and attack vectors, save for the gaps we have identified above. Looking at the attack coverage across the censorship

phases we note that there is good representation for response mitigation. What is striking is that the technique of using CRS-agnostic infrastructure, services, and software provides protection against more attacks than the technique of using CRS-specific ones. The main reason is due to the potential collateral damage that would be incurred should the censor interfere with CRS-agnostic platforms and services which also serve non-CRS purposes. Also, CRS-agnostic software can provide compromise & denial protection because it is popular and likely already present on the client's computer. The value churn strategy protects many of the components with its two techniques both providing similar attack coverage. Unpredictable indirect values are the most useful form additionally providing mitigation for combing, coercion, and client linking.

We observe that there is a trend to raise the bar for the censor to detect the data channel, by mitigating the low-hanging distinguishers. However it is unclear what the capabilities of the censor are with regard to the mitigated low-hanging distinguishers and if the censor is even willing to engage in sophisticated traffic analysis. While it is difficult to find out the true capabilities of a censor, it is perhaps still useful to find out how difficult attacking lower- and higher-hanging distinguishers actually are. While the literature does provide analysis of attacks they are usually tuned against particular CRS implementations and the results are of limited scope and do not tell us if they are applicable to a realistic censor. This is an avenue of research where studies of distinguisher effectiveness at Internet scale could provide clues about which distinguishers are the most significant threats.

Another trend that emerges is that many recent CRSs leverage existing third-party infrastructure and data channels to defend against the threat of blocking. Indeed, it seems as if the indistinguishable values strategy has become the more favored one over obfuscating and unpredictable values. The trend seems to indicate that CRS data channels are moving from being CRS-specific to those that are CRS-agnostic. What this means is that whereas CRS-specific implementations are crafted to meet all CRS design specifications, CRS-agnostic implementations meet those same CRS specifications through happy coincidence. This leads us to conclude that there are extrinsic properties that need to be accounted for when evaluating such a CRS. This could be undesirable for three reasons. The first is that it introduces dependencies on external parties that may not be invested in the CRS's success. The second is that the synergy that existed to allow the CRS to leverage a particular extrinsic property is not by design, which means that there may be—from the perspective of the CRS—unintended states that could render the CRS ineffective. Finally, the fate of the CRS is tied to the leveraged service and how widely it is adopted and spreads and this limits the potential client base of the CRS. A closer look at these issues is needed to evaluate how critical they are and perhaps also how CRS designs can be adapted to provide more control over extrinsic properties.

2.8 Revisiting Use Cases—Security and QoS

We now reconsider CRS design from the perspective of the users and use cases, and the security and performance properties we first defined in Section 2.1. Table 2.4 provides a reference for and summary to the discussion to follow. Our aim is to see what properties are being satisfied with actual CRS designs and also to note trends, threats, and possible synergies.

In this table we do not include a column for unblockable data channel since we conclude that all CRS designs in the literature provide circumvention through some means, which have already been discussed in Section 2.7. Instead, in the table we focus on those properties that help us distinguish CRS designs from each other.

Overall, we note that CRS designs have little synergy with each implementing its own designs from scratch, most often with tightly integrated functionality; this precludes other systems from leveraging the design and effort already invested. Even designs that share a common base suffer from this shortcoming. A prime example is the Tor network and the various solutions for ensuring its data channels are undetectable. Although the Tor community is actively trying to address the problem with the pluggable transport framework, the desired level of reuse and modularity is not currently in place.

Only a few CRSs provide publisher anonymity, and the majority of those originally debuted in the early 2000's, or if not they leverage those early systems for this property. It seems that recently the emphasis has shifted to providing read access whereas earlier the specter of chilled speech was at the forefront. Similarly, protection against publisher coercion is only provided by earlier systems. In contrast, we note that recent systems, particularly those that leverage CRS-agnostic platforms, actually expose publishers. It becomes clear why this is the case; all of these systems require the publisher to interact directly with the CRS, either to set up accounts and/or store and manipulate files. This is not a consequence of some inherent limitation of CRS-agnostic or co-opted designs but rather due to this not being a design goal. This particular limitation can be addressed by incorporating the anonymizing and coercion resistance techniques used in earlier systems, or by utilizing those systems directly.

More generally, the trend for direct communication between clients and overt/covert destinations, eschewing path obfuscation means that clients can be linked to their communication partners. The reason is that these systems are only designed to bypass blocking, and are not intended to provide other security properties. This situation can be ameliorated with the use of an appropriate path obfuscation design; certain CRS designs (*e.g.* Tor or Web MIXes) that provide publisher anonymity can be readily utilized for this purpose. However, these designs do enjoy good performance across all the measures we have

Table 2.4: Overview of security and performance properties of various CRS designs and the strategies they use to achieve them. White squares denote that the system does not provide that security property or does not use that strategy, while black squares mean that it does. Dashes denote that the security property does not apply to that particular system. The systems are grouped by the similarity of the strategies they mainly leverage. CRSs that utilize steganographic techniques are labeled with †, and those that leverage CRS-agnostic platforms are labeled with ∗.

UV	RL	LI	OV	IV	CV	OSoI	System	Year	Undetectable Data Channel	Anonymous Publisher	Unlinkable Client-Dest. Path	Incoercible Client-Dest. Path	Incoercible Publisher	Deniable Covert Dest.	Deniable Client	Deniable Forwarder	Deniable Covert Dest.	Reachability	Interactivity	Throughput	Goodput	Latency
■	□	□	□	□	□	■	**TAZ-Rewebber**	[GW98] 1998	□	□	■	■	■	□	□	□	□	■	■	■	■	■
■	□	□	□	□	□	■	**Freenet**	[CSWH00] 2000	□	■	■	■	■	□	□	□	□	□	□	□	□	□
■	□	□	□	□	□	■	**Publius**	[WRC00] 2000	□	□	■	■	■	□	□	■	□	□	□	□	□	□
■	□	□	□	□	□	■	**Tangler**	[WM01] 2001	□	□	■	■	■	□	□	■	□	□	□	□	□	□
■	□	□	□	□	□	■	**Serjantov**	[Ser02] 2002	□	■	■	■	■	□	□	□	■	□	□	□	□	□
■	□	■	■	□	□	■	**Tor-Hidden Services**	[DMS04a] 2004	□	■	■	■	■	□	□	□	□	■	■	■	■	■
■	■	□	□	□	□	■	**Tor Bridges**	[DM06a] 2006	□	■	■	–	–	□	□	–	■	■	■	■	■	■
■	□	□	□	□	■	■	**FlashProxy**	[FHE+12] 2012	□	■	□	–	–	□	□	–	■	■	■	■	■	■
■	■	■	□	□	■	■	**DEFIANCE**	[LMP+12] 2012	□	■	□	–	–	□	□	–	■	■	■	■	■	■
□	■	□	□	□	□	■	**Untrusted Messenger**	[FBW+03] 2003	□	■	■	–	–	□	□	–	■	■	■	■	■	■
□	■	□	□	□	□	■	**Köpsell and Hillig**	[KH04] 2004	□	■	■	–	–	□	□	–	■	■	■	■	■	■
□	■	□	□	□	□	■	**Unblock∗**	[SCL+12] 2012	■	–	□	–	–	■	□	–	■	■	■	■	■	■
□	□	■	□	□	□	■	**BridgeSPA**	[SJP+11] 2011	–	–	–	–	–	□	■	–	–	–	–	–	–	–
□	□	■	■	□	□	■	**ScrambleSuit**	[WPF13] 2013	■	–	–	–	–	□	□	–	■	■	■	□	■	■
□	□	□	■	□	□	■	**TriangleBoy**	[Hsu00] 2000	□	–	□	–	–	□	□	–	■	□	■	■	□	□
□	□	□	■	□	□	■	**Web MIXes**	[BFK00] 2000	□	■	■	–	–	□	□	–	■	■	■	■	■	■
□	□	□	■	□	□	■	**Tor**	[DMS04a] 2004	□	■	■	–	–	□	□	–	■	■	■	■	■	■
□	□	□	■	□	□	■	**Dust**	[Wil11] 2010	■	–	–	–	–	□	□	–	■	■	■	□	■	■
□	□	□	■	□	□	■	**COR∗**	[JACF11] 2011	□	■	■	–	–	□	□	–	■	■	■	■	■	■
□	□	□	■	□	□	■	**Telex**	[WWGH11] 2011	■	–	□	–	–	■	□	–	□	■	■	■	■	■
□	□	□	■	□	□	■	**Decoy Routing**	[KEJ+11] 2011	■	–	□	–	–	■	□	–	□	■	■	■	■	■
□	□	□	■	□	□	■	**Cirripede**	[HNCB11] 2011	■	–	□	–	–	■	□	–	□	■	■	■	■	■
□	□	□	■	□	□	■	**Obfsproxy**	[Din12] 2012	■	–	–	–	–	□	□	–	■	■	■	■	□	■
□	□	□	■	□	□	■	**OSS∗**	[FNB13] 2013	■	–	■	–	–	■	■	–	■	■	□	□	□	□
□	□	□	■	□	□	■	**TapDance**	[WSH14] 2014	■	–	□	–	–	■	□	–	□	■	■	■	■	■
□	□	□	■	□	□	□	**DenaLi†**	[NFS14] 2014	■	–	■	–	–	■	■	–	□	■	□	□	□	■
□	□	□	□	■	□	■	**Infranet†∗**	[FBH+02] 2002	■	□	□	□	■	■	■	■	□	□	□	□	□	□
□	□	□	□	■	□	■	**Collage†∗**	[BFV10] 2010	■	□	□	□	■	■	■	■	□	□	□	□	□	□
□	□	□	□	■	□	■	**GoAgent∗**	[goa11] 2011	■	–	□	□	■	■	■	□	■	■	■	■	■	■
□	□	□	□	■	□	■	**StegoTorus†**	[WWY+12] 2012	■	–	–	–	–	□	□	–	■	■	■	□	■	■
□	□	□	□	■	□	■	**Freewave∗**	[HRBS12] 2012	■	–	■	–	–	□	■	–	■	■	■	□	■	■
□	□	□	□	■	□	■	**SkypeMorph**	[MLDG12] 2012	■	–	■	–	–	□	□	–	■	■	■	□	■	■
□	□	□	□	■	□	■	**CensorSpoofer**	[WGN+12] 2012	■	–	□	–	–	□	□	–	■	□	■	■	□	□
□	□	□	□	■	□	■	**MIAB†∗**	[IKV12] 2012	■	□	□	□	■	■	■	■	□	□	□	□	□	□
□	□	□	□	■	□	■	**SWEET**	[ZHCB13] 2013	■	–	□	–	–	■	□	–	■	□	■	■	■	■
□	□	□	□	■	□	■	**FTE**	[DCRS13] 2013	■	–	–	–	–	□	□	–	■	■	■	□	■	■
□	□	□	□	■	□	■	**Marionette**	[DCS15] 2015	■	–	–	–	–	□	□	–	■	■	■	□	■	■
□	□	□	□	■	□	■	**CloudTransport∗**	[BHS14] 2013	■	–	□	–	–	■	■	–	■	■	■	■	■	■
□	□	□	□	■	□	■	**Facet∗**	[LSH14] 2014	■	–	□	–	–	□	□	–	□	□	■	□	■	■
□	□	□	□	■	□	■	**Facade†**	[JBF+14] 2014	■	□	□	□	■	■	■	■	□	□	■	□	□	□
□	□	□	□	■	□	■	**meek∗**	[FLH+15] 2015	■	–	□	□	■	■	■	□	■	■	■	■	■	■
□	□	□	□	■	□	■	**Castle†∗**	[HNGJ15] 2015	■	□	□	□	■	■	■	■	□	■	□	□	■	■
□	□	□	□	■	□	■	**Rook†∗**	[VK15] 2015	■	□	□	□	■	■	■	■	□	■	□	□	■	■
□	□	□	□	□	□	■	**Eternity**	[And96] 1996	□	□	□	■	■	□	□	□	□	□	■	■	■	■
Strategies									**Security**									**Performance**				

defined and this indicates that there are security-performance trade-offs to be considered, that depend on the use case and CRS design adopted.

The trend of forgoing client security in favor of performance depends on an assumption that it is not realistic to assume that the majority of clients will be targeted or harmed. However recent revelations, in particular those by Edward Snowden [Pau13], call into question these design choices. The dramatic reach of the "Five Eyes"—a program of cooperation and surveillance data sharing between the governments of Australia, Canada, New Zealand, the United Kingdom, and the United States of America—necessitates a reevaluation of the basic assumptions most CRSs make with respect to the censor's sphere of influence and visibility. Systems are typically not designed to withstand global passive adversaries, for instance, and designs often count on distribution across diverse jurisdictions to make this assumption realistic. Given that a significant number of government entities cooperate and have far-reaching network capabilities, systems designed with these assumptions may be more brittle than previously believed. The recent seizure of several Tor hidden services [Lew14], for example, may be evidence of this level of global cooperation. While it is unclear just how large the spheres of visibility and influence are for any given censor, their reach is likely far larger than anticipated by current CRS designs.

Taking another look at the high-performance designs we identify an aspect worth noting. While in all of these designs the potential throughput is comparable to that of unfettered Internet access, in reality only those designs that leverage CRS-agnostic platforms and services—which are provisioned for Internet-scale performance—are likely to actually enjoy this level of performance. The other systems leverage individual nodes on the Internet normally not resourced for high throughput and thus are more likely to only achieve middling levels of performance. We will revisit this dichotomy later in the following section when we consider the implications of utilizing CRS-agnostic entities in CRS design.

Overall we note the lack of an all-in-one CRS design that provides all of the security properties we have identified. If we were to consider hybrid designs then we see that recent CRS-agnostic designs that provide undetectable data channels coupled with the early publishing designs may provide all of the desirable properties. Unfortunately, we note the stark contrast between the performance profiles of both these types of designs which may indicate a lack of synergy and the need for more research and thought in how to overcome any inherent limitations. A more performance-profile synergistic match seems to be CRS designs leveraging steganographic channels for access to materials on popular platforms and the early publishing designs as above. The similarity in performance properties indicates that these should be relatively straightforward to combine. Indeed, we note that in all cases if the publisher and client were to access the popular platform—which gives plausible deniability of CRS participation—through an anonymity CRS, say

Tor, then all of the security properties would be covered. Alas, this hybrid would be on the low end of the performance spectrum. If we allow ourselves to relax one security property, such as coercion resistance, then higher-performance hybrids are apparent, *e.g.* Tor hidden services and Obfsproxy, which is likely a combination in use today.

2.9 Revisiting Collateral Damage

We now take a closer look at the philosophy of collateral damage and the part it plays in the design of data channels and the selection of overt destinations for CRSs. The theory goes: if we pick the perfect non-CRS service or platform to leverage, one that the censor would balk at blocking due to the inherent collateral damage that is contingent on the limitations of its classifier, then CRS activity would go unmolested. For example, we see this with CloudTransport which leverages the Amazon cloud storage service as the non-CRS to use as cover. If we are able to perfectly blend in, using indisguishable values for example, then the censor is unable to cleanly remove (all) CRS activity, and thus we force the censor to tolerate the CRS activity. What this means is that the CRS is causing the non-CRS services and platforms to act as concentration points for maximizing collateral damage potential.

Counter-intuitively, the censor may actually benefit from this concentration since the attack surfaces and CRS security failures are also concentrated and now well defined, *e.g.* only traffic to Amazon's cloud storage service, and hence easier to deal with (contain). We present three illustrative points where non-CRS service or platform concentration produces negative outcomes.

The non-CRS service or platform is now a single point of failure, which means that if the censor does decide to block it, perhaps because the CRS designer overestimated the cost of the potential collateral damage or despite it, the CRS is effectively contained while the impact is limited to the non-CRS service or platform only.

Local entities, or the censor itself, may develop local alternatives for the targeted non-CRS service or platform and draw legitimate non-CRS service/platform users away, leaving CRS users exposed. There are many instances of local alternatives such as Weibo, YouKu, and Baidu.[5] These local alternatives have the added benefit of being better censored since they sit inside the SoI and thus the censor can nullify any potential CRS usage of the local

[5]We do not claim that these alternatives are a direct effort to quash CRS activity, but these alternatives certainly do diminish the potential collateral damage since the non-CRS service/platform user base has been thinned.

alternative while minimizing the impact to availability of the service/platform to regular users.

Usually non-CRS services and platforms that have been leveraged in the past are operated by single corporations. The censor can strike back at CRS systems by attacking the non-CRS service or platform and/or the entity that operates it. The strike can be in the form of actual network-level attacks [MWD+15] that cause the operator to reconsider, or decry [DB15], its role as a host to CRS activity. Indeed, there is uncertainty around exactly how the operating entity may respond. In the worst case it may even act as an informant to the censor and monitor CRS activity. This is most troubling when we recall that many high-performing CRS designs that utilize these platforms do not provide protection against client-destination linkage. Since the long-term viability of leveraging third parties for CRS duty is unclear,[6] it becomes critical that these security properties actually be in place if non-CRS services are leveraged.

The picture this paints is that concentration is a poor direction since it has synergy with the censor's desire for containment. We should reexamine the trend of leveraging non-CRS entities and platforms as a shield since this adds extrinsic factors into the CRS-design. As we see from the discussion above, if collateral damage is to remain a deterrent then it must not come from the concentration provided by an easily contained entity.

2.10 Conclusions

We conclude this chapter by identifying areas for future work that have been illuminated by our discussion so far.

We identified that reuse and modularity were lacking and identified Tor pluggable transports as one effort to address this shortcoming for the Tor ecosystem. The current state is not at the desired level but there is progress in the right direction. Fog [ifi13b] attempts to create a platform for pluggable transport component composability, but this effort has stalled [ifi13a]. The greatest challenge has been a correct decomposition of CRS component functionality, such that components are both composable and consistent with the constraints of Internet networking. Jumpbox [MMBY14] alleviates, but does not solve, the problem at the network interface layer, by providing a standard interface for encapsulating pluggable transport traffic to look like regular web traffic. Khattak *et al.* [KSM14] provide a systematization of extant pluggable transports and recommend the notion of a "tweakable transport" as a possible way forward.

[6] In that, it is not yet known if public opinion will sway platform providers to protect CRS activity or if business concerns will make CRS activity unwelcome. Recent events do not look promising.

We noted the rising trend of leveraging popular CRS-agnostic services and platforms as a means of mitigating censor actions through collateral damage in the response phase. We noted that these tend to concentrate the attack surface and that this helps the censor better contain the CRS. We propose *diffusion*, which is the return to distribution of risk over many entities, stakeholders, and attack surfaces.

Practically, diffusion means that CRS designs ought to leverage ubiquitous Internet protocols, *e.g.* TLS, and network topologies, *e.g.* peer-to-peer, that avoid single points of failure. Diffusion will also avoid providing a constrained set of stakeholders, since ubiquitous technologies belong to no one and everyone, for the censor to target and pressure. The end result would be that the censor should have a difficult time containing CRS activity to some portion or subset of the network value space, and mitigate the impact and extent of the potential collateral damage.

Our attention is also drawn to the lack of CRSs designed to enable free communication among users within a censor's SoI. Anderson's Eternity Service set the initial direction for considering censorship circumvention by assuming the existence of some entity beyond the censor's SoI who could develop and deploy CRSs. It is increasingly becoming realistic to expect, however, that users never egress the censor's sphere of influence or visibility. Regimes have been known to shut off Internet access completely for periods of time, including Egypt and Libya; [DSA+11] enabling citizenry to organize and communicate using the internal network during such times becomes an important use case. Unfortunately, there has been little research attention given to developing censorship circumvention systems entirely contained within the censor's SoI. DenaLi [NFS14] is a notable exception but it is designed only for operation within a single LAN and, as a result, does not provide many desirable user experience properties. However, it does show that there are viable solutions for CRS schemes that operate within the censor's SoI. Following the principle of diffusion is key in this scenario since there is no non-CRS service or platform to leverage within the SoI.

While we develop defenses within the framework of the current Internet architecture, we, like some before us [TS14], also call for a next-generation Internet designed with censorship resistance as an explicit feature by default.

An active research avenue focuses on the need for empirical data about real-world censorship with respect to CRS user impact and the censorship techniques used [BF13]. What few investigations exist are informative, but provide only partial and possibly outdated snapshots of the state of censorship. In particular, many CRS designs assume the censor is capable and willing to engage in sophisticated traffic analysis, as we noted in Section 2.7.5. As discussed earlier, a thorough analysis to evaluate censor attacks on distinguishers is needed.

We also lack research into the relationship between the many CRS techniques and the costs to the censor, particularly in terms of policy and decision making. This chapter examines and categorizes CRS techniques based on qualitative technical measures and makes relative comparisons amongst the various strategies and techniques. A game-theoretic analysis of optimal strategies would be helpful in understanding the censor decision function and identify useful techniques for CRS designs.

Furthermore, there is an implicit assumption that collateral damage is an overwhelming factor in the censor's decision function. Current CRS schemes are contingent on and try to maximize collateral damage (FPR), but do not evaluate the impact of information leakage (FNR) on censor behavior nor identify parameters that might affect it. Filling this gap is an important next step in illuminating the dynamics of the censorship resistance game and providing feedback on best practices for CRS design.

Indeed, we pursue the last two areas of research in Chapter 3, where we apply game-theoretic analysis to censorship resistance and investigate the impact of information leakage, collateral damage, and accuracy of the censorship apparatus on the censor's behavior.

Chapter 3

Game-Theoretic Approaches to CRS Design

3.1 Introduction

In Section 2.10 we noted a lack of insight into the censor's decision function since, so far, the literature has treated that aspect of the censor as a black box. In this chapter we investigate this aspect of censorship through the lens of game-theoretic analysis. Since the problem space is large, we reduce our scope to the data channel and defenses against detection of CRS-related traffic. This is timely because there is currently a lot of activity within the community to develop better designs and implementations that address censorship threats to the data channel. Specifically, we seek to understand how the error rates of the censorship apparatus, both FPR and FNR, affect the censor's behavior and if, and how, the base rate of covert traffic can be used as a parameter in CRS designs.

Game theory is the study of how groups of rational, self-interested entities behave in response to one another's actions. In the context of censorship-resistant communications, a game-theoretic approach can be used to assess the optimal behavior of a rational censor and the designers of a CRS.

To facilitate this, we will analyze the behavior of the two parties, or *players* from now onwards, in increasingly detailed versions of an abstract "censorship game", designed to capture the fundamentals of the censorship resistance dynamics, while still being simple enough to readily analyze. This serves to reveal the essential components of the problem domain.

These players try to maximize their benefits by thinking strategically about their actions, using information that they have about the environment and the other players. A central assumption is the theory of "rational choice", which states that an entity seeks to maximize its utility independent of the other player's utility and will choose an action that is at least as good as any other action available to them. The utilities can be modeled by a utility function (U) that assigns ordinal values to the utilities. That is, if a player prefers outcome a over outcome b and outcome b over outcome c then the utilities are ordered $U(a) > U(b) > U(c)$.

Technological Limits

Recall from Section 2.2.1 that the censorship apparatus is limited by shortcomings of the classifier, the computational and memory costs of real-time processing, and the partial view of the attack surface, amongst other considerations. It is important, then, to take into account the rate at which objects of interest are misclassified. The two types of errors—false positives and false negatives—govern the confidence the censor has in their censorship apparatus. The prevalence of each of these type of errors provides an important input for both the censor and the circumventor in defining their respective strategies.

False Positives

From the censor's perspective, false positives are the legitimate traffic, and users, that were misclassified and blocked—the *collateral damage*. The censor naturally seeks to keep this as low as possible.

As we have noted earlier in Section 2.7.5, the collateral damage strategy has been leveraged by numerous censorship resistance systems. However, in most cases the circumventor assumes an all-or-nothing approach to censorship, which can be limiting when the censor is content with partial blocking.

False Negatives

The censor tries to prevent as many clients, or as much traffic, as it can from circumventing its blocks—termed *information leakage*. Due to the limits of technology it is unable to identify all of them.

The circumventor's aim is always to have as much, if not all, of its traffic classified as a (false) negative. Strategies to obfuscate distinguishers or make them indistinguishable from

non-CRS traffic, as well as steganographic and encryption techniques are all instrumental in achieving this goal.

We note here that since the circumventor is a rational player its aim is not to produce collateral damage, or indeed to explicitly reduce the censor's utility. It is only concerned with maximizing its own utility, independent of the censor utility.

3.2 Censorship Games

In our model, a censorship game is a game played between two players. One player, called the *censor*, has comprehensive control over the network of a target area (its SoI), and wishes to prevent certain undesirable communications from being transmitted over that network, while maximizing throughput of legitimate traffic.[1] The other player, called the *circumventor*, wishes to send censored traffic (*e.g.* political speech that the censor disapproves of) over the censor-controlled channel, and may or may not care about the level of throughput for other "legitimate" communications on the censor-controlled network.

The circumventor is able to disguise circumvention, or covert, traffic to match a certain profile of legitimate cover traffic, and exercises control over the amount of traffic that is sent by altering the *base rate* (BR) of the censorship resistance system (CRS) they have deployed. The base rate can be set to any value in the range $0 \leq BR \leq BR_{max}$, where BR_{max} is the maximum amount of traffic that the CRS could transmit if it was fully utilized.

The censor possesses the ability to shut off all traffic (both legitimate and circumvention). The censor may also, but not always, possess the ability to differentiate the circumventor's traffic from the legitimate traffic that it is disguised as, by means of some censorship *apparatus*. This ability to differentiate is prone to errors classified as false positives or false negatives.

Each player has a separate *utility function* that maps from the choice of action taken by both players to the total reward acquired by one of them.

The game is played in a series of discrete rounds, happening in sequential discrete timesteps. At the start of each round, both players simultaneously select an *action*, from their action set, on the basis of the actions selected by the two players in all previous rounds of the game, and on the basis of their own utility functions and calculations.

[1] This is a simplification since the censor may also care about other aspects that contribute to their utility, such as international perception, political fallout, and citizen unhappiness to name a few.

In a censorship game, a *strategy* for the circumventor is a specification of how the base rate parameter will be set at different timesteps in the game, and a strategy for the censor is specification at different timesteps in the game of whether the channel will be left open (allowing all traffic through) or not, and whether or not the apparatus will be used, if it is available. In this setting, we do not model either circumventor or censor expending resources to develop better CRSs or apparatus. For example, a strategy for the censor might be to leave the channel open if the base rate of the circumventor was below a certain level in all previous time steps, and to close it permanently otherwise. An example strategy for the circumventor might be to send no traffic at all for some time, and then send a very large burst of traffic. A *strategy profile* is a specification of a strategy for each player.

A *Nash equilibrium* is a strategy profile where neither player could improve their utility by unilaterally adopting a different strategy. This is a stable point of the game, which we might expect to observe frequently in reality. We will characterize the behaviors of the two agents in terms of the Nash equilibria of the game.

We also assume throughout that both the censor and circumventor have perfect information about each other. That is, both players know what the other *has* done (but not necessarily what they will do next), and knows the exact utility function and parameters being used by the other player.

3.3 A Simple Censor Model

We begin by considering the simplest version of the game where the censor controls only one channel, which carries only one type of traffic. We assume that, absent the traffic of the circumventor, this channel carries a total amount of legitimate traffic L. We normalize both BR and L by setting $L = 1 - BR$.

We now proceed with closed-form analysis of the game in three steps, gradually increasing the complexity of the model.

3.3.1 Step 1: Single Round, No Apparatus

In this version of the game, the two players play just one round of the game, and the censor has no access to an apparatus that would allow it to differentiate between the traffic of the circumventor and the traffic of legitimate users.

The action space of the censor, denoted X_{cen}, consists of two strategies: 1 and 0 (On and Off). Playing "On" means the censor allows all traffic to pass through, unimpeded, while "Off" means all traffic transmission is halted.

The action space of the circumventor is a real number $BR \in [0, 1]$, which is the amount of circumvention traffic the circumventor chooses to send (as a fraction of the total traffic).

The utility functions of the censor and circumventor are respectively given by:

$$U_{cen} = (-\alpha_{act} X_{cen} + \alpha_{bct}(1 - X_{cen}))BR + (\beta_{alt} X_{cen} - \beta_{blt}(1 - X_{cen}))(1 - BR) \tag{3.1}$$

$$U_{cir} = (\gamma_{act} X_{cen} - \gamma_{bct}(1 - X_{cen}))BR + (\delta_{alt} X_{cen} - \delta_{blt}(1 - X_{cen}))(1 - BR) \tag{3.2}$$

Variables $\alpha_{[act,bct]}$, $\beta_{[alt,blt]}$, $\gamma_{[act,bct]}$, and $\delta_{[alt,blt]}$ are parameters that depend on the specific players of the game. The subscripts *act* and *bct* stand for *allow* and *block circumvention traffic*, respectively. The subscripts *alt* and *blt* stand for *allow* and *block legitimate traffic*, respectively. The α_{act} and α_{bct} are the loss, or gain, of utility to the censor of allowing, or blocking, one unit of circumvention traffic, respectively. Similarly, β_{alt} and β_{blt} are the gain, or loss, in utility to the censor of having one unit of legitimate traffic transported via, or blocked on, the channel, respectively. The ratios of α_{act} to β_{alt} and of α_{bct} to β_{blt} characterize different types of censors. For example, an employer interested in reducing employee idleness by preventing communication with social media sites, but ensuring that productive online activities are not affected, might have a relatively low α_{act}, but a relatively high β_{alt}. In contrast, a military agency trying to censor leakage of state secrets might have a very high α_{bct} relative to their β_{blt} parameter. The counterpart parameters γ_{act} and γ_{bct} show the utility gained, or lost, by the circumventor of a single unit of circumvention traffic to be transported, or blocked, respectively. δ_{alt} and δ_{blt} show the utility gained, or lost, of a single unit of legitimate traffic to be transported, or blocked, respectively. All of these parameters can be normalized to the range $[0, 1]$, where 0 means ambivalence and 1 means strong sensitivity.

Conventionally both δ parameters are assumed to be zero since typically CRS designers are not concerned with the fallout of CRS usage nor are there any technical provisions to reduce the impact of the fallout on non-CRS traffic in the designs in the literature. Also, γ_{bct} is also assumed to be zero since typically CRS designs are ambivalent to blocked CRS traffic. Thus the circumventor's utility function is reduced to the following:

$$U_{cir} = \gamma_{act} X_{cen} BR \tag{3.3}$$

Analysis

It is apparent that the censor maximizes its utility by playing "On" if $\beta_{alt}(1 - BR) - \alpha_{act}BR > \alpha_{bct}BR - \beta_{blt}(1-BR)$, and "Off" otherwise.[2] Consequently, the Censor leaves the channel open if it believes the circumventor will play $BR \leq \frac{\beta_{alt}+\beta_{blt}}{\alpha_{act}+\alpha_{bct}+\beta_{alt}+\beta_{blt}}$; or $BR \leq F$ for brevity, where $F = \frac{\beta_{alt}+\beta_{blt}}{\alpha_{act}+\alpha_{bct}+\beta_{alt}+\beta_{blt}}$.

If the players know each others' strategies, the utility of the circumventor is maximized by setting $BR = F$. However, although this is a Pareto Optimal solution, it is actually *not* a Nash equilibrium of the game. This is because the censor and circumventor decide their actions simultaneously, and so do not know each others' actions in advance. Given that the censor plays "On", the circumventor's best response is actually to pick $BR = BR_{max}$, since this maximizes the utility of the circumventor. Consequently, the profile where the censor plays "On" and the circumventor plays $BR = F$ is not a Nash equilibrium.

To find the Nash equilibrium, we note that if the censor plays "Off", the circumventor is equally happy to play $BR = BR_{max}$ instead of any other value of BR (since all settings of BR yields zero utility). This means the circumventor should play $BR = BR_{max}$ regardless of what the censor does, simplifying the game considerably. Knowing that the circumventor's utility is maximized by playing BR_{max} regardless, the censor would choose to play "On" if and only if $BR_{max} < F$. In a game where this holds true, the Nash equilibrium is for the censor to leave the channel open, and the circumventor to play BR_{max}. Otherwise, the Nash equilibrium is for the censor to close the channel and for the circumventor to play BR_{max}.

Thus, we can see that, in this simplified game, the Nash equilibrium depends on both the maximum amount of traffic the circumventor can send, and on the tradeoff between the costs and benefits to the censor of allowing and blocking circumvention traffic versus keeping legitimate traffic flowing.

However, in nature, we rarely observe the equilibrium where censors elect to close their channels entirely. In the next section, we show that a circumventor interested in maintaining communications over a longer, uncertain time horizon, will behave differently, leading to a different equilibrium from the one observed here.

3.3.2 Step 2: Multiple Rounds, No Apparatus

As in the Prisoner's Dilemma, the Nash equilibrium in the simple censorship game described above results from a failure to model the temporal dynamics of the game. Intu-

[2]Note that the analysis is invariant under affine transformations of the players' utility functions.

itively, if both censor and circumventor know that exactly one round of the game will be played, there is no reason for the circumventor to hold back: they will always send the largest possible amount of traffic, and if the censor doesn't block, the circumventor gets as much reward as possible. If the censor does block, then the circumventor would not get any reward regardless of what they played. In the face of such an opponent, the censor of course must block, to avoid the unacceptable volume of illegitimate traffic that would be sent.

The key result for cooperation in temporal games, due to Aumann [Aum59], is that defection follows if the players *know* when the game will end. This is because, in the last round of the game, the players are simply playing the static game again (there is no temporal component, because the game will now end, just like in Case 1 above). Once the players know how the final round will be played, then they can also infer how the second last round should be played using exactly the same logic, by treating the game as ending one round earlier than before. Inductively, the players will play the first round in the same fashion as they would the last. However, when the game is played for an infinite or indefinite number of rounds, then this need not be so.

Suppose that after each round of the game, another round is played with probability p, and otherwise the players stop. This can model scenarios where the CRS or communication technology has become deprecated, or because the conditions of censorship have changed. A strategy in the context of this "supergame" (*i.e.* the game of playing many rounds of the censorship game described in Case 1) consists of specifying a policy for how a player plays, in light of everything their opponent has done in the past.

We analyze this game using the same utility function from case 1 since it is still applicable. Again we assume that the δ and γ_{bct} parameters are zero due to typical CRS designs not being concerned with the fallout of CRS activity and discount the blocked CRS traffic.

Analysis

An interesting Nash equilibrium now emerges (though not necessarily a unique one). The censor adopts a policy to play "On" as long as the circumventor has never played $BR > F$ at any point in the past, and to play "Off" if even one prior iteration of the game involved the circumventor sending more traffic than that. The circumventor adopts a policy of playing $BR \leq F$ at every step.

To show that the censor leaving the channel open and the circumventor playing $BR = F$ is a Nash equilibrium, we use proof by induction.[3]

[3]This proof assumes that $BR \ll 1$, which is supported by empirical evidence from statistics we collected on the Tor network that appear in Section 4.7.

In the first round, the circumventor could deviate and send up to $BR = BR_{max}$ traffic. However, doing so would result in a total utility of $\gamma_{act} BR_{max}$ for this turn, and zero utility thereafter. In contrast, using $BR = F$ this turn, and defecting next turn instead, would result in an expected total utility of $\gamma_{act}(F + pBR_{max})$. Provided that $BR_{max} < F + pBR_{max}$, it is thus better to wait another turn before sending more traffic than F. It follows that deviation for the circumventor will always be better in "one more turn", if $BR_{max} < \frac{F}{1-p}$.

Unfortunately, there are other equilibria in this game. Notably, if $BR_{max} > F$, then the policy where the censor always blocks, and the circumventor always sends BR_{max} is a Nash equilibrium as well.

Interestingly, we note that p could be replaced by any discounting factor for the utility of future rewards. So if, instead of representing the chance of a future game, p represented the preference of each party for rewards today as opposed to in the future, a similar result could be derived. In practice, most companies do use such a discounting factor when considering the benefits of future rewards, since events in the future are fundamentally uncertain. To provide a censorship resistance example: a whistleblower may use a discounting factor where they are uncertain about their ability to communicate in the future and the value of the information they wish to transmit may be of such high impact that maintaining the channel for future use may be ignored.

We can conclude from this analysis that it is the best policy for the circumventor interested in maintaining a long-term communication channel to keep $BR \leq F$.

3.3.3 Step 3: Multiple Rounds, With an Apparatus

We now consider the case where the censor has some apparatus capable of distinguishing the target, covert, traffic (BR) from the cover traffic (L). The apparatus correctly labels a fraction TPR (the true positives) of the circumvention traffic, but also incorrectly labels a fraction FPR (the false positives) of the legitimate traffic as circumvention traffic. Similarly, traffic not positively labeled can be partitioned to that which is truly not circumvention traffic, *i.e.* TNR (true negatives), and that which has been missed by the apparatus, *i.e.* FNR (false negatives). We note that $FNR = 1 - TPR$ and $TNR = 1 - FPR$. The output of the apparatus is traffic with the "Positive" tag or "Negative" tag, referring to if the apparatus deems the traffic as being CRS-related or not, respectively.

The new action space of the censor has two variables, denoted X_p and X_n, where both can take the values 0 and 1 (Block and Allow). X_p governs traffic tagged "Postive" and the censor can either block or allow this traffic. Similarly, X_n governs traffic tagged

"Negative" and the censor can again either block or allow the traffic. The action space of the circumventor remains unchanged from before.

The presence of the apparatus serves to alter the utility functions of the players as follows:

$$\begin{aligned} U'_{cen} =& BR'(-\alpha_{act}(TPR \cdot X_p + FNR \cdot X_n) + \alpha_{bct}(TPR(1 - X_p) + FNR(1 - X_n))) + \\ & (1 - BR')(\beta_{alt}(FPR \cdot X_p + TNR \cdot X_n) - \beta_{blt}(FPR(1 - X_p) + TNR(1 - X_n))) \end{aligned} \tag{3.4}$$

$$U'_{cir} = BR'(\gamma_{act}(TPR \cdot X_p + FNR \cdot X_n)) \tag{3.5}$$

The parameters are all normalized as before to the range $[0, 1]$.

To help build intuition, as an example let us consider the censor's sensitivity to blocking circumvention traffic (α_{bct}). Its contribution to the censor's utility function is $BR' \cdot \alpha_{bct}(TPR(1 - X_p) + FNR(1 - X_n))$ because a fraction BR' of the traffic *is* circumvention traffic, and of that, TPR of it is reported as positive, which will get blocked if $X_p = 0$, and $FNR = 1 - TPR$ of it is reported as negative, which will get blocked if $X_n = 0$. Similar reasoning follows for the other parameters.

Analysis

Ultimately the dynamics of this game are similar to those in case 1 or 2 (depending on whether we incorporate temporal dynamics or not), with adjustments to the parameters of the censor. First, we analyze the censor's strategy space and make the following observations.

The censor has four strategies to play. Strategy $(X_p, X_n) = (1, 1)$ is the same as not having an apparatus since the censor ignores the "Positive" tag on traffic and allows it through as well as allowing all the traffic with the "Negative" tag.

Strategy $(X_p, X_n) = (0, 0)$ is again the same as not having an apparatus and is also the same as blocking all traffic since the censor disagrees with traffic tagged "Negative" and blocks it as well as blocking all the traffic tagged "Positive".

Strategy $(X_p, X_n) = (0, 1)$ is where the censor goes along with the tagging of the apparatus and blocks traffic labeled "Positive" and allows traffic labeled "Negative".

Strategy $(X_p, X_n) = (1, 0)$ implies that it is always better for the censor to disagree with the apparatus completely and do the opposite of what its tagging suggests. So now, traffic labeled "Positive" is allowed through while traffic labeled "Negative" is blocked. For

the sake of simplicity, we assume that should the censor find that disagreement is more beneficial then it simply switches the tags which makes this strategy equivalent to strategy $(0,1)$ above. This is the same as assuming that $TPR \geq FPR$ and, equivalently, that $TNR \geq FNR$.

We now consider these strategies in more detail. Setting $(X_p, X_n) = (1,1)$ in Equation 3.4 gives the following:

$$U'_{cen^{(1,1)}} = BR'(-\alpha_{act}) + (1 - BR')(\beta_{alt}) \tag{3.6}$$

Similarly, the other settings yield the following utility equations:

$$U'_{cen^{(0,0)}} = BR'(\alpha_{bct}) + (1 - BR')(-\beta_{blt}) \tag{3.7}$$

$$\begin{aligned} U'_{cen^{(0,1)}} = & BR'(-\alpha_{act} \cdot FNR + \alpha_{bct} \cdot TPR) + \\ & (1 - BR')(\beta_{alt} \cdot TNR - \beta_{blt} \cdot FPR) \end{aligned} \tag{3.8}$$

To discover when it is better to play each strategy we compare each one against the other. Since the censor's utility depends on the circumvention traffic we state the results of this comparison in terms of BR'.

For the censor to chose $(1,1)$ over $(0,0)$ then $U'_{cen^{(1,1)}} \geq U'_{cen^{(0,0)}}$ and the following must hold:

$$BR' \leq \frac{\beta_{alt} + \beta_{blt}}{\alpha_{act} + \alpha_{bct} + \beta_{alt} + \beta_{blt}}, \tag{3.9}$$

or $BR' \leq F_{ab}$, where $F_{ab} = \frac{\beta_{alt}+\beta_{blt}}{\alpha_{act}+\alpha_{bct}+\beta_{alt}+\beta_{blt}}$. The subscript ab denotes that when the inequality holds the censor gets more utility by allowing all traffic through than by blocking it. Note that $F \equiv F_{ab}$.

For the censor to chose $(1,1)$ over $(0,1)$ then $U'_{cen^{(1,1)}} \geq U'_{cen^{(0,1)}}$ and the following must also hold:

$$BR' \leq \frac{FPR(\beta_{alt} + \beta_{blt})}{TPR(\alpha_{act} + \alpha_{bct}) + FPR(\beta_{alt} + \beta_{blt})}, \tag{3.10}$$

or $BR' \leq F_{am}$, where $F_{am} = \frac{FPR(\beta_{alt}+\beta_{blt})}{TPR(\alpha_{act}+\alpha_{bct})+FPR(\beta_{alt}+\beta_{blt})}$. Similar to the convention used above, the subscript am denotes that when the inequality holds the censor gets more utility by allowing all traffic than by using the apparatus (the m stands for machine, since the apparatus is a kind of machine).

For the censor to chose $(0,1)$ over $(0,0)$ means that $U'_{cen^{(0,1)}} > U'_{cen^{(0,0)}}$. Therefore the following must also hold:

$$BR' \leq \frac{TNR(\beta_{alt} + \beta_{blt})}{FNR(\alpha_{act} + \alpha_{bct}) + TNR(\beta_{alt} + \beta_{blt})}, \tag{3.11}$$

or $BR' \leq F_{mb}$, where $F_{mb} = \frac{TNR(\beta_{alt}+\beta_{blt})}{FNR(\alpha_{act}+\alpha_{bct})+TNR(\beta_{alt}+\beta_{blt})}$. Again similar to before, the subscript mb denotes that when the inequality holds the censor gets more utility by using the apparatus than by blocking all traffic.

Each of F_{ab}, F_{am}, and F_{mb} is a threshold on BR' that drives the censor's decision to allow, block, or use the apparatus. We would like to discover the ordering between the thresholds so that the censor can make informed (strategic) choices. We make an observation that simplifies the analysis: the terms $\alpha_{act} + \alpha_{bct}$ and $\beta_{alt} + \beta_{blt}$ are common and can be replaced with α and β, respectively. When determining the relative ordering of the three thresholds, we will assume, as above, that $TPR \geq FPR$ (and equivalently, that $TNR \geq FNR$).

We begin by noting that $F_{ab} \geq F_{am} \Leftrightarrow FPR \leq TPR$ since:

$$\begin{aligned} & F_{ab} \geq F_{am} \\ \Leftrightarrow \quad & \frac{\beta}{\alpha+\beta} \geq \frac{FPR \cdot \beta}{TPR \cdot \alpha + FPR \cdot \beta} \\ \Leftrightarrow \quad & \frac{\alpha+\beta}{\beta} \leq \frac{TPR \cdot \alpha + FPR \cdot \beta}{FPR \cdot \beta} \\ \Leftrightarrow \quad & \frac{\alpha}{\beta} \leq \frac{TPR \cdot \alpha}{FPR \cdot \beta} \\ \Leftrightarrow \quad & FPR \leq TPR \end{aligned} \tag{3.12}$$

Similarly, we also note that $F_{mb} \geq F_{ab} \Leftrightarrow FNR \leq TNR$ since:

$$\begin{aligned} & F_{mb} \geq F_{ab} \\ \Leftrightarrow \quad & \frac{TNR \cdot \beta}{FNR \cdot \alpha + TNR \cdot \beta} \geq \frac{\beta}{\alpha+\beta} \\ \Leftrightarrow \quad & \frac{FNR \cdot \alpha + TNR \cdot \beta}{TNR \cdot \beta} \leq \frac{\alpha+\beta}{\beta} \\ \Leftrightarrow \quad & \frac{FNR \cdot \alpha}{TNR \cdot \beta} \leq \frac{\alpha}{\beta} \\ \Leftrightarrow \quad & FNR \leq TNR \end{aligned} \tag{3.13}$$

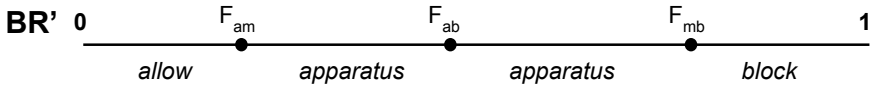


Figure 3.1: Best censor strategies at critical circumvention traffic thresholds. The censor's strategies are in *italics*. The circumventor's strategies are to send a proportion of circumvention traffic, $0 \leq BR' \leq 1$, with the critical thresholds marked as F_{am}, F_{ab}, and F_{mb}.

Since $F_{mb} \geq F_{ab}$ and $F_{ab} \geq F_{am}$, it is clear that the total ordering is $F_{mb} \geq F_{ab} \geq F_{am}$.

Given this ordering, the censor will play according to the following strategies, which are depicted in Figure 3.1. When $BR' \leq F_{am}$ the censor will allow all traffic to flow. When $F_{am} \leq BR' \leq F_{ab}$ or $F_{ab} \leq BR' \leq F_{mb}$ then the censor will use the apparatus rather than allowing or blocking all the traffic, respectively. Finally, when $BR' > F_{mb}$ the censor should block all traffic.

Turning to the circumventor we see that she actually only has two reasonable choices: sending $BR' = F_{am}$ (in which case all of her circumvention traffic will get through), or $BR' = F_{mb}$ (in which case only a fraction FNR of her circumvention traffic will get through). The decision rests on whether $FNR \cdot F_{mb} \geq F_{am}$; *i.e.*, when the inequality holds, the circumventor should send $BR' = F_{mb}$ circumvention traffic, and otherwise she should send $BR' = F_{am}$.

The key takeaway from the analysis in this section is that neither party has an incentive to deviate from the equilibrium points, as defined by the circumvention traffic thresholds F_{am}, F_{ab}, and F_{mb}. That is to say that as long as the circumventor does not send more than F_{mb} traffic, the censor will not block it, but will apply its apparatus to reduce the amount of circumvention traffic that gets through, or allow it entirely if it is below F_{am}.

It is clear then that the introduction of the apparatus, with its inherent TPR and FPR, does not produce a deviation from the character of the Nash equilibrium that we found in the simpler cases 1 and 2. The main effect is on the amount of traffic, BR', the circumventor can send through while ensuring that the inequalities above remain true.

3.4 More Realistic Censor Models

So far we have analyzed censor utility functions that are linear in nature. In reality, the censor may be more risk averse. We mean by this that the censor's stakes (costs) to

blocking CRS traffic, and not making mistakes, ramp up faster as rates of errors increase than the linear model above. One way to capture this is to utilize an exponential utility function for the censor.

The following is an example of an exponential censor utility function.

$$U''_{cen} = e^{-(C \cdot FPR \cdot (1-BR) + D \cdot FNR \cdot BR))} \tag{3.14}$$

$$U''_{cir} = E \cdot FNR \cdot BR \tag{3.15}$$

Similar to the earlier α and β, the non-negative parameters C and D control the sensitivity of the censor to false positives and false negatives respectively. Like γ before, the non-negative parameter E controls the circumventor's sensitivity to circumvention traffic getting through the censor's SoI; $E = 1$ for the remainder of this discussion. As before, the variable FNR is the percentage of the circumvention traffic allowed (*i.e.* the false negatives) and FPR is the percentage of legitimate traffic blocked (*i.e.* the false positives). This function allows a wide range of plausible censor utility functions to be modeled, and results in utility values between 0 (maximum dissatisfaction) and 1 (maximum satisfaction).

A second simplification we have thus far made was to only consider a single protocol that the CRS could blend in with. In reality there are a plethora of protocols that a CRS could use for cover, *e.g.* HTTP, TLS, and VoIP to name a few. Furthermore, it is likely that some protocols are more critical, or at least more important, than others and interfering with them would cost the censor more dearly.

Unfortunately, when we take these factors into consideration the preceding closed-form style of analysis becomes more complex and less straightforward to reason about. We change tacks here and leverage numerical simulation to help us analyze and gain further insights. We exploit our finding from the closed-form analysis above that a protocol remains unblocked as long as the circumventor does not transmit more than a certain amount of traffic over it. We create a simulation that utilizes Equation 3.14 and Equation 3.15 above and iterates over parameter values to help us find potential Nash equilibria for various types of censors.

The aim of the analysis that follows is to explore how to identify cover protocols that are good candidates as cover traffic for the amount of circumvention traffic that we wish to send. We focus on the quantity of the cover traffic a protocol provides rather than its other qualities such as its importance or the ease with which it can be imitated.

3.4.1 Strategy Simulator

Our simulator models the censorship game as follows. The circumventor moves first, and produces a CRS which impersonates one or more protocols and distributes circumvention traffic over these protocols according to some distribution. The censor can masquerade as a CRS client, and is able to establish which protocols are being impersonated and how much circumvention traffic is being sent over each. We examine the case where the impersonation is good—the censor does not have an apparatus that can distinguish legitimate uses of the protocol from uses of the protocol to carry circumvention traffic. Therefore, the censor must choose to either block a protocol entirely—blocking both cover traffic (causing false positives) and the circumventor's traffic (causing true positives), or leaving it entirely unblocked.

We let the censor move second because it is likely that the censor can move faster than the circumventor—the circumventor must roll out new software to many users in order to change strategy whereas the censor needs only to make a configuration change. In each round the censor will choose a blocking strategy, *i.e.* which protocols they will block, to maximize their utility. The goal of the circumventor is to find the right proportion of the total amount of circumvention traffic to send over each protocol such that the censor's best strategy is the one the circumventor finds gives the most utility. This will be the equilibrium strategy since if either party changes their choice, they will decrease their own utility.

An interesting consequence of this model is that the utility function of the circumventor does not matter, as all they can do is choose between the collection of scenarios which the censor has decided to be optimum for a particular strategy of the circumventor. Therefore, as long as the circumventor's utility function is monotonically increasing in terms of the false negative rate, the same equilibrium will be reached regardless of the function's shape.

The simulator models relative importance of protocols, for both the censor and the population in the censor's SoI, by utilizing popularity of the protocol by traffic volume. As a concrete source of information we use traffic-volume data supplied by the 2014 survey of US Internet traffic [San14]. Our simulator makes some simplifying assumptions to reduce the computational complexity of the simulation. We will provide more detail in Section 3.4.1 where this becomes relevant.

False-Positive Intolerant Censor

We first consider a censor with low tolerance to false positives. We define this to mean that there is at least one protocol which they are unwilling to block (a *critical* protocol), even

if blocking the protocol would result in blocking all of the circumventor's traffic. In this case the circumventor should choose the critical protocol and send all censorship-resistance traffic over it. The censor will not block it, and so all of the circumvention traffic will get through. Any alternative strategy for the circumventor would be less good, as choosing multiple critical protocols would be more effort for no gain, and choosing a non-critical protocol for some traffic might lead the censor to block it.

False-Positive Tolerant Censor: Variant 1

A more interesting case is where there is no such critical protocol. To give a concrete example, assume that the circumventor can impersonate six protocols with the same relative quantities of traffic as the top six types of traffic from the survey: Netflix streaming video (33.81%), YouTube streaming video (14.63%), HTTP (6.08%), BitTorrent (4.85%), iTunes (3.12%) and Facebook (2.60%).[4] We shall call the most prevalent protocol the top protocol, and the least prevalent the bottom protocol, with the rest forming an ordering in between.

As the censor utility function, we use Equation 3.14 with $C = 0.3$ and $D = 0.25$. This is illustrated on Figure 3.2 for three values of true positive rates: 100% (top), 50% (middle) and 0% (bottom).

We now need to compute the censor utility function for all combinations of censor strategy and circumventor strategy. The censor can choose to block any selection of protocols of the six considered (there is no reason to block any others). As a result there are $2^6 = 64$ scenarios.

The circumventor can choose to send units of traffic in any distribution over the protocols, but we exclude any distribution where the traffic distributed over protocol a is greater than that distributed over protocol b when the quantity of cover traffic going over protocol b is greater than that of a. We do this because if any excluded scenario were chosen, if a and b were swapped, the censor utility function would be lower for every censor scenario (assuming the censor prefers a lower false-positive rate).

Even making this assumption there are still an infinite number of circumventor scenarios if we allow any fractional value for the amount of traffic. So, to reduce the scenario space we quantize all circumvention traffic into multiples of 5 units up to a total of 100 units, resulting in 282 circumventor scenarios. The result of simulating all scenarios is shown

[4]Note that these percentages do not add up to 100% since there will remain traffic types that are not targetted by the CRS and thus the situation where the censor needs to block all Internet access will not arise.

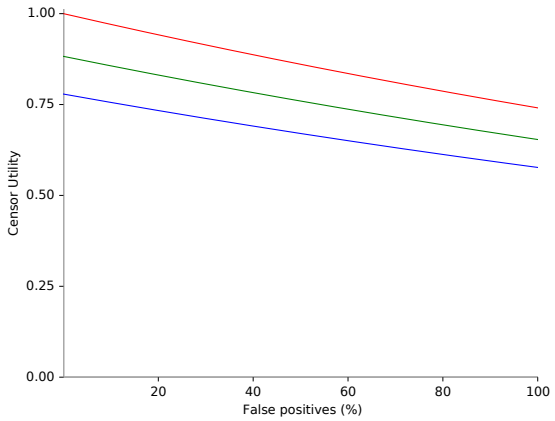


Figure 3.2: Utility of a censor with high false-positive and false-negative tolerance.

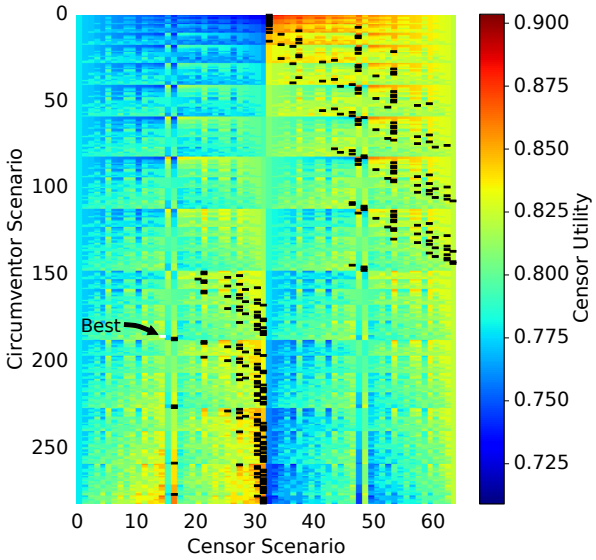


Figure 3.3: Utility of a censor with high false-positive and false-negative tolerance.

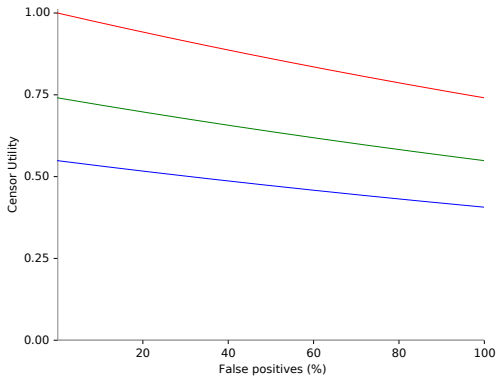


Figure 3.4: Utility of a censor with high false-positive and low false-negative tolerance.

in Figure 3.3, where blue is low utility and red is high utility. The censor scenarios are sorted in order of increasing false-positive rate. The circumventor scenarios at the top have traffic heavily skewed to the protocols with the most cover traffic; those at the bottom have traffic more evenly distributed over the protocols. The small rectangles show the optimum censor strategy for each circumventor strategy (white with the arrow labeled "Best" for the equilibrium and black for others).

Even small changes in the circumventor scenarios result in large changes in optimum censor scenario, but the equilibrium for this censor type is for the circumventor to distribute circumvention traffic quite evenly over the protocols, but not completely. The top protocol should get 40 units of traffic and the next four with 15 units of the traffic each with the sixth not used at all. The censor will block protocols 3, 4, and 5, allowing 55 units of circumvention traffic through. Were the attacker to block protocols 1 and 2, the additional false positives would not justify the extra 55 units of true positive (circumvention) traffic. Were the circumventor to move some traffic onto protocol 6, it would be blocked because it has a smaller false-positive cost.

False-Positive Tolerant Censor: Variant 2

Let us now consider a censor who is equally tolerant to false positives, but far more sensitive to false negatives than before, by changing D from 0.25 to 0.6 with the result shown in Figure 3.4. Now a 50% false negative rate shows significantly lower censor utility than variant 1 (the middle green line). The resulting simulation is significantly different as well, as can be seen in Figure 3.5.

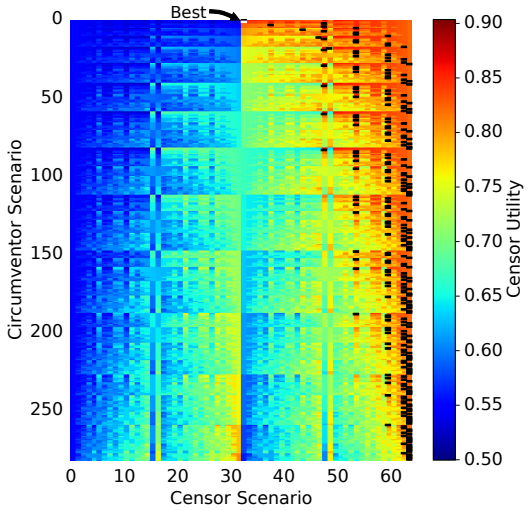


Figure 3.5: Utility of a censor with high false-positive and low false-negative tolerance.

Now the optimum strategy for the censor is almost always to block many protocols, resulting in a high false-positive rate (the right-hand side of the graph). The equilibrium strategy is for 95 units of circumvention traffic to be distributed on protocol 1 and 5 units to be distributed on protocol 2. The censor will block protocol 1, but leave protocol 2 unblocked. This lets only 5 units of circumventor traffic through, but it is better than none, which almost every other strategy results in. For example, sending 100 units of traffic over protocol 1 results in it being blocked. Sending 80 units over protocol 1 and 20 units over protocol 2 results in both protocols being blocked. Putting only 5 units over protocol 2 is small enough that the extra benefit to the censor of blocking it is not large enough to justify the high false positives.

3.4.2 Parameter Analysis

The analysis above provides some insight into how different censor types behave and the optimum strategy for distributing traffic given the traffic volumes of potential cover protocols from real-world data. We now analyze what occurs when the number of protocols is varied as well as the amount of cover traffic they provide.

Protocol Popularity

The popularity, or amount of cover traffic available, of a protocol plays a significant role in the resulting Nash equilibrium and hence censor and circumventor strategies. We investigate this by taking a hypothetical protocol and varying its popularity, *i.e.* units of cover traffic, relative to all other non-cover traffic on the censor's network. Note that since there is only one protocol the circumventor can only play one action, send all 100 units of circumvention traffic over the protocol.[5] We use this setup to re-evaluate the fault-tolerant censors from above.

We see that for the censor with $C = 0.3$ and $D = 0.25$ the censor does not change their blocking pattern until the cover protocol gets to be a little more than 83 units of the total bandwidth. After this inflection point the censor switches to allowing the 100 units of circumvention traffic through since the collateral damage outweighs the benefit of information blocking. The takeaway is that, in this scenario, if we could only target one protocol it had better provide at least 83 units of cover traffic for each 100 units of circumvention traffic, or we would not be able to use it as cover to safely send all the circumvention traffic past the censor.

We can verify this simple case with one protocol using Equation 3.1 and rewriting it with α_{blt} and β_{act} replaced with C and D—the remaining parameters set to zero—to produce $BR'' \leq \frac{C}{C+D}$ which yields $BR'' \leq \frac{0.3}{03+0.25} \approx 0.545$. The censor will allow circumvention traffic to flow (100 units of it since there is only one channel) if it is 54.5% of the *total* traffic, *i.e.* the sum of the circumvention traffic and the cover traffic. This means that the cover protocol must be 45.5% of the total traffic, corresponding to the $\frac{83}{83+100}$ suggested by the simulation.

However, for the censor with more sensitivity to information leakage, *i.e.* $C = 0.3$ and $D = 0.6$, the inflection point occurs at a much larger 203 units of cover traffic, which closed-form analysis also confirms. This means that for this censor to allow 100 units of circumvention traffic a very popular protocol needs to be used as cover. Table 3.1 and Table 3.2 illustrate these trends.

While it seemed like it is better to target a protocol that is the majority of bandwidth on the network in general, the above examples show that there are censors for whom this approach can not be employed since their sensitivity to information leakage, D, is too high as compared to their sensitivity to collateral damage, C.

[5] We do not model the situation where the circumventor can hold back sending all the traffic they wish to send. We do this to simplify the analysis and also to illustrate the difference in the results where the cover protocol is not popular and where it is.

Table 3.1: Cover protocol bandwidth effects on utility, $C = 0.3$, $D = 0.25$

Bandwidth	U_{cir}	U_{cen}
10	0	0.97
50	0	0.86
83.5	1	0.78
90	1	0.78
99	1	0.78

Table 3.2: Cover protocol bandwidth effects on utility, $C = 0.3$, $D = 0.6$

Bandwidth	U_{cir}	U_{cen}
10	0	0.97
50	0	0.86
100	0	0.74
201	1	0.55
210	1	0.55

Dynamics of Cover Bandwidth over Two Transports

The number of cover protocols can play a role in how the censor behaves. We investigate this by utilizing two hypothetical cover protocols where the sum of their cover traffic is kept constant. We then vary the amount of cover traffic units between the two to investigate the effects on the censor's best responses. We choose just below the inflection point from the analysis above as the total cover traffic units to distribute between the two protocols, *i.e.* 83 units of cover traffic. We do this to see if there is any difference in the censor's behavior.

We see from Table 3.3 that leveraging two cover protocols, where one is very small compared to the other, against the first censor ($C = 0.3$, $D = 0.25$) causes reduced utility. As the cover traffic ratio between the two protocols decreases we see that the circumventor loses more utility which implies that in this scenario it is more beneficial to leverage a single cover protocol than multiple protocols.

Against the second censor ($C = 0.3$, $D = 0.6$), for whom we saw that only a very large amount of cover traffic (203 units) could cause it to deviate from its block-everything strategy, we see from Table 3.4 that now targeting two protocols instead (with a much smaller sum of 83 units of cover traffic) can cause the censor to allow 100 units of circumvention traffic to flow over them.

Table 3.3: The effect of cover traffic distributed over two protocols on utility, $C = 0.3$, $D = 0.25$

Bandwidths	U_{cir}	U_{cen}
82,1	0.95	0.79
72,11	0.85	0.78
62,21	0.70	0.79
52,31	0.60	0.78
42,41	0.50	0.78

Table 3.4: The effect of cover traffic distributed over two protocols on utility, $C = 0.3$, $D = 0.6$

Bandwidths	U_{cir}	U_{cen}
82,1	0	0.78
72,10	0.5	0.78
62,20	0.10	0.78
52,30	0.15	0.78
42,41	0.20	0.78

It is interesting that the two censors produced such different results; in one case targeting two protocols (with their sum equal to the noted inflection point) produced a reduced utility for the circumventor, while in another it allowed some portion of traffic to flow where none was allowed before even though the amount of cover traffic did not increase. It shows us that choice of not only which protocols (*i.e.* the amount of cover traffic they offer) but also the ratio of cover traffic between them can have an impact on censor behavior, such that it is beneficial to CRS activity. It is an avenue for future work to explore these aspects more to understand and ultimately leverage the censor's sensitivity to particular protocols and their combinations.

3.5 Closing the Loop

Our censorship games depended on perfect information and this makes it necessary to discover the correct type for the censor and the values of the parameters. However, this may be difficult, if not impossible, since the censor's preferences by their nature are unobservable and the censor does not cooperate and hides the information. Hence, our discussions have been concerned with a parameterized censor to allow us to explore various dimensions of censorship. This parameterized "open-loop" analysis allows us to gain insight, but in

order to move towards applicability to real-world scenarios we must reconsider the role that observations can play. Indeed, we *can* still make observations about the censor's behaviors, which are dictated by these hidden parameters. Indeed, we can now replace our need for actual parameter values with observations of the censor. This insight allows us to conduct a "closed-loop" analysis where we can potentially predict real-world behavior and CRS outcomes.

Our approach is that, instead of working with utility for specific parameter values, we gather up utility functions into equivalence classes of observed censor actions. Furthermore, we only consider equivalence classes (*i.e.* censor behavior), and hence parameter values, that directly impact the circumventor's utility function. This has the added benefit of the reduction in complexity in terms of equivalence class space and makes the problem more tractable and enables us to find effective strategies for designing and deploying CRSs.

We do not completely, and accurately, attempt to map all parameters for all censors, CRSs and users, but the framework presented here can help in refining censor behavior models and be a jumping off point for future work.

3.5.1 Methodology

We first create a repository of censor *equivalence classes* based on observations of censor behavior. These are collections of censor actions, or action profiles, that characterize its behavior in the dimensions that the CRS is affected by. The profiles have a few conditions; they are distinct from one another and the actions in the profile need to be observable and maximize the censor's utility. In the setting we have presented, the action profile is the blocking pattern that the censor adopts. Each of the patterns is distinct and is easily observable, *e.g.* by probing which protocols are blocked and at what level of traffic.

We then consider past (passively) observed censor behavior and the conditions (or inputs) that cause it and map them to the equivalence classes. Where past observations are not available, an active probing test suite can collect the needed data. Indeed, there are repositories of past observed censorship events that we can mine for data that we require. Projects like OONI [FA12] track worldwide censorship events while the Tor project tracks country-level blocking incidents [Lew09, Lew12]. Both of these projects can provide valuable insight about the censor's behavior under conditions similar to the ones we are interested in. The active probing test suites are generally geared towards the "how" of censorship rather than the conditions that cause it, such as the rate of circumvention traffic, which is what we are interested in. An added wrinkle is that the probing may itself cause the censor to react and change its behavior and so must be carefully evaluated

to not contaminate the testing environment. Nevertheless, with further enhancements and judicious deployment we expect that active probing can yield a fruitful source of information to refine our equivalence class models for particular censors.

By these methods we would converge at 1) those equivalence classes that matter for the CRS, 2) the region the equivalence classes occupies in the parameter space, and 3) the boundaries between classes that transition a censor from one profile to another.

In this manner we could predict a particular censor's behavior for given inputs and hence can design CRSs that allow us to maximize the circumventor's utility. What follows is an application of our methodology on the parameterized censor from our discussions so far. A similar application would follow for a particular real-world censor and CRS using the historical and active data collection schemes we mentioned earlier.

3.5.2 Censor Equivalence Classes

We apply this methodology to our censor utility function, in Equation 3.14. First, we enumerate the blocking patterns that we expect to appear due to U''_{cen} for the scenario presented in Section 3.4.1 with the false-positive tolerant censor with six protocols. We compile a heatmap of best responses by censors of varying sensitivity values, C and D, in the range $[0, 1]$. The results are presented as the heatmap in Figure 3.6.

The interesting thing to note is that within this range out of the 64 possible blocking patterns the censor's best responses are limited to just 11, meaning that those are the patterns that the circumventor actually needs to address. From this map of the blocking patterns we can probe the censor's type to converge on the equivalence class of a particular censor by sending different proportions of circumvention traffic over the protocols and noting the behavior of the censor.

Since we are only interested in censor patterns that provide positive utility to the circumventor we also produce a circumventor utility heatmap (Figure 3.7) to compare with the blocking pattern heatmap. There are 15 contiguous regions with the same circumventor utility. These follow the same general trends of the blocking patterns but with some censor equivalence classes providing two different utilities for certain ranges of values. We note that there is a contiguous region (the black region in the top half) that provides no circumventor utility, and this corresponds with the pattern to block the top protocol—where the circumventor also sends all circumvention traffic over the top protocol—and the pattern to block all protocols. The bottom light shaded region provides the most utility (*i.e.* all circumvention traffic is allowed through) and this corresponds to the block-nothing pattern.

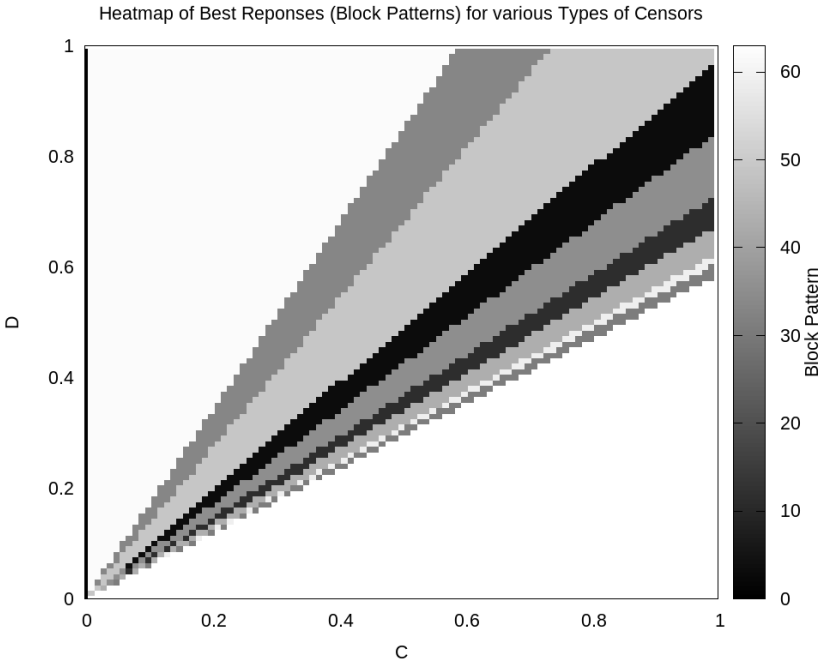


Figure 3.6: Best censor responses (blocking patterns) for various censor types, *i.e.* values of C and D. Each shade represent one blocking pattern and all regions with the same shade represent a single censor equivalence class. The lighter shades denotes blocking patterns where fewer protocols are blocked and darker shades denotes patterns where more protocols are blocked.

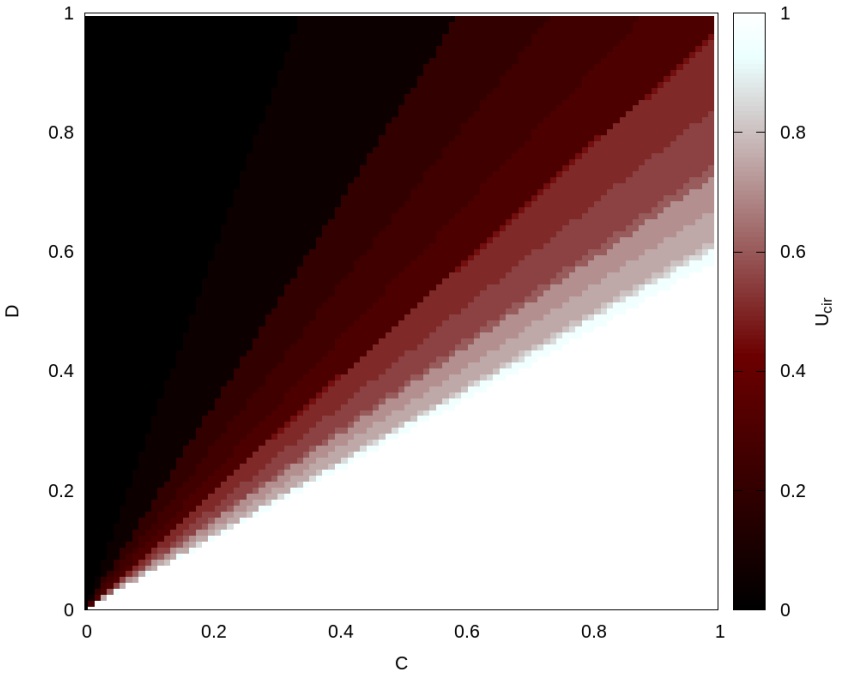


Figure 3.7: Circumventor utility for best responses for various censor types, *i.e.* values of C and D. Each shade represent 0.05 units of circumventor utility. The lighter shades denote high utility and darker shades denote low utility, with black denoting zero utility (*i.e.* no circumventor traffic allowed through).

This framework allows us to discover the overall shape of the game. Given the traffic proportions of the cover protocols that the CRS can target we can use the methodology above to discover which censor strategies are likely to come into play and the potential circumventor utility we can achieve. This can allow the CRS designer to decide if it is worth playing the game and to help them target the right set of cover protocols that allow positive circumventor utility.

3.6 Related Work

Microeconomic approaches of incentive analysis and game-theoretical models have been adopted in numerous applications of network security for preventing attacks and designing adversarial intrusion detection models. In surveys [AB10, RES+10, MZA+13] of the evolution of computer networks and security systems we see a drastic change from the use

of heuristic and ad hoc solutions, to analytical paradigms that are based on rich game-theoretic models. This new shift has enabled researchers to account for players' incentives and attitudes towards decision making in various environments.

In the context of censorship resistance systems that are mainly inspired by peer-to-peer file/media sharing frameworks, researchers have focused on two orthogonal approaches: randomized file and functionality sharing where each node is assigned random resources, and a discretionary model where peers can choose and modify their precise contributions to the network [AMNO07, AM06]. Danezis and Anderson [DA04] studied these two frameworks and showed that, in contrast to the initial intuition, the random model is less costly to attack for all possible attacker strategies, and that the cost to censor a set of nodes is maximized when resources are distributed according to node preferences. Contemporaneous to the work in this chapter, Tschantz *et al.* [TAPT14] promote the idea that evaluating censorship resistance designs solely on technical attributes is shallow and at times intractable and present game-theoretic analysis as an alternative. The analysis and contributions are limited to considering abstract cost functions and preliminary conclusions about the viability of economic analysis as a means of evaluating CRS designs.

To the best of our knowledge, our work is the first to offer a framework for game-theoretic analysis of censorship resistance on the data channel in a variety of scenarios.

3.7 Conclusion

In this chapter, we focus attention on the censorship games wherein two rational and self-interested players, namely censor and circumventor, play their best strategic responses in a perfect information game. Considering a linear utility model, we start by analyzing the simplest pure Nash equilibrium analysis and enrich the model step by step. We then analyze the exponential utility setting and describe a simulated approach to equilibrium analysis.

Our simple closed-form analysis yields insight about the existence of Nash equilibriums that can be leveraged by CRS designs. Extending our analysis to more realistic censorship scenarios, we leveraged simulation as an aid to discovering and analyzing equilibrium points. This approach has application to real-world CRS-design problems, namely, of how to select useful cover protocols and how to distribute circumvention traffic over them. Finally, we provide intuition about how one might go about discovering the censor's type using active probing as a method of convergence.

We note from our analysis that knowing and controlling the amount of circumvention traffic BR relative to the cover traffic, *e.g.* by selecting cover protocols carefully, can allow

the game to stabilize on a CRS-friendlier Nash equilibrium. Unfortunately, we do not yet have a systematic way to learn this value. The next chapter addresses this shortcoming.

Chapter 4

Privacy-preserving Collection of CRS Statistics

Portions of this chapter were previously published in the proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security [EDG14].

4.1 Introduction

We now turn our attention to providing an empirical basis for CRS design. Today, there are many popular anonymity networks and services such as Tor [DMS04a], JAP [KH04] (commercially offered as JonDonym [Jon13]), i2p [jra03] and Anonymizer Universal [Ano13].

These networks provide the link obfuscation CRS security property. This obfuscation is implemented with relays that form a communication path between a client and a destination that hides information about who is connecting to whom, both from network observers as well as from the destination itself. While they have been improved upon and have grown in popularity, anonymity networks remain notorious for being difficult to study [Loe09, Win13]. This is partly due to their inherent privacy properties, but also due to ethical considerations: they are live systems, and any data collection about their use may put in danger real users by compromising their anonymity.

Unfortunately, previous research on client behavior [MBG+08] led to controversy due to private client information being gathered—even though it was destroyed and never exposed [Sog11]. This set a precedent that client information, no matter how it is collected, is off-limits for legitimate research, which had a chilling effect on research in this area.

Even with the risks, there are three main motivations for collecting empirical data. The first is that developers of anonymity networks have so far been unable to inspect egress trends. This information can guide designs that enhance performance and provide features that better address the needs of users. For example, network designers would like to be able to determine how much of the network's traffic is for the purpose of protecting the user's identity from the website she visits, and how much is for the purpose of censorship circumvention—protecting the identity of the website she visits from the *censor*. These different user bases have different security and privacy requirements, and knowledge of the prevalence of each set can help tune the network appropriately. The second motivation is to inform the research community with realistic information about usage trends to guide research in censorship resistance mechanisms, performance tuning, and resource allocation. Finally, one of the important open questions in any anonymity network is how to model client behavior since this is exactly the information that needs to remain confidential. With realistic statistics we can shed light not only on client behavior but also use it to ensure that when we test novel designs or system changes we can model their effects on clients in a more realistic manner, leading to more ecologically valid results.

In order to reap these benefits, data-collection systems must then be mindful of four main risks:

1. The network is run by volunteers and *anyone* with resources may join the network by contributing nodes with bandwidth or computation cycles to relay traffic. This limits the trustworthiness of nodes.

2. The data that may be collected at nodes is sensitive and directly publishing it may break the non-collusion assumption required by relay-based anonymity networks to maintain user anonymity.

3. The nodes that collect or process statistical information should not become targets of compulsion attacks by making them more attractive targets of miscreants and authorities.

4. Low-latency anonymity networks are vulnerable to correlation attacks [MD05, ØS06, JWJ+13] that observe traffic volumes entering and leaving the network. Published statistics must hide information that would allow a client-side adversary with a partial view of the network (an ISP, for example) to mount such attacks.

To mitigate the risks, we propose *PrivEx*, a system for collecting aggregated anonymity network statistics in a privacy-preserving manner.

PrivEx collects aggregated statistics to provide insights about user behavior trends by recording aggregate usage of the anonymity network. To further reduce the risk of inadvertent disclosures, it collects only information about destinations that appear in a list of known censored websites. The aggregate statistics are themselves collected and collated in a privacy-friendly manner using secure multi-party computation primitives, enhanced and tuned to resist a variety of compulsion attacks and compromises. Finally, the granularity of the statistics is reduced, through a noise addition method providing (ϵ,δ)-differential privacy, to foil correlation attacks.

The novel contributions in *PrivEx* are:

1. A safe mechanism to collect client statistics from anonymity network egress nodes;
2. Two secure multi-party protocols that protect the intermediate values of a distributed differential privacy (DDP) computation, optimized for the problem at hand;
3. Reduced noise in the results of the DDP computation leading to higher utility while still maintaining the desired level of privacy, both as tunable parameters;
4. A security analysis detailing the resistance to compulsion, compromise and correlation attacks;
5. An evaluation of the overhead and performance of a proof-of-concept implementation of *PrivEx*; and
6. Preliminary analysis of data collected from a limited deployment.

4.2 Background

Anonymous Communication Networks (ACN). Anonymous communication networks allow clients to hide their accesses to web pages and other Internet destinations from certain network observers (typically ones who can view network traffic on at most a small portion of the Internet).

Low-latency networks, such as Tor, JAP/JonDonym, or i2p, obfuscate network traffic by routing it through multiple nodes: an *ingress* node, some number of *middle* nodes, and an *egress* node. The routing can be predetermined by the network, as in JAP/JonDonym, or source-routed subject to some constraints, as in Tor and i2p. To achieve security against network observers, traffic is encrypted so that the contents and metadata, including the

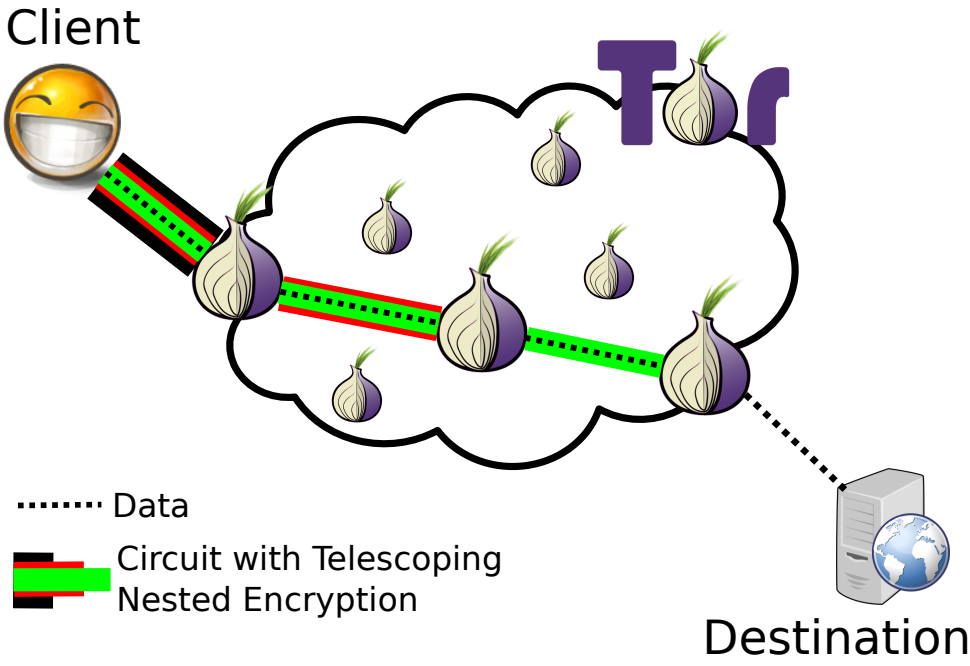


Figure 4.1: An overview of the Tor network and typical traffic flow (dotted line), highlighting Tor circuits, which use telescoping nested encryption.

destination, are hidden from the ingress and middle nodes, as well as anyone observing the ACN ingress or internal traffic.

Simpler anonymizing networks, such as Anonymizer Universal, use only a single node and as a result are extremely susceptible to legal compulsion attacks (through court orders, for example) [Sin07, Pou13]; hence, they will not feature in our discussions further.

Tor. Tor [DMS04a] is a popular ACN that provides anonymity by decoupling the routing information between the client and the destination. Clients use three intermediary nodes to route their traffic using onion routing. This prevents the destination from learning who the client is, and it also prevents an observer local to the client from learning which destination the client has connected to.

Tor, by default, uses three intermediate nodes in a connection between a client and destination (Figure 4.1). The client uses a telescoping mechanism to construct a *circuit* between herself and the last node, known as the exit node, which is the egress point of

the client's traffic. As this is the location where *PrivEx* will perform its privacy-preserving data collection, we will refer to this node as the data collector (DC) in the remainder of the paper. Each client circuit has a unique identifier to help the DC manage the flow of traffic for multiple clients at once. The default behavior is for the Tor client software to switch to a new circuit every 10 minutes.

The DC node knows the destination but not the originator of a connection. This is necessary to prevent it from relating the observed destination to any client and hence learn about her habits, activities or interests. Traditionally, exit nodes are required to delete any information about the connections that exit the Tor network through them. Publishing such information may be combined by an adversary (such as an ISP or national firewall) with a client-side view of the network to correlate exit activity with client activity to deanonymize the network traffic.

Thus, to not introduce any new attack vectors, any effort to collect traffic data at exit nodes, even in aggregate, will have to minimize the information leaked to the adversary. This must hold even in the case that the adversary is able to compromise the node or compel the node operator to reveal the state of the node.

We will use Tor as a model ACN in which to integrate *PrivEx* in the discussions that follow. This should aid in clarifying the descriptions and to help the reader relate *PrivEx* to real-world ACNs, but does not restrict the generality and applicability of *PrivEx* to other systems.

Differential Privacy. Traditional differential privacy [Dwo06] protects a sensitive central database—a table where rows hold sensitive data about individuals—that is to be kept private. This central database holds *raw* records that are only to be released to the public in noisy or aggregated form. The database allows multiple queries from clients who spend from a *privacy budget* for each query.

Established differential privacy mechanisms add noise to the results of client queries to ensure that personal information—*i.e.*, information about a particular entity that contributes to the results of a query—cannot be gleaned from those results. Intuitively, given any two "neighbouring" databases, one containing an entity's data and another without that entity's data, but otherwise equal, then the probability of observing any particular output to a given query will be close for the two databases.

PrivEx implements a privacy mechanism based on adding noise from a Gaussian distribution.[1] Adding an appropriate amount of Gaussian noise to the results of queries produces (ϵ, δ)-differential privacy: if D and D' are two neighbouring databases (as described

[1]We discuss later why we use Gaussian instead of Laplacian noise.

above), then the probabilities $P_D(x)$ and $P_{D'}(x)$ that a given query outputs x when using the databases D and D' respectively, are related by $P_D(x) \leq e^{\epsilon} \cdot P_{D'}(x) + \delta$. [DKM+06]

In our setting, the database consists of one row for each censored website whose visits we wish to count, and queries will simply be of the form "output the counts in each row of the database (plus noise)".

4.3 Threat Model

PrivEx maintains its security properties against an adversary that is local to the client or the website servers they visit. The adversary is able to monitor traffic between the client and the ingress of the anonymity network, or traffic between the egress of the network and the client's destination, but not both at the same time. This assumption is similar to the one required to argue Tor is secure. As a result, this adversary is presumed to be unable to defeat the anonymity system. However, if any information is also revealed by the DC node, such as client usage statistics, that data could possibly be used to correlate traffic. A secure statistics gathering system, like *PrivEx*, should prevent any such attacks.

We allow the adversary to operate nodes in *PrivEx*; *i.e.*, deploy or compromise nodes in the network and be part of the aggregation service itself. The adversary may also use the anonymity network to relay its own traffic in order to induce desired statistics into the aggregation process. Malicious nodes can report spurious data without generating or processing the corresponding traffic.

PrivEx is secure when there is at least one honest data collector and at least one honest-but-curious tally key server (described in Section 4.4). While dishonest data collectors can report "junk" statistics and malicious servers can disrupt the protocol, the security requirement in *PrivEx* is that no client traffic pattern information from honest data collectors is ever exposed: neither while it is stored on the data collectors, while it is in transit in the network, nor while it is being processed by the aggregating service. That is, malicious parties can disrupt the statistics reported by *PrivEx*, but cannot expose private data. In the distributed-decryption variant of *PrivEx* (see Section 4.4.2), we can further *detect* misbehaving servers, apart from those that collect the actual data. We discuss the security implications of malicious actors and publishing client statistics in further detail later in Section 4.5.1.

4.4 The *PrivEx* Schemes

The goal of *PrivEx* is to count how many clients are visiting each of a list of particular known censored websites.[2] This is to establish an approximation of the base rate of CRS-related activity on the network. These statistics are gathered and reported in a privacy-sensitive manner so that the outputs of *PrivEx* cannot be used to perform traffic correlation attacks. Note that it is straightforward to adapt *PrivEx* to collect statistics for any type of event that the egress nodes can count, such as the traffic volume per circuit, variance in circuit-management statistics, client navigation behavior, and so on.

The DC nodes in *PrivEx* run on the same machines as the egress nodes of the underlying ACN. The DC listens for events of interest from the egress node, and securely aggregates them. In our setting, an event will consist of the ACN egress node reporting that a particular circuit has asked to perform a DNS lookup of a particular website.

PrivEx collects and aggregates statistics over a fixed period of time, called an *epoch*. We pick an epoch according to the granularity of the statistics we wish to collect—for our example ACN, Tor, we have chosen one hour as the epoch, to match the current frequency with which the Tor client updates their CRS information.

We introduce two *PrivEx* scheme variants that provide secure and private aggregate statistics of events collected by the DCs. They differ in the cryptographic primitives used to protect the data while it is in storage and in the protection that they offer against malicious actors.

The first scheme, based on secret sharing (*PrivEx*-S2), is secure in the honest-but-curious setting but can be disrupted by a misbehaving actor. The second scheme, based on distributed decryption (*PrivEx*-D2), is secure in the covert adversary setting in that misbehaving servers can be identified. Most importantly, however, in both schemes, the disruption of the protocol by malicious parties does not result in information leakage.

4.4.1 *PrivEx* based on Secret Sharing

There are three types of participants in *PrivEx*-S2: Data Collectors (DCs), Tally Key Servers (TKSs), and a Tally Server (TS). The DCs relay traffic between the ACN and the destination; they collect the statistics we are interested in. TKSs are third parties who combine and store the secret shares received from DCs and relay aggregates of those secret

[2]This list can optionally have an "Other" entry to count the *total* number of visits to non-censored websites as well.

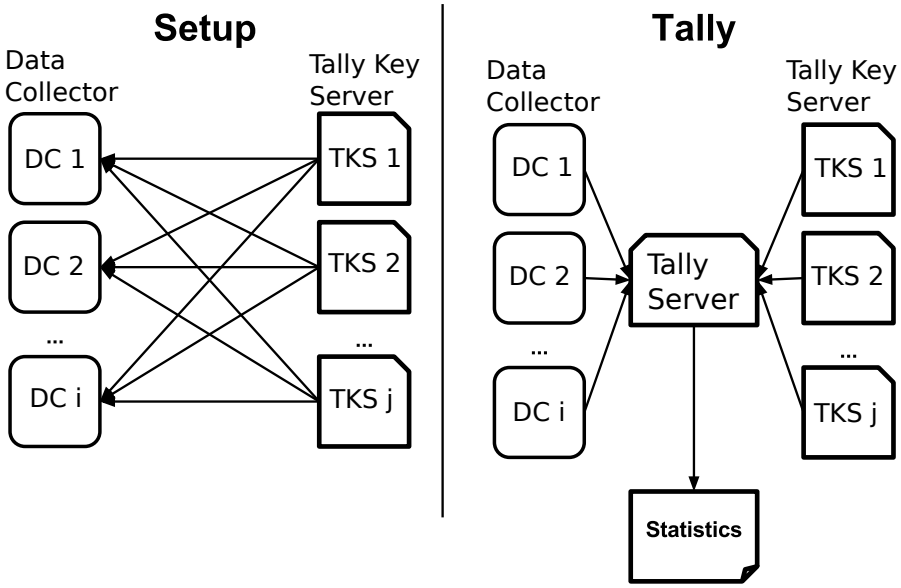


Figure 4.2: *PrivEx*-S2 variant based on secret sharing.

shares to the TS. The TS simply adds up the secret shares provided by the DCs and the TKSs to produce the aggregated results. Figure 4.2 depicts an overview of our scheme.

Setup. At the beginning of every epoch, each pair of DC (i) and TKS (j) share a secret key (K_{ij}). This key can be the result of an ephemeral Diffie-Hellman exchange, or more simply, each DC i can seed each TKS j with a shared key through a secure channel (*e.g.*, TLS 1.2 using a ciphersuite that provides forward secrecy).

Each DC maintains a number of secure counters, each cryptographically storing the count of accesses to a specific destination (wID). The DC cryptographically initializes a database of records, each representing a secure counter, with the following schema: [wID, C_{wID}] where

$$C_{\text{wID}} = \left(n_{\text{wID}} - \sum_j \text{PRF}(K_{ij}; \text{wID}) \right) \bmod p$$

Here, n_{wID} is the noise for this counter (see Section 4.4.4), PRF is a keyed pseudorandom function, and p is a smallish prime (such as $p = 2^{31} - 1$). After this step, the DCs securely delete their shared keys K_{ij} and the noise n_{wID}.

Each TKS (j) also uses K_{ij} to compute its contribution to the count for each wID as:

$$S_{\text{wID}} = \left(\sum_i \text{PRF}(K_{i,j}; \text{wID}) \right) \bmod p$$

and then securely deletes its copy of the K_{ij}. Alternatively, in order to mitigate failing DCs, the TKSs can store the keys until the tally phase but this opens up the TKSs to compulsion attacks to reveal the keys, and hence the individual DC statistics.

Counting. Upon a DNS lookup for a domain on the censored website list, the DC simply adds 1 to the appropriate secure counter as follows: $[\text{wID}, C'_{\text{wID}} = (C_{\text{wID}} + 1) \bmod p]$. We choose p large enough to expect no overflow of counting events—we can only reliably aggregate up to p events per counter.

Aggregation. At the end of every epoch, all the DCs and all the TKSs send their databases of secure counters to the TS.

The TS simply adds up all the shares received from the DCs and TKSs and publishes the results, which are the aggregated destination visit totals from all the DCs plus the total of the noise added by each DC at the setup stage in each counter. Once the results are published for the current epoch, the tally server deletes the received data and awaits the next epoch's data to tally.

After sending their data for the epoch to the tally server, all the DCs and TKSs securely delete their databases and reinitialize through the setup phase, starting the cycle again.

4.4.2 *PrivEx* based on Distributed Decryption

We now describe *PrivEx*-D2, depicted in Figure 4.3. *PrivEx*-D2 utilizes the Benaloh encryption scheme [Ben94]—a distributed additive homomorphic encryption scheme. This scheme is a variant on ElGamal: a (private,public) key pair is $(a, A = g^a)$ and an encryption of a message $m \in \mathbb{Z}_q$ with randomness $r \in \mathbb{Z}_q$ is $E_A(r; m) = (g^r, A^r \cdot h^m)$, where g and h are generators of a cryptographic group $\mathbb{G}$ of order q. Note the additive homomorphism: $E_A(r_1; m_1) \cdot E_A(r_2; m_2) = E_A(r_1 + r_2; m_1 + m_2)$, where the multiplication is componentwise. Decryption is $D_a(C_1, C_2) = DL_h(C_2/C_1^a)$. Note that decryption requires the taking of a discrete log, but if the message space M is small (as is the case for counts of website visits, or in Benaloh's original application, counts of votes), this can be done

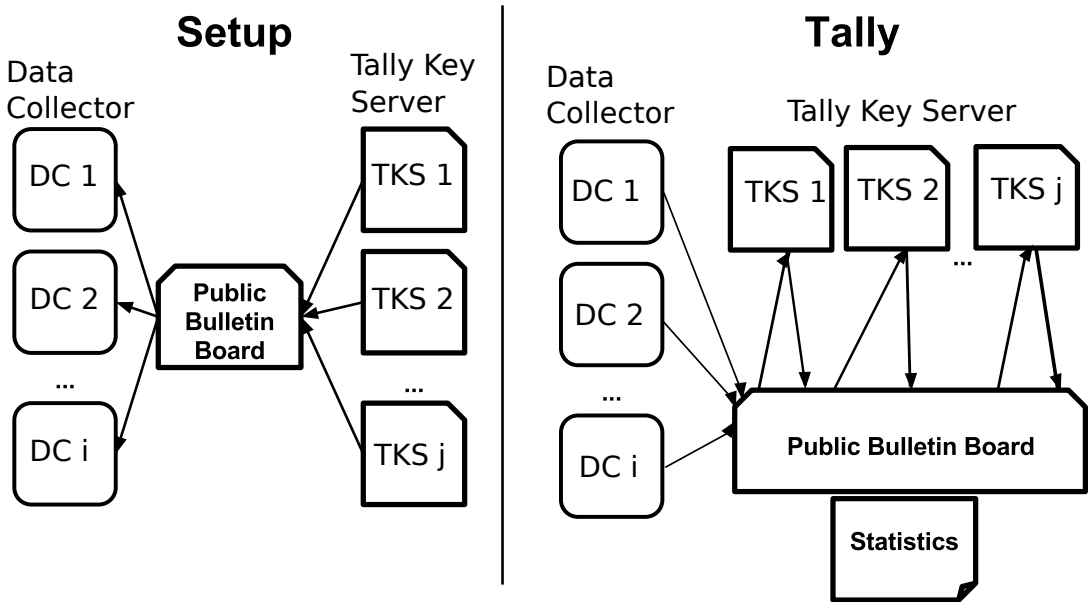


Figure 4.3: *PrivEx*-D2 variant based on distributed decryption.

with the kangaroo [Pol78] or baby-step-giant-step [Sha71] methods in time $O(\sqrt{|M|})$, or even faster if more space is consumed by a pre-computation table.

Note that *PrivEx*-D2 uses a public bulletin board (PBB) instead of a Tally Server; the PBB is used as a repository of results and public keys from the DCs and TKSs. We can instantiate it with a database server which maintains tables for the TKS public keys and intermediate decryption results, and the final statistics of the epoch. To mitigate misbehavior by an untrusted PBB, the messages stored thereon should be digitally signed by their authors using long-term authentication keys.

Setup. At the beginning of every epoch, each TKS (j) picks a random (ephemeral) private key $a_j \in \mathbb{Z}_q$ and computes its public key $A_j = g^{a_j}$. They publish the public keys to the PBB, along with a non-interactive zero-knowledge proof of knowledge (using the Fiat-Shamir heuristic) of the private key a_j. Each DC then checks each proof, and calculates the compound key A by taking the product of all the published keys: $A = \prod_j A_j$. Now each DC, for each secure counter for website w in its table, computes the amount of noise n_w to be added (see Section 4.4.4), and stores $E_A(r_w; n_w) = (g^{r_w}, A^{r_w} \cdot h^{n_w})$. Note that the randomness r_w will be freshly chosen for each counter, and discarded immediately after encryption, along with the plaintext n_w.

Counting. When the DC observes a visit to a website under observation, it multiplies (component wise) the appropriate encrypted counter by $E_A(r;1) = (g^r, A^r \cdot h)$ where r is random.[3] After c_w visits, the secure counter will hold $(g^r, A^r \cdot h^{c_w+n_w})$ for some r. It can optionally also re-randomize the all the other counters to ensure that two subsequent snapshots of the database do not reveal which counter has been incremented.

Aggregation. At the end of the epoch, each DC (i) publishes to the PBB a commitment to its encrypted counters for each website (w): $C\left(\langle(g^r_{i,w}, A^{r_{i,w}} \cdot h^{c_{i,w}+n_{i,w}})\rangle_w\right)$, where C is an appropriate commitment function. After all DCs have posted their commitments to the PBB, each posts the opening of its commitment (the list of encrypted counters $\langle(\alpha_{i,w}, \beta_{i,w})\rangle_w = \langle(g^r_{i,w}, A^{r_{i,w}} \cdot h^{c_{i,w}+n_{i,w}})\rangle_w$). Each TKS j then checks that the DCs' openings are consistent with their commitments, and consolidates the openings by computing $\alpha_w = \prod_i \alpha_{i,w}$. It then computes $\alpha_w^{(j)}$, TKS j's share of the decryption, as $\alpha_w^{(j)} = (\alpha_w)^{a_j}$, and posts that back to the PBB, along with a non-interactive zero-knowledge proof of equality of discrete logarithms to (g, A_j) to show that the computation was correct. Everyone can then check the proofs and compute the value $h^{c_w+n_w} = \left(\prod_i \beta_{i,w}\right) / \left(\prod_j \alpha_w^{(j)}\right)$, where $c_w = \sum_i c_{i,w}$ and $n_w = \sum_i n_{i,w}$. From here, $c_w + n_w$ can be computed using one of the discrete logarithm algorithms mentioned above. A proof of security for *PrivEx*-D2 can be found in Section 4.5.3.

Filtering Statistics by Client Origin

So far, we have assumed there is a single list of censored websites whose visits we are interested in counting. However, different websites are censored in different countries, and we may wish to count a visit to, say, Wikipedia if the user is in China, but not in the UK, a visit to the Pirate Bay if the user is in the UK, but not in Sweden, etc.

In this section, we present an extension to the *PrivEx*-D2 protocol that allows us to maintain *per-country* lists of censored websites, and only count a visit by an ACN user to a given website if that website appears on that user's country's list.

To do this, we of course need to determine what country the user is in. This is best done at the ingress point to the ACN, where the true IP address of the user is visible. Indeed, Tor already collects this information so that it can display per-country counts of numbers of users. [Tor10b] It is of course vital that the DC *not* learn this potentially

[3] For a slight efficiency gain, r can be 0, so that the multiplication is by $(1, h)$. The downside is that this can leak information to an attacker that can observe the internal state of a DC at two different times within one epoch, yet cannot observe that DC's DNS lookups.

identifying information about the client. The ingress node will therefore forward to the DC an *encrypted vector* encoding the country. The length of the vector is the number of countries N_C for which we are monitoring accesses to censored websites, plus one for "Other". The vector is then $V = \langle E_A(r_c; \delta_{c,c^*}) \rangle_{c=0}^{N_C}$ where c^* is the country the user is in and δ_{c,c^*} is 1 if $c = c^*$ and 0 otherwise. The r_c are uniform random elements of $\mathbb{Z}_q$. The ingress node also provides a zero-knowledge proof that each element of the vector is an encryption of either 0 or 1, and that the sum of the plaintexts is 1. We note this is the same proof as used in electronic voting schemes, for example. [Ben94]

The DC will check the zero-knowledge proof, and when it observes a connection to, say, Wikipedia, will multiply into its count not $E_A(r; 1)$, as above, but rather $\prod_c V_c$, where the product is over those countries c that censor Wikipedia. The remainder of the protocol is unchanged. Each vector V is associated to a circuit at circuit construction time and the DC knows which circuit requested the website.

4.4.3 *PrivEx* Scheme Comparison

Both schemes provide the security features we desire, but in some settings one may be preferable over the other.

In volunteer-resourced ACNs, such as Tor, some nodes will inevitably have low computation and bandwidth resources and it is best to minimize their computational, memory, and bandwidth overhead. In such cases, *PrivEx*-S2 is preferable since some messages are overall shorter and the computational overhead of frequent operations is smaller.

The length of the epoch can affect our choice of scheme since the relative time to set up and process the statistics increases for shorter epochs. While it is not a current requirement, if we wanted more near-real-time statistics, say every 5 seconds, then we would prefer *PrivEx*-S2 since the overhead is nearly negligible compared to *PrivEx*-D2. There are limits to how short the epoch can be, however, due to network latency affecting protocol communication.

On the other hand, *PrivEx*-D2 provides traitor detection of the TKSs and Denial of Service (DoS) resistance. In *PrivEx*-S2, any DC or TKS can DoS the system for the epoch if it does not report its statistics, whereas in *PrivEx*-D2 only DCs that report statistics for the epoch are included in the aggregation process and misbehaving TKSs (traitors) can be detected using cryptographic proofs ensuring that the computations were done correctly. Furthermore, *PrivEx*-D2 can optionally enjoy stronger perfect forward secrecy—against node seizure and adversaries that can view the memory contents multiple times in an epoch—by re-randomizing even those counters that have not been changed with every increment operation.

4.4.4 Calculating and Applying Noise

We introduce noise to our results to mitigate the risk of the correlation attack that reporting exact results may introduce. A more thorough discussion of the correlation issue is found in Section 4.5.2. In this section, we present the details of how the appropriate amount of noise is computed and added to the tallies.

How Much Noise?

We add noise to protect the privacy of users, but at the same time, if we add too much noise, it will hamper the *utility* of *PrivEx*; after all, we are deploying *PrivEx* to answer certain questions about ACN usage. We adopt a principled approach to adding noise to our statistics—one that allows the level of privacy and utility to be set to desired levels. For this purpose we have selected the model of differential privacy that can provide (ϵ, δ)-differential privacy through the addition of noise using a Gaussian mechanism with mean 0 and a standard deviation σ selected for the level of privacy and utility we require.

We wish to protect information about whether any individual user's information is in the published data set, or is not in it. To do this, we need to set an upper bound—called the sensitivity (S)—on the maximum contribution one user can make to the count in any epoch. For Tor, we use the fact that, by default, one circuit is created every ten minutes, so that if our epoch length is, say, one hour, and we always ignore repeated visits to the same website by the same circuit, we can set $S = 6$—the security implications of implementing this are discussed in Section 4.5.1. For other ACNs, an appropriate sensitivity can be similarly selected.

As we are interested in practical applications of *PrivEx*, we provide the means to calculate the exact values of ϵ and δ through the lens of the privacy and utility levels we desire.

What we are interested in controlling is the *advantage* (over random guessing) of an adversary in guessing whether a particular user's data is contained in the published (noisy) statistics, *even if the adversary knows all the other inputs to the statistics*. That is, discounting the known information, the adversary is trying to determine whether the published statistics are more likely to represent a true measurement of 0 (the user is not present) or S (the user is present).

Therefore, the adversary's task is to tell if a given statistic is drawn from the distribution $N(0, \sigma)$ or $N(S, \sigma)$. Given a reported statistic, if it is less then $\frac{S}{2}$, the adversary's best guess is that the true statistic is 0, and S otherwise. It is easy to see that the advantage

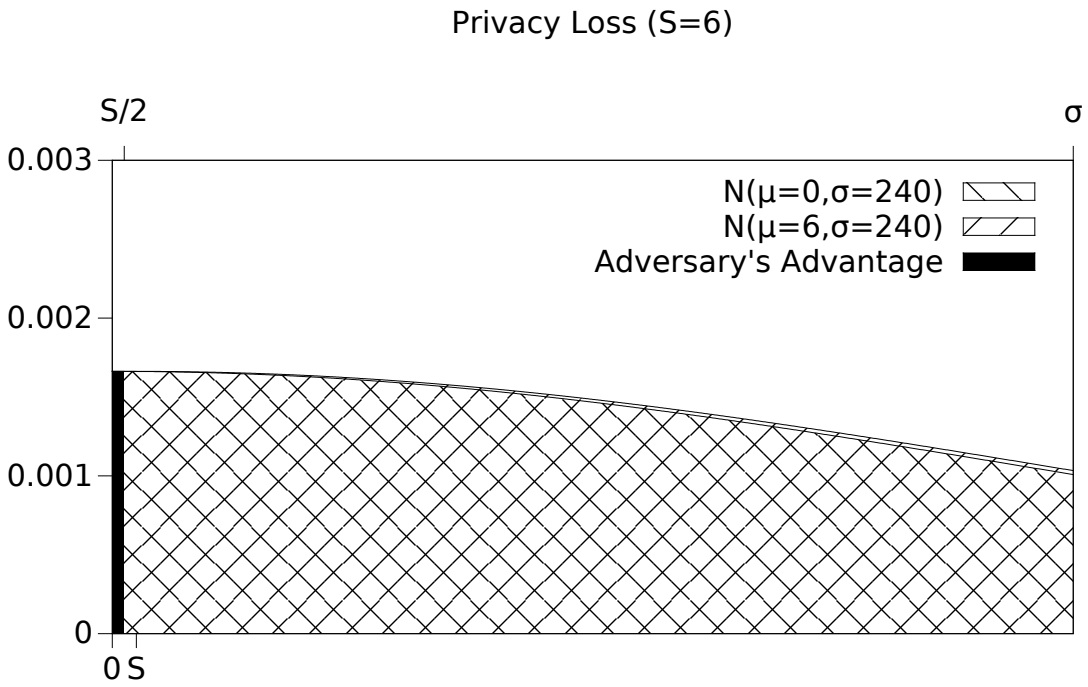


Figure 4.4: The advantage is 0.5% (shaded area) of the adversary in guessing the correct value of the statistic. Note the almost total overlap of the two probability distributions.

of the adversary is then given by the area under the $N(0, \sigma)$ normal curve between 0 and $\frac{S}{2}$, as depicted in Figure 4.4.

The adversary's advantage can then be minimized by selecting σ large enough such that $Pr[0 < N(0, \sigma) < \frac{S}{2}] = Pr[0 < N(0, 1) < \frac{S}{2\sigma}]$ is as close to 0 as desired. However, choosing σ too large will hamper utility, as we discuss next.

To address our utility needs, we must first decide on a question to ask. A typical question would be, "On average, how many visits are there to a given censored website per epoch?", and we may be content to know the answer to within some *resolution* K, say 100 or 1000. This gives us two benefits over the privacy adversary: we only care about average behavior over many epochs, and not specific users at specific times (in order to carry out a correlation attack); and we only care about results to within K, not to within single users' contributions.

If we average over λ epochs, the standard deviation of our noise becomes $\frac{\sigma}{\sqrt{\lambda}}$. Then, if we want to distinguish two hypotheses that differ by K (*e.g.*, does this website see closer to 0 visits per epoch or closer to $K = 1000$ visits per epoch over the ACN—a question we cannot

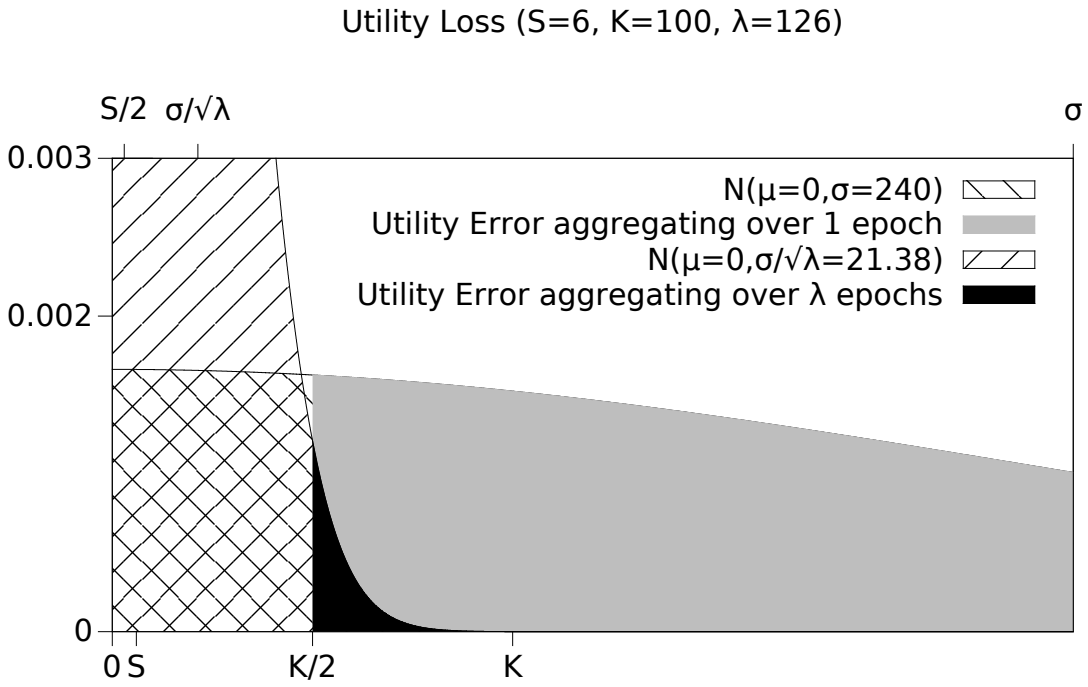


Figure 4.5: The probability of error is 0.1% (dark shaded area) when the reported statistic (averaged over $\lambda = 126$ epochs) appears closer to K than to 0. Compare this to the much larger error of 41.75% (lighter shaded area) when $\lambda = 1$.

answer today), our *utility error*—the probability we answer incorrectly—is $Pr[N(0, \frac{\sigma}{\sqrt{\lambda}}) > \frac{K}{2}] = Pr[N(0,1) > \frac{K\sqrt{\lambda}}{2\sigma}]$, as depicted in Figure 4.5. Slightly different questions would produce slightly different formulas for the utility error, but they will be computable in the same vein.

Therefore, for a given sensitivity S and tolerance P on the advantage of the privacy adversary, we can work out the desired standard deviation σ for our noise by solving for $Pr[0 < N(0,1) < \frac{S}{2\sigma}] \leq P$ using a standard normal curve z-table. Then, given a tolerance U on the utility error, and a resolution K for our question, we can determine the number of epochs λ we will need to average over by solving for $Pr[N(0,1) > \frac{K\sqrt{\lambda}}{2\sigma}] \leq U$ similarly.

In the presence of possibly malicious DCs, the situation is only slightly more complicated. Malicious DCs (who do not immediately forget the amounts of noise with which they initialized the secure counters) know the amount of noise they added. By removing that from the reported tally, the remaining amount of noise (contributed by the honest DCs) is less than expected.

As we will see in Section 4.4.4, each DC i adds noise selected from a normal distribution whose standard deviation is proportional to its *weight*—the probability w_i that that DC will be selected by a user. If we can assume a lower bound H on the total weight of honest DCs, we can adjust the above calculations in a simple manner. (In Section 4.5.1 we will argue that $H = 0.8$ is a reasonable lower bound for Tor.) Honest DCs tune the amount of noise to add by adjusting the value of σ to $\sigma_H = \frac{\sigma}{H}$. This has the effect that honest DCs add more noise so that it maintains the desired privacy level, at the expense of requiring an increase in λ by a factor of H^{-2} (an increase of about 56% for $H = 0.8$) to achieve the same level of utility as before.

A Worked Example. Using Tor as our ACN, and one-hour epochs, so $S = 6$, we want to find σ given a desired privacy adversary advantage of at most 0.005. Consulting a z-table, we find that we want $\frac{S}{2\sigma} \leq 0.0125$, so $\sigma \geq 240$. Then, if we want utility error $U = 0.01$, the z-table says we need $\frac{K\sqrt{\lambda}}{2\sigma} \geq 2.33$, so for $\sigma = 240$, $K\sqrt{\lambda} \geq= 1120$ will suffice. Then if $K = 1000$, λ can be as low as 2 epochs, if $K = 100$, then $\lambda = 126$ epochs (or 5.25 days), but to get an average number of visits per epoch to within $K = 1$, we would need over 140 *years*.

We now analyze the case where some fraction of DCs may be malicious. Assume that we expect that the total honest weight is at least 80%. We adjust σ to $\sigma_H = \frac{\sigma}{H} = 240/0.8 = 300$. Then, for the same utility error as above, $K\sqrt{\lambda} \geq 1400$ will suffice. For the same values of K we would now need 2 epochs, 8.2 days, and over 224 years respectively.

In the preceding analysis we only need consider the amount of noise to add in terms of the standard deviation σ of the distribution we sample from. We can link this back to (ϵ, δ)-differential privacy by observing the parameters' relation to σ as follows [HR12]:

$$\sigma = \frac{S}{\epsilon} \cdot \sqrt{\ln\left(\frac{1.25}{\delta}\right)}$$

Thus, rather than, as in previous works [HR12, DKM+06] , having the system designer select not-very-meaningful values of ϵ and δ, and computing σ as above to determine how much noise to add, we *instead* determine σ directly using parameters specifically pertinent to the system and to the questions it is trying to answer.[4]

[4]Since we only ever make one query we do not need to calculate how much privacy budget we have left after publishing our aggregated statistics.

Distributed Noise Application

The DCs independently apply the noise as we never want the raw (un-noisy) data to be divulged. We can distribute the application of noise since we know from Dwork *et al.* [DRV10] that if individual databases are differentially private then so is their sum.

A naive way to go about this, and one that avoids the use of third parties, is for the DCs to publish their noisy data directly to the public. The consequence of this is that each DC would need to add enough noise so that its *individual* statistics provided the desired bound on the advantage of the privacy adversary. This would make the total noise considerably larger (by a factor of the square root of the number of DCs), and so the number of periods λ to average over must increase by a factor of the number of DCs in order to keep the desired bound on the utility error.

This is why *PrivEx* works with *global* noise instead of *local* noise: each DC adds some amount of noise, whose *total* is distributed as $N(0, \sigma)$ for the desired σ, but does so using secure multiparty computation so that the individual noise components are never revealed.

We then need to calculate how much noise *each* DC should add. What we want is for each DC i to add noise from $N(0, \sigma_i)$, where σ_i is proportional to the probability w_i that the DC will get used. In Tor, for example, high-bandwidth nodes get used with higher probability, so they will see more usage, and add more noise, while more impoverished nodes will have less usage and less noise.

Then, given the desired σ, we want to solve for the σ_i such that $\sigma_i \propto w_i$ (so $\sigma_i = w_i \cdot \phi$ for some ϕ independent of i) and $\sum_i N(0, \sigma_i) \sim N(0, \sigma)$. Since $\sum_i N(0, \sigma_i) \sim N(0, \sqrt{\sum_i \sigma_i^2})$, we have that $\sigma^2 = \sum_i ((w_i \cdot \phi)^2)$, so solving for ϕ, we find that $\sigma_i = w_i \cdot \phi = \sigma \cdot \frac{w_i}{\sqrt{\sum_i (w_i^2)}}$. In *PrivEx*, the values of ϕ and σ are made available to the DCs from the PBB or TKSs.

That we are adding together a potentially large number of independent noise sources is the reason we target Gaussian rather than Laplacian noise: while adding many Gaussians yields a Gaussian, a Laplacian distribution cannot be decomposed into sums of other independent random variables.

We note that, when adding noise, it is important for each DC to preserve non-integral and negative values for the noisy count, so that, when added together, extra biases are not introduced. As the encryption used in our counters takes integer plaintexts, we must use a fixed-point representation where all of our values are expressed as multiples of some small number γ. If there are N DCs, then in order that adding N values of resolution γ together will be unlikely to produce an error of more than 1, we set $\gamma \leq \frac{1}{2\sqrt{N}}$.

For $N \approx 1000$, as in the current Tor network, $\gamma = 0.01$ will suffice.[5] Note, however, that this fixed-point representation expands the plaintext space by a factor of $\frac{1}{\gamma}$, and so increases the time to compute the discrete logarithm in the final step of the *PrivEx*-D2 protocol by a factor of $\frac{1}{\sqrt{\gamma}}$.

Targeted Temporal Queries

PrivEx publishes the noisy total statistics for each epoch. The amount of noise is computed to protect privacy, and a number of epochs' statistics must be averaged to gain utility. However, these epochs do not need to be consecutive, so, for example, one could ask questions like, "Is Wikipedia visited via this ACN more often on weekends or weekdays?". The *number* of epochs to average will not change, however, so if the epochs of interest are spread out in time, the total time to answer such a question will increase.

4.5 Security Analysis

4.5.1 Resistance to Attacks

We now address the attacks that are of the most concern. Recall that our requirement for security is that *PrivEx* should not reveal private information to an adversary, even if it fails to produce meaningful answers to the system designers' questions. Of course, we also require that *PrivEx* produce meaningful answers in the *absence* of an adversary.

Legal or Other Compulsion

A DC can be compelled to reveal its database of collected statistics through a legal order or extra-legal compulsion. If this database is stored in the clear then privacy would be violated. *PrivEx* mitigates this threat by storing an encrypted database with the property that the DC cannot decrypt the database on its own. Recall that at the setup stage in *PrivEx*, all DC databases were encrypted using shared keys with, or public keys of, the tally key servers.

The adversary can also compel the servers to comply in the decryption of individual DCs' measurements (with less noise than the aggregate). This would indeed be troublesome, but we mitigate this by ensuring that the *PrivEx* servers are distributed across

[5] This also deals with an issue with rounding and differential privacy identified by Mironov. [Mir12]

diverse legal boundaries making compulsion infeasible. Indeed, as long as at least one server is uncompromised then all DC data is safe. Furthermore, since we start with fresh keys for each epoch, this compulsion could not occur retroactively.

PrivEx requires that we bound the sensitivity—the maximum number of times one client can access a particular website in one epoch. We do this by maintaining, in plaintext, a list of websites visited during the *lifetime* of a circuit, which is 10 minutes in Tor. This introduces a potential information leak if the adversary is able to compromise an honest DC *while* circuits are being served; this would reveal the censored websites visited by each circuit. While this in itself does not link a client to a destination an adversary may use this information to correlate traffic patterns it can record at the client side of the circuit. However, if the adversary can compromise an ACN relay while it is actively serving an open circuit, then the encryption keys it could recover could compromise those circuits anyway even without access to the plaintext list.

Malicious Actors

Data Collector. The DC can behave maliciously by reporting untrue statistics. While there is no safeguard to an attack on the integrity of the statistics we are interested in, the confidentiality of the statistics collected at other DCs and the aggregate statistics that are output by *PrivEx* are safe from the actions of a misbehaving DC as long as the security of the encryption schemes that we use remains intact. We may mitigate the impact of this attack by using range proofs at additional computation and communication costs, but this still does not remove the threat entirely. In Section 4.4.4 we suggested that $H = .8$ is a reasonable lower bound on the amount of honest DC weight for Tor. The reason we give this value is that if more than 20% of the exit weight of Tor is compromised, then Tor is sufficiently susceptible to circuit linking attacks [ABEG13], and could more easily compromise clients without using the less-noisy statistics provided by the degraded *PrivEx*.

Finally, we note that if a DC is compromised, the adversary can also perform a correlation attack, and can likely read the memory, including encryption keys protecting any active circuits, thus retroactively deanonymizing them. This is a shortcoming of the underlying ACN; *PrivEx* does not exacerbate this problem.

Tally Key Server. The tally key servers collectively play a critical role in the *PrivEx* schemes and hence are vectors of attack. A bad actor may try to gain access to the statistics in a less secure manner or an insecure intermediate form (*i.e.* without noise).

We guard against this in both variants of *PrivEx* by ensuring that in the setup stage all DCs initialize their databases by encrypting each secure counter using the key material

provided by, or shared with, all the participating TKS servers. This ensures that even if all but one TKS try to decrypt the data in an information-leaking manner, a single honest server's key material and noise added by the DCs prevents any information from being revealed.

In *PrivEx*-S2, a single DC or TKS can launch a denial of service attack by not sending its share, which would mean that for that epoch no results could be determined. In *PrivEx*-D2, we can identify the misbehaving TKS, which introduces consequences to DoSing. In either case, no private information is leaked.

Tally Server and Public Bulletin Board. The TS and PBB are unable to learn anything extra by misbehaving since none of the intermediate data is ever in the clear and their inputs and outputs are public, making verification possible.

4.5.2 Correlation Attack with Auxiliary Information

Data Collector traffic information may not reveal anything on its own, but there is a danger that an attacker could fruitfully combine it with auxiliary information, such as observations of a target user's, or indeed of many users', network traffic.

For example, if we did not add noise, but simply released accurate counts only if they were in excess of some threshold, then an adversary could generate its own network traffic to push the counts above the threshold, and then subtract its own traffic (for which it knows the true counts) to yield accurate counts of potentially a single user's traffic.

The differential privacy mechanism proposed adequately addresses this threat. It ensures that, for any adversary, the response of *PrivEx* if the target user *did* visit a target website in a given epoch will be hard to distinguish from the response if the user *did not*.

We also note that since there is only one question *PrivEx* answers (How many visits were made via the ACN to each of this list of websites in this epoch?), and it can be asked only once per epoch, differential privacy's notion of a "privacy budget" is not required in our setting.

4.5.3 Security Proof for *PrivEx*-D2 Variant

We now show that the *PrivEx*-D2 scheme from Section 4.4.2 (using group $\mathbb{G}$ of order q with generators g and h) is secure if ElGamal encryption (using the same group $\mathbb{G}$ with generator

g) is IND-CCA1. The latter fact is known to be true under reasonable assumptions [Lip10], which establishes the security of *PrivEx*-D2.

The security property we seek is this: even if some of the DCs and all but one of the TKSs are adversarial, the adversary will (for each website under consideration) learn no information about the counts of the individual honest DCs, save for their sum.

We do this with a typical real-or-random game. Because the protocol uses non-interactive zero-knowledge proofs based on the Fiat-Shamir heuristic, the proof is in the random oracle model.

We denote the number of DCs by N, of which n are honest, and the number of TKSs by M, of which only number 1 is honest. The adversary game $\mathcal{G}_0$ against the *PrivEx*-D2 protocol proceeds as follows:

Setup phase. S1: The adversary receives the honest TKS's public key A_1 from the challenger, along with a non-interactive zero-knowledge proof of knowledge (NIZKPK) of the corresponding private key a_1 such that $A_1 = g^{a_1}$. S2: The adversary then outputs the adversarial TKSs' public keys $A_2, \ldots, A_M$, along with the corresponding NIZKPKs of a_j such that $A_j = g^{a_j}$ for $j = 2, \ldots, M$.

Counting phase. C1: The adversary supplies one plaintext p_i for each honest DC ($i = 1, \ldots, n$). C2: The challenger chooses a uniformly random bit b. If $b = 0$, the challenger sets $p'_i = p_i$ for each i. If $b = 1$, the challenger picks uniformly random $p'_i \in_R \mathbb{Z}_q$ under the single constraint that $\sum_i p'_i = \sum_i p_i$.

Aggregation phase. A1: The challenger computes its ciphertexts $\langle (g^{r_i}, A^{r_i} \cdot h^{p'_i}) \rangle_{i=1}^{n}$, where $A = \prod_j A_j$ and each r_i is uniform random from $\mathbb{Z}_q$. The challenger sends commitments to these ciphertexts to the adversary. A2: The adversary selects the ciphertexts for the adversarial DCs arbitrarily, and multiplies them to yield the single ciphertext (x, y). It outputs a commitment to (x, y) to the challenger. A3: The challenger opens its commitments by sending $\langle (g^{r_i}, A^{r_i} \cdot h^{p'_i}) \rangle_{i=1}^{n}$ to the adversary. A4: The adversary opens its commitment by sending (x, y) to the challenger. A5: The challenger computes the product α of the first components of all the openings as $\alpha = x \cdot \prod_i g^{r_i}$, and returns α^{a_1} to the adversary, along with the NIZKPK of equality of discrete logarithms that $\log_g A_1 = \log_\alpha \alpha^{a_1}$. Note that $\alpha^{a_1} = x^{a_1} \cdot A_1^{\sum_i r_i}$.

Output phase. The adversary now ouputs its guess b' for the value of b. That is, it tries to decide whether the ciphertexts output in step A3 corresponded to the plaintexts supplied in step C1, or to random plaintexts with the same sum. The advantage of the adversary is $\left| Pr[b' = b] - \frac{1}{2} \right|$.

We now construct game $\mathcal{G}_1$ such that the advantage of the adversary in winning game $\mathcal{G}_1$ is the same as that of it winning game $\mathcal{G}_0$ in the random oracle model. To do this,

we observe that the challenger can program the random oracle to forge any NIZKPK it creates, and can use the NIZKPK extractor to learn the adversary's private values for any NIZKPKs created by the adversary. Therefore, in game $\mathcal{G}_1$, we simply remove the challenger's NIZKPKs from steps S1 and A5, and change step S2 so that the adversary outputs the *private* keys $a_2, \ldots, a_M$. In addition, the binding and hiding properties of the commitment mean that the adversary has to compute its (x, y) before seeing the challenger's ciphertexts. Therefore, we can rearrange the steps of the Aggregation phase so that we remove the commitment steps A1 and A2, and swap the order of A3 and A4.

Now suppose an adversary $\mathcal{A}$ has non-negligible advantage in game $\mathcal{G}_1$. We next, using $\mathcal{A}$ as a black box, construct an adversary $\mathcal{B}$ for the IND-CCA1 game for ElGamal that has the same advantage. The IND-CCA1 game for ElGamal is as follows.

E1: The challenger $\mathcal{E}$ chooses a private key e uniformly at random from $\mathbb{Z}_q$, and outputs the public key $E = g^e$. E2: The adversary $\mathcal{B}$ constructs some (polynomial) number of ciphertexts (α_i, β_i) and sends them to $\mathcal{E}$. E3: $\mathcal{E}$ decrypts the ciphertexts and returns the plaintexts β_i/α_i^e to $\mathcal{B}$. E4: $\mathcal{B}$ chooses two plaintexts $m_0, m_1 \in \mathbb{G}$ and sends them to $\mathcal{E}$. E5: $\mathcal{E}$ chooses a bit b_e uniformly at random, and sends an encryption $(g^r, E^r \cdot m_{b_e})$ of m_{b_e} to $\mathcal{B}$, where $r \in_R \mathbb{Z}_q$. E6: $\mathcal{B}$ outputs its guess b'_e for the value of b_e. The advantage of $\mathcal{B}$ is $\left|Pr[b'_e = b_e] - \frac{1}{2}\right|$.

Here is how $\mathcal{B}$, acting as the challenger to adversary $\mathcal{A}$ for game $\mathcal{G}_1$, can win the above IND-CCA1 game. This interaction is depicted in Figure 4.6. In step E1, $\mathcal{E}$ sends its public key E to $\mathcal{B}$. $\mathcal{B}$ sends $A_1 = E$ to $\mathcal{A}$ in step S1. In step S2, $\mathcal{A}$ outputs $a_2, \ldots, a_M$ to $\mathcal{B}$. Let $\hat{a} = \sum_{j=2}^{M} a_j$, and let $A = \prod_{j=1}^{M} A_j = E \cdot g^{\hat{a}}$.

Now in step C1, $\mathcal{A}$ supplies $p_1, \ldots, p_n$ and in step A3, $\mathcal{A}$ supplies (x, y), both to $\mathcal{B}$. Now $\mathcal{B}$ turns back to $\mathcal{E}$ and submits $(x^{-1}, 1)$ as a ciphertext in step E2; $\mathcal{E}$ will compute $1/x^{-e} = x^e$ and return it to $\mathcal{B}$ in step E3. $\mathcal{B}$ now sets $m_0 = 1$, picks $\lambda \in_R \mathbb{Z}_q^*$, sets $m_1 = h^\lambda$, and submits (m_0, m_1) to $\mathcal{E}$ in step E4. In step E5, $\mathcal{E}$ returns $(R, S) = (g^r, E^r \cdot m_{b_e})$ to $\mathcal{B}$.

$\mathcal{B}$ now needs to compute its ciphertexts $\langle (g^{r_i}, A^{r_i} \cdot h^{p'_i}) \rangle_{i=1}^n$ to send to $\mathcal{A}$ in step A4, such that the p'_i equal the p_i if $b_e = 0$, and the p'_i are random, but with the same sum as the p_i, if $b_e = 1$. (That is, $\mathcal{B}$ implicitly sets $b = b_e$.) To do this, $\mathcal{B}$ picks $s_i, \ldots, s_n$ uniformly at random from $\mathbb{Z}_q$, and picks $\mu_1, \ldots, \mu_n$ uniformly at random from $\mathbb{Z}_q$ subject to the condition that $\sum \mu_i = 0$.

Now let $(R_i, S_i) = (g^{s_i} \cdot R^{\mu_i}, A^{s_i} \cdot R^{\mu_i \cdot \hat{a}} \cdot S^{\mu_i} \cdot h^{p_i})$. If $b_e = 0$, so that $m_{b_e} = 1$, we have that $(R_i, S_i) = (g^{s_i + r\mu_i}, A^{s_i} \cdot g^{r\mu_i \hat{a}} \cdot E^{r\mu_i} \cdot h^{p_i}) = (g^{s_i + r\mu_i}, A^{s_i + r\mu_i} \cdot h^{p_i})$, as required. On the other hand, if $b_e = 1$, so that $m_{b_e} = h^\lambda$, we have that $(R_i, S_i) = (g^{s_i + r\mu_i}, A^{s_i} \cdot g^{r\mu_i \hat{a}} \cdot E^{r\mu_i} \cdot h^{p_i + \lambda\mu_i}) = (g^{s_i + r\mu_i}, A^{s_i + r\mu_i} \cdot h^{p_i + \lambda\mu_i})$. Since $\sum_i \mu_i = 0$, the $p_i + \lambda\mu_i$ are random values in $\mathbb{Z}_q$ summing to $\sum_i p_i$, again as required. $\mathcal{B}$ then sends $\langle (R_i, S_i) \rangle_{i=1}^n$ to $\mathcal{A}$ in step A4.

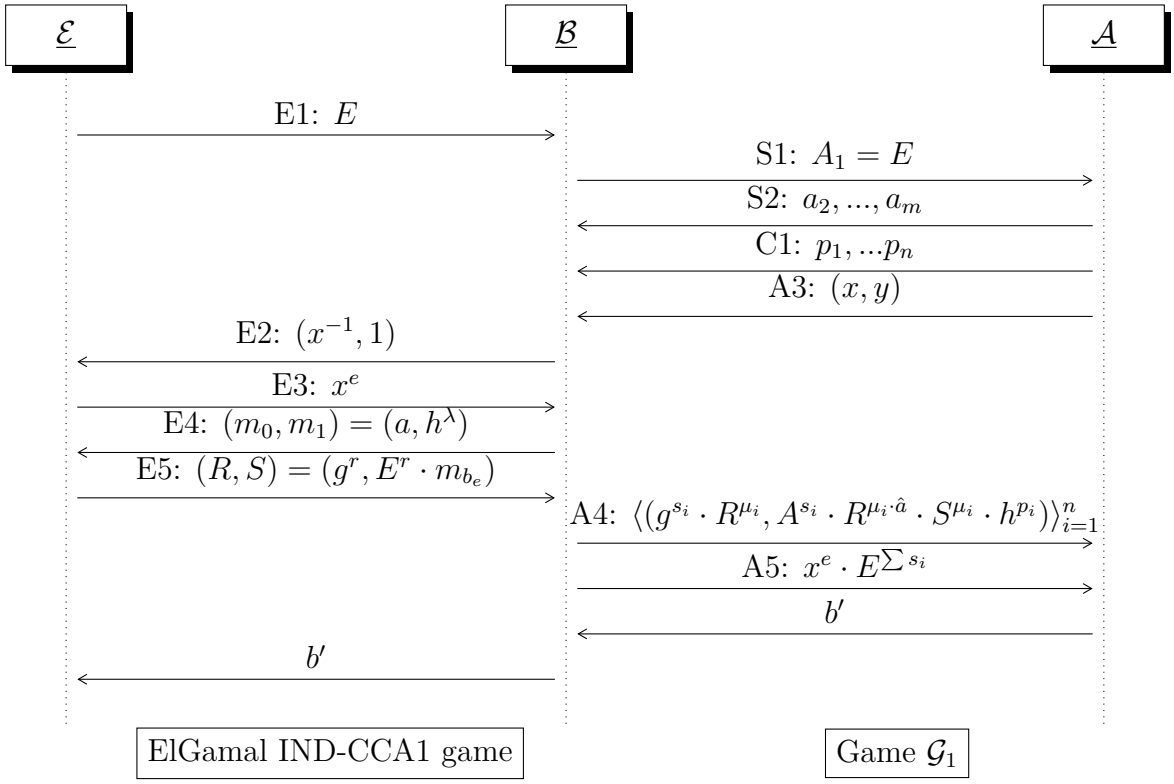


Figure 4.6: $\mathcal{B}$ using adversary A for game $\mathcal{G}_1$ to win the IND-CCA1 game for ElGamal against $\mathcal{E}$.

$\mathcal{B}$'s last move is then to send $(x \cdot \prod R_i)^e$ to $\mathcal{A}$ in step A5. It can easily compute this, as it retrieved x^e from $\mathcal{E}$ in step E3, and $(\prod R_i)^e = E^{\sum s_i + r\mu_i} = E^{\sum s_i}$ since $\sum \mu_i = 0$.

Finally, $\mathcal{A}$ guesses the value of b, and since $b = b_e$, $\mathcal{B}$ simply passes that guess on to $\mathcal{E}$ as its own guess for b_e, winning the IND-CCA1 game for ElGamal if and only if $\mathcal{A}$ wins game $\mathcal{G}_1$.

4.6 Implementation

We have built proof-of-concept implementations for both variants of *PrivEx*. They are implemented in Python using the Twisted library for asynchronous networking between the parties of the system.

Each *PrivEx* scheme uses a few hundred lines of Python code. The code is available for download from our *PrivEx* website.[6] Both schemes use TLS 1.2 with ECDHE for communication between endpoints to ensure that the key material remains confidential during transit and benefits from perfect forward secrecy. We set up long-lived TLS connections when *PrivEx* first comes online; their communication and computational costs are amortized over many epochs.

We have not implemented the Country of Origin feature at this time since we would like to see *PrivEx* deployed in the Tor network with the core feature set before expanding on it. The core implementation above is ACN agnostic, and we aim to integrate it with Tor in the near future.

In the tables in this section, the "Per node" column is calculated by taking the total cost for each type of *PrivEx* node and dividing it by the count of that type of node, to find the cost at each type of node. This helps identify potential bottlenecks.

4.6.1 Computational Overhead

We present *PrivEx* overhead statistics to show that both schemes have low computation requirements. The hardware for our experiments is a 3 GHz quad-core AMD desktop computer with 4 GiB of RAM running stock Ubuntu 14.04.

Using a test harness we measure the time the core components of each *PrivEx* scheme take under parameters typically found in the Tor network. We simulate a network of 10 TKSs and 1000 DCs; the latter reflects the number of exits in the current Tor network. The number of censored websites to collect statistics for is 1000 and each website is "visited" one million times. No actual website connections are made by the DCs since we are interested in capturing the overhead of the *PrivEx* schemes.

***PrivEx*-S2.** From Table 4.1, we note that the setup phase of *PrivEx*-S2 takes 4.1 s on average and that the tally phase takes 470 ms on average (adding the "per node" times, as the nodes act in parallel). Without any ACN traffic (*i.e.* no DC increment operations), the total overhead wall-clock time per epoch is 4.6 s. The key figure to note is that the addition operations at the DC nodes take less then 1 μs each (900 μs for 1000 visits per DC) on average. This low cost is important, as this operation will be called the most often and the impact on DC nodes must be negligible so that they can service ACN requests without undue overhead.

[6] https://crysp.uwaterloo.ca/software/

Table 4.1: The overhead per epoch (with 95% confidence intervals) incurred by participants in the *PrivEx*-S2 scheme for 10 TKSs and 1000 DCs with 1000 websites with one million visits per epoch.

Operation	Total per epoch (ms)	Per node (ms)
TKS initialize	0.012 ± 0.004	0.0012 ± 0.0004
TKS register	41000 ± 3000	4100 ± 300
DC initialize	40000 ± 3000	40 ± 3
DC register	312 ± 8	0.312 ± 0.008
DC increment	900 ± 90	0.90 ± 0.09
DC publish	1.7 ± 0.1	0.0017 ± 0.0001
TKS publish	0.56 ± 0.06	0.056 ± 0.006
TS sum	470 ± 20	470 ± 20
Epoch Total	83000 ± 6000	—

***PrivEx*-D2.** From Table 4.2, we note that the setup phase of *PrivEx*-D2 takes 297 ms on average with the DC nodes bearing the most cost. The entire tally phase takes 1.69 m on average per epoch (adding the "per node" numbers, as these operations occur in parallel). Combining the overhead for both phases, the epoch overhead wall-clock time is 1.7 m on average. We see in *PrivEx*-D2 that the addition operation takes 3.9 μs on average and again, like *PrivEx*-S2 above, this is desirable since it is the most frequent operation.

Discussion. *PrivEx*-S2 has lower computational cost than *PrivEx*-D2, by a factor of almost 10 in our example. Yet, it is clear from these results that the computational overhead at each type of node in *PrivEx* is low and that the time requirements are a small fraction of the duration of an epoch. Indeed, even if there are applications where statistics need to be gathered for shorter epochs, *PrivEx* can still be useful; as we saw earlier, for each setup-tally cycle the *PrivEx*-S2 scheme incurs less than 4.6 s of overhead while the *PrivEx*-D2 scheme incurs less than 1.7 m of overhead, which is very small compared to the typical example epoch length of one hour. This means that the statistics collection frequency can be as low as 5 s and 2 m respectively. This flexibility allows one to match the appropriate *PrivEx* scheme to the application's statistics frequency and threat model requirements.

Table 4.2: The overhead per epoch (with 95% confidence interval) incurred by participants in the *PrivEx*-D2 scheme for 10 TKSs and 1000 DCs with 1000 websites with one million visits per epoch.

Operation	Total per epoch (ms)	Per node (ms)
TKS initialize	10.9 ± 0.2	1.09 ± 0.02
DC combine key	4.05 ± 0.02	0.00405 ± 0.00002
DC initialize	295000 ± 600	295 ± 0.6
DC increment	3.9 ± 0.1	0.0039 ± 0.0001
PBB productize	50400 ± 400	50400 ± 400
TKS decrypt	448000 ± 3000	44800 ± 300
PBB DL Lookup	6293 ± 40	6290 ± 40
Epoch Total	800000 ± 3000	—

4.6.2 Communication Overhead

We now give a closed-form analysis of the communication costs of the two *PrivEx* schemes. In the following description, DC_N, TKS_N, and W_N represent the number of DC nodes, TKS nodes, and websites for which we are collecting statistics, respectively.

An overhead in common for both schemes is the list of websites and the constants for DDP calculations σ, ϕ, and γ. We make the conservative assumption that the website domain name in the URL will not be more than 255 characters long, therefore the maximum length of the URL list is $255 \cdot W_N$ bytes. The constants require 8 bytes in total. In the experimental setting above this overhead is $\sim$249 KiB, the overwhelming majority of it being the website list. While it is not as significant, we note that the website lists and values for the constants need not be transmitted every epoch, instead only being sent when there is a drastic change in the network conditions or the website lists are updated.

***PrivEx*-S2.** In the setup phase, each DC sends 16 bytes of key material to each TKS for a total of $16DC_N \cdot TKS_N$ bytes.

In the tally phase, each DC sends 4 bytes to the TS for each website in the database for a total of $4W_N \cdot DC_N$ bytes. Similarly, each TKS also sends the same amount to the TS for each website for a total of $4W_N \cdot TKS_N$ bytes.

In each epoch, the total communication cost, in bytes, is

$$16DC_N \cdot TKS_N + 4W_N(DC_N + TKS_N)$$

Table 4.3: Communication overhead (in KiB) of *PrivEx*-S2 for 1000 websites, 10 TKSs and 1000 DCs per epoch, using closed-form analysis. Note that we charge the data transfer to the sender so nodes, *e.g.* TS, that only receive data show no communication overhead.

	Setup	Tally	Total	Per node
DC	156.25	3906.25	4062.50	4.06
TKS	0	39.07	39.07	3.91
TS	0	0	0	0
Total	156.25	3945.32	4101.56	—

For 10 TKSs and 1000 DCs tracking 1000 websites we see from Table 4.3 that the total communication cost for every epoch is ∼4 MiB, but the cost for each type of node is far lower at only ∼4 KiB.

***PrivEx*-D2.** In the setup phase, each TKS sends 96 bytes of key material and zero-knowledge proof to the PBB for a total of $96TKS_N$ bytes. Then, each DC retrieves the key material and the proofs from the PBB for a total of $96TKS_N \cdot DC_N$ bytes.

In the tally phase, each DC sends a 32-byte commitment to the PBB for a total of $32DC_N$ bytes. After all DCs have sent their commitments, the PBB sends each DC the commitments of the other DCs for a total of $32DC_N^2$ bytes. Then, each DC sends a 64-byte opening of the commitment for each website to the PBB for a total of $64W_N \cdot DC_N$ bytes. The PBB then sends, in parallel, the opening of the DC's commitments to each TKS for a total of $TKS_N(DC_N(64W_N + 32))$ bytes. In response each TKS sends the results of the partial decryption for each website in the database, along with a zero-knowledge proof of equality of discrete logs for a total of $TKS_N(32W_N + 64)$ bytes.

In each epoch, the total communication cost, in bytes, is

$$32\,(W_N(2DC_N \cdot TKS_N + 2DC_N + TKS_N) + DC_N^2 + 4DC_N \cdot TKS_N + 5TKS_N + DC_N)$$

From Table 4.4 we see that, in our experimental setting, the total communication cost for each epoch is ∼703 MiB, while each of the TKS and DC nodes send only ∼32 KiB and ∼63 KiB respectively. The bulk of the communication cost is borne by the PBB node.

Discussion. Both schemes scale linearly with the number of websites and TKSs. *PrivEx*-D2 scales quadratically with the number of DCs while *PrivEx*-S2 remains linear. While it

Table 4.4: Communication overhead of *PrivEx*-D2 for 1000 websites, 10 TKSs and 1000 DCs per epoch, using closed-form analysis. Note the units in the column headings. Note that we charge the data transfer to the sender so nodes, *e.g.* DCs, that only receive data show no communication overhead.

	Setup (KiB)	Tally (MiB)	Total (MiB)	Per node (KiB)
DC	0	61.07	61.07	62.54
TKS	0.94	0.31	0.31	31.74
PBB	937.5	641.17	642.09	657500
Total	938.44	702.55	703.47	—

is true that the *PrivEx*-D2 scheme is generally more expensive, we note that each DC and TKS transmits only tens of KiB of traffic per epoch, which is comparable to *PrivEx*-S2. However, the PBB transmits hundreds of mebibytes due to the higher security and privacy guarantees it allows. To mitigate the impact of this load, it is expected that the PBB will be well resourced for this task. Indeed, we expect that in real deployments the number of TKSs would be closer to three and the number of websites would be closer to 100. In that scenario, the total communication cost would be approximately 55 MiB per epoch.

The *PrivEx*-S2 scheme is relatively lightweight, enjoying very low overhead and perhaps a better choice in low-bandwidth environments or where the size of the website list will be very large.

Even so, in absolute terms, both *PrivEx* schemes have low overhead for DC and TKS nodes. We note that in the Tor network, even relays in the 1^{st} percentile by bandwidth (18.4 KBps)—which are also the least likely to be chosen in circuits in any event—can manage the load easily. [Tor10a]

From the perspective of the DC, which is also a node in the ACN, *PrivEx* does not significantly impact bandwidth usage which can be better used to service ACN traffic. From the perspective of the TKS, TS, and PBB, even though we expect that the servers would be well provisioned for the task of aggregating statistics, the resource requirements are low enough that they would also not be significantly impacted by participating in *PrivEx*.

4.7 Real-World Deployment

Having designed and implemented *PrivEx*, we now use it to learn about the nature of censorship traffic on the Internet. We are particularly interested in seeing the breakdown of censorship traffic as it compares to non-censorship traffic on a particular network.

Methodology. For our study we target the Tor network for two reasons. First, a privacy-preserving study of this nature has not been conducted and would yield useful insights about Tor user behavior, specifically how much traffic is censorship resistance related. Second, it would provide a proof-of-concept validation to the community that privacy-preserving and utility-preserving data collection is practical and spur *PrivEx* uptake and further research of these types of systems.

The first reason above may seem counterintuitive since we have classified Tor as a CRS, and hence all traffic on the network should be considered censorship resistance related. From an abstract and global Internet perspective this observation is certainly true. However, the Tor network is an ecosystem serving many purposes, including but not limited to censorship resistance. For a censor to block Tor—where such a block would have potential collateral damage due to the defensive strategies we have discussed in Chapter 2—the cost of information leakage due to the CRS activity on the network must be higher than the cost of the collateral damage it would suffer. Recall from Chapter 3 that knowing the base rate of CRS activity, or BR, helps fill in information that would help evaluate if a block is economically responsible. Indeed, this analysis can tell Tor designers if more collateral damage needs to be leveraged to tip the balance to prevent such a block. Hence, Tor is an appropriate candidate for this type of study.

We utilize DNS requests as a means of learning about how often Tor users are interested in CRS-related websites. We are aware that a DNS request does not necessarily translate to an actual visit; *e.g.*, web pages that track users through third-party advertising networks will cause DNS queries to third-party domains but the user never actually "visits" those domains. This is acceptable for our study since we assume that a domain appearing on the censor's blacklist is due to it being a target of censorship.

We compiled a list of nearly 6100 censored websites by scraping the GreatFire.org website that tracks Chinese censorship by running connectivity tests from behind the national firewall to websites on the Internet[7] and a leaked list of websites blocked in Germany.[8]

[7] https://en.greatfire.org/

[8] https://bpjmleak.neocities.org—see archived version at https://web.archive.org/web/20140707204711/https://bpjmleak.neocities.org/ for the list that has since been removed due to pressure from the German authorities.

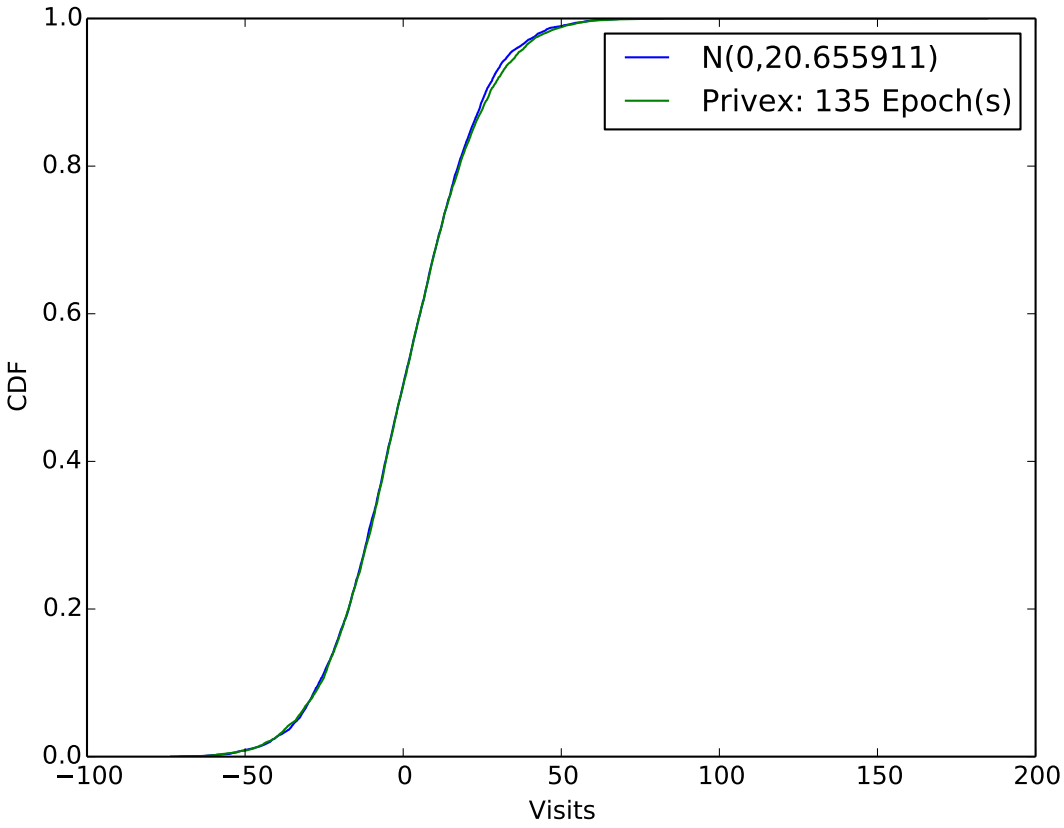


Figure 4.7: A CDF of the aggregated statistics collected by *PrivEx* as compared to the Gaussian noise added. A follow-up Kolmogorov-Smirnov test confirms that actual visits were registered in the statistics collected with *PrivEx*.

Apparatus. We utilized the *PrivEx*-S2 variant for two main reasons. First, neither CRS-client nor CRS-server software would have to be modified. Second, the low operational resource requirements lowered the bar for entry and were helpful in recruiting volunteers and resources.

Our initial proof-of-concept deployment consists of two TKSs, a TS, and a DC. The TKSs are operated by third parties, one using a virtual hosting provider on the Isle of Man and the other through the university network provider at the KU Leuven in Belgium. The DC is co-hosted with the Tor exit node, nicknamed gurgle, operating at the University of Waterloo in Canada. The TS is hosted on another machine at the same institution.

For the duration of the data collection reported here, gurgle had a probability of 0.15% of being selected as the egress node from the Tor network. This means that we expect to see this proportion of all traffic exiting the Tor network.

Collected Results. We set the epoch to one hour and collected statistics for 135 epochs, which is more than the 126 epochs required for the level of privacy and utility from our worked example in Section 4.4.4.

We validate that our implementation produces results with the same characteristics as our analysis indicates and that they are reliable. We produce a CDF (Figure 4.7) of the aggregated statistics and plot it against a CDF of the Gaussian noise function we utilized with the standard deviation set to $\frac{\sigma}{\sqrt{epochs}} = \frac{240}{\sqrt{135}} = 20.655911$, where σ is the same as in the worked example above. We then ran a Kolmogorov-Smirnov test to ensure that the distance between the two plots was positive and large enough to indicate that they were drawn from two different distributions. The test showed a distance of 0.016 with likelihood of between 0.05 and 0.10 that the observed difference is due to randomness. This indicates that there were actual visits to the domains in our list and that we applied the expected level of noise.

The results show that the average number of hits from the censored list is the range 586–686 and those for off-list DNS requests is the range 31810–31910. This is a likelihood in the range of 1.8–2.2% that a given DNS request coming to gurgle is for a site in our compiled list, with probability exceeding 99%. This result provides an idea of the magnitude of the answer to the question of how CRS-related traffic compares to the rest of the network.

From this measure we draw a conclusion that since the base rate is so low, the accuracy of the censorship apparatus must be of a higher magnitude in order to avoid a large amount of false positives, *i.e.* collateral damage. For example, using the higher base rate of 2.2% above, a censorship apparatus that provides a 100% true positive rate (*i.e.*, no information leaks) and a 1% false positive rate (*i.e.*, collateral damage) would only be correct 69% of

the time when it claimed that a some event was CRS related. The rest is collateral damage, in stark contrast to the ostensible 1% rate stated above. To compensate for the additional error and achieve the original 1% figure the apparatus would need to have a false postive rate of 0.01%. If a censor wants to ensure negligible collateral damage in this low base rate setting their apparatus must have a false positive rate of 0.001% which may be very difficult to achieve.

4.8 Related Work

Differential Privacy. While *PrivEx* utilizes differential privacy (DP), there are many key differences in the setting in which it is traditionally applied and the *PrivEx* setting.

In classical DP there is a trusted centralized database—usually a third-party host—who can see the real data and is considered secure. Instead, in *PrivEx* the data is distributed across nodes in the network where no entity has access to all of the real data from all of the nodes. The only data that is revealed to anyone is the aggregated statistics with noise added. An adversary would have to compromise a large fraction of the DCs, or *all* of the TKSs, in order to access the private data of the honest parties.

In the usual DP setting the database is static across epochs and clients use up their *privacy budget* to make a number of database queries—the results of which are usually private unless they choose to make them public. As discussed at the end of Section 4.5.2, in *PrivEx*, the database is completely refreshed at the start of every epoch and only a single constant query is ever made every epoch, the result of which is then made public.

A number of works consider the problem of securely computing functions in a distributed differential privacy setting.

Dwork *et al.* [DKM+06] provide a method for generating shares of random Gaussian noise in a multiparty setting mirroring the distribution of noise in our setting. The key difference is that the parties work together to first produce noise shares which are then used to perturb the data in their individual databases whereas in *PrivEx* the noise is calculated independently using network state and does not incur extra protocol rounds. Also, they assume that $\frac{2}{3}$ of the participants will be honest while *PrivEx* makes no such explicit restriction; *i.e.*, a lone honest DC may enjoy the same level of privacy as the designer intended, albeit with longer aggregation periods to gain the same level of utility as designed.

In the two-party setting of distributed differential privacy, Goyal *et al.* [GMPS13] explicitly evaluate the accuracy-privacy tradeoffs for computing Boolean functions. Mironov

et al. [MPRV09] investigate calculating the distance between two vectors while McGregor *et al.* [MMP+10] do the same for Hamming distance. All these works explore the limits of DDP in the two-party setting. We contrast our work by noting that we consider a different type of problem (the summation of integral inputs) and we evaluate the tradeoff between the accuracy and privacy in the multiparty setting.

The closest related work is by Beimel *et al.* [BNO08]. The inputs in that setting are binary, while those in ours are integral. While the binary inputs can indeed be adapted to integers, there remain three key differences. Their protocol requires more rounds of communication than ours, while we also allow for malicious parties, making *PrivEx* a more practical solution in our setting. Finally, in their setting, to preserve DP, the database of each DC is kept private and only binary outputs are released, whereas in our setting all DCs release their private data, albeit with noise added to preserve DP.

Also of interest is work by Kasiviswanathan *et al.* [KNRS13] where network graphs are analyzed to investigate how the removal and addition of nodes in the graph affect the privacy of the information about the structure of the graph. While they also consider differential privacy in the network setting, the key difference is that they investigate ways to safely reveal information about the nodes of the network themselves, whereas we are interested in the information that can be revealed by studying the traffic flowing *through* the network; *i.e.*, the network users' information.

A general key difference to the previous literature is that *PrivEx* provides a way to reason about the privacy and utility that the system provides whereas these previous works leave it up to the system designer to work out. We provide an explicit statement of, and relationship between, privacy and utility that are pertinent to data collection in ACNs—this provides an easier-to-analyze system and potentially an easier path to deployment.

Secure Multiparty Computation. Secure multiparty computations have been used in scenarios where the parties that perform the operations are not trustworthy. This means that they should not learn the inputs of the calculations, should provide (implicit or explicit) proofs that the calculations were performed correctly, and should not learn anything more than the output of the calculation.

A closely related work is SEPIA [BSMD10] by Burkhart *et al.* where networks collect data and wish to learn aggregate information about their networks without revealing their individual inputs. It develops a number of operations that can be performed on network data that can be evaluated by a pool of servers in a secure multiparty computation. While both *PrivEx* and SEPIA try to achieve similar goals in the collection of network statistics and use similar secret sharing schemes, there are a number of differences. First, while

the authors of SEPIA briefly mention differential privacy as a possible defence, *PrivEx* provides a thorough treatment of how to use differential privacy to protect the aggregated statistics in a principled manner. Related to that is that SEPIA also requires that honest DCs sanitize their inputs, *i.e.* remove sensitive information, whereas *PrivEx* accomplishes the same with the addition of DP noise. Second, *PrivEx* is secure as long as there is one honest data collector—adding the appropriate level of noise, as outlined in Section 4.4.4—and one honest TKS. This is in contrast to the SEPIA requirement that at least half of the aggregators be honest. This is especially useful since *PrivEx* collects data from an anonymity network where the stakes for information leakage are potentially higher and hence require greater robustness to bad actors. Finally, we note that the data collectors in SEPIA are provisioned for processing large quantities of traffic and data as they are part of the ISP infrastructure, but these conditions may not apply in a volunteer-resourced network like Tor. *PrivEx* has low overhead for the DCs.

The secret sharing scheme is based on the scheme presented by Barthe *et al.* [BDG+13] which itself is an extension of previous works by Kursawe *et al.* [KDK11], Jawurek *et al.* [JK12] and Shi *et al.* [SCR+11]. The novelty of *PrivEx* is that it introduces addition using additive secret shares for coercion resistance and perfect forward secrecy, which these previous works do not address.

Anonymity Network Data Collection. The work by McCoy *et al.* [MBG+08] provided many insights about Tor client behavior. Unfortunately, the method of safeguarding the privacy of the collected data was considered by the community at large to be insufficient. [Sog11] Similarly, Diaz and Sassaman [DSD04] provided insights about mix input traffic in Mix-stlye anonymous email networks by using actual traffic obtained from a public node. Here too, the use of actual traffic data had the potential to deanonymize clients. *PrivEx* ameliorates this state of affairs by providing researchers the means to collect statistical data about clients of anonymous networks in a privacy-preserving and compulsion-resistant manner.

Anonymity networks have to be careful about how they collect data about their network and users since they are in a position of power and can potentially expose the entire network. The operators of Tor also collect client-specific network usage data from their guard and bridge nodes but not the exit nodes. The reason why it is considered safer to do the former and not the latter—in the context of protecting client anonymity—is that the guards/bridges already know who the clients that connect through them are so an adversary who compromises those nodes would not learn any extra information.

A key difference between *PrivEx* and the present Tor data collection environment is that in that latter, the true client statistics (aggregated at a per-country level, for example)

are stored in a centralized database. *PrivEx* does not allow any entity to learn any real client data except the nodes that originally collected the data.

4.9 Future Work

As a potential additional application of *PrivEx*, we note that while the Tor network does not typically try to hide the fact that a client is using Tor, there may be risks to revealing statistics gathered through widespread ingress data collection similar to those addressed by *PrivEx* of egress data collection. To address these potential risks, *PrivEx* can be applied to the present guard/bridge data collection process, and provide the same benefits as those that have been shown here for exit nodes.

An open question is whether *PrivEx*-like systems can be extended to collect data across subsets of the network. The risks are that this will give the adversary the ability to partition the data and perhaps learn something from the statistics that he should not have. If this can be done safely, one direct benefit is that we could, in a privacy-preserving manner, troubleshoot specific issues that are localized.

A limitation of *PrivEx*, since it is not needed for the scenarios we study, is that only a single query can be made of the database. We would like to investigate how to support multiple related queries—*e.g.*, network load or circuit latency—while maintaining *PrivEx*'s privacy and utility features.

4.10 Conclusion

We have presented *PrivEx*, a decentralized system for privately collecting client statistics in anonymity networks. We have detailed two variants of *PrivEx*, one based on secret sharing and the other on distributed decryption. Both schemes are efficient and resilient to coercion attacks and malicious actors. We introduce noise, as defined in the differential privacy setting, into our aggregation process to prevent information leakage that would otherwise occur when the statistics are published.

We have used Tor as a case study and show how it can incorporate *PrivEx*; other anonymity networks can similarly deploy *PrivEx*. In this case study we collect statistics about client destination visits at the DC nodes. We show that this can be done in an efficient manner with low computational and communication overhead for conditions typical in the Tor network.

The statistics we have gathered from our small-scale deployment tell us that the incidence of CRS traffic on the Tor network is close to 2% of the total website visits, which is relatively low. This suggests that the censor would need to be very tolerant to collateral damage or otherwise field a very accurate classifier leveraging high-quality distinguishers, or block the network activity entirely. Due to the scale of the deployment we cannot make specific conclusions about clients' behaviors or their usage trends. With better and more extensive website lists coupled with the deployment of *PrivEx*-D2 for country-specific statistics we may be able to fill in the details of the broad-strokes picture that we can currently see.

Finally, with *PrivEx*, our aim is to convince administrators and users of anonymity networks that client data collection *is* possible while maintaining anonymity and privacy. The benefits are that valuable information about usage trends will help guide performance and maintenance efforts. From the research perspective the benefits will be more accurate usage statistics, client models, and clearer indicators of future directions that anonymous communications and censorship resistance research should take.

Chapter 5

An Analysis of Path Selection Security in Tor

Portions of this chapter were previously published in the proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society [EBA+12].

5.1 Introduction

The security properties of CRSs are dictated in large part by implementation details. In this chapter we investigate a concrete example of a CRS, Tor [DMS04b], that mitigates against CRS-client linking and publisher anonymity attacks, which we discussed in Section 2.4. The CRS strategies being invoked are those of obfuscated values to protect paths and destinations, and rate limiting the visibility of client communications from censor-controlled nodes.

Tor is the most widely used volunteer-resourced anonymous communication network. It is designed to provide communicating parties with anonymity from their communication partners as well as unlinkability from passive third parties observing the network. This is done by distributing trust over a series of Tor routers, which the network clients select to build paths to their Internet destinations.

If the adversary can anticipate or compel clients to choose compromised routers then CRS clients can lose their anonymity. Indeed, the client router selection protocol needs to be secure against adversarial manipulation and leak no information about clients' selected routers. It is a key ingredient in maintaining the privacy properties that Tor provides.

When the Tor network was first launched in 2003, clients selected routers uniformly at random—an ideal scheme that provides the highest amount of path entropy and thus the least amount of information to the adversary. However, for load balancing reasons, the router selection algorithm was changed in 2004 so that clients weight their selection by the amount of bandwidth that routers offer to the network; a router that offers more bandwidth to the network is selected more often by clients.

Another key change to the original router selection algorithm in Tor is the use of *entry guards*. The concept of entry guards emerged as a solution to safeguard against a variety of threats to end-user anonymity [WALS02, ØS06, BDMT07]. Originally, every time a client created a circuit she would pick her first hop from the pool of all Tor routers meaning that the probability of picking a high-bandwidth adversarial router would be high. Entry guards are a restricted set of a few routers, picked by the client upon joining the network, to serve as the first hop for all subsequent circuits. The effect is that whereas before the adversary router could be picked every time a circuit is created, now it only gets a chance when the client creates their entry guard list. More details follow in Section 5.2.1.

Guards were adopted into Tor with specific parameters that seemed likely to provide acceptable security and load balancing characteristics for the network and end users. Those parameters include the number of entry guards that a client begins with, and the amount of time a client can use his/her entry guard before switching (rotating) to new entry guards.

Context and motivation. Since 2011, there has been renewed interest in reevaluating these fixed parameters in combination with network conditions, such as churn and load balancing, to more carefully determine the security that entry guards provide to users.

Dingledine [Din11b] formalized the open issues related to Tor's entry guard design, which are paraphrased below:

- Quantify the vulnerability due to natural guard churn, which is the added compromise due to guard nodes going offline.
- Quantify the client compromise rates at different amounts of adversarial guard bandwidth in the network.
- Quantify the vulnerability due to guard rotation and compare with natural churn. Which of these is the dominant contributor to client compromise? Also, how does varying the rotation periods affect the compromise rates?
- Quantify the client compromise effects of different guard list sizes.

While analysis [ØS06] provides evidence of security benefits and there is a consensus within the Tor community that entry guards provide load balancing benefits, there is yet no empirical evidence of the effects and limitations inherent in their design and in their implementation. Indeed, a lot of faith is placed in the design of guards and it is pragmatic to ensure that this faith is well placed.

Understanding and improving entry guards. To gauge the security and performance impact of entry guards in Tor and to provide direct answers to the questions above, we conduct an empirical analysis of Tor's entry guard selection and rotation algorithms by constructing a simulation framework called *Changing of the Guards* (*COGS*).

Contributions. This chapter offers the following contributions to the field of anonymous communications and censorship resistance:

- We present *COGS*, our simulation framework that is designed to provide quantitative data about guard design choices.
- With *COGS*, we conduct an empirical characterization of entry guards fueled by real data on Tor routers captured by the Tor data-collecting service.[1] In particular, we analyze natural churn, entry guard rotation, the number of entry guards chosen, and other parameters in terms of their effects on security and performance through large-scale simulation of Tor's current entry guard selection and rotation algorithms.
- We investigate the trade-offs between the variables above from the perspectives of security and performance.
- We present answers to open research questions posed by Dingledine with discussion on future guard design research.

Our results indicate that Tor's guard flag allocation process improves overall guard stability and that guard rotation is a major contributor of client compromise yet is self-limiting. We find that reducing the number of guards and increasing the churn period improves client security by providing less compromised guard sets and increasing the time to first compromise. However, we also find that for *certain* client/adversarial models using more guards provides far superior security than possible under Tor's current defaults.

[1] https://collector.torproject.org

5.2 Background

In this section, we present a detailed overview of Tor's design and system architecture.

5.2.1 Tor Overview

We now augment the brief description in Section 4.2 about how Tor works with further details about components of Tor that are relevant to *COGS*.

Recall that a Tor client can remain anonymous from Internet servers, and the parties in communication can remain unlinked from each other from the perspective of an observer. We shall now discuss the mechanics of how this is achieved.

The Tor network is composed of volunteer-operated nodes called *Onion Routers* (ORs), also known as *relays* or *nodes*. These ORs provide network connectivity and bandwidth capacity for end-user traffic. Anyone may operate a relay and indeed a strength of Tor is the diversity and number of its network nodes. When an OR joins the network it announces its details, such as its network address/port, its donated bandwidth capacity, and its *exit policy*—stating to what Internet addresses and ports outside of the Tor network this relay is willing to send traffic—to the (distributed) *directory authority*. The OR will then be listed on the global list of relays and be a candidate for routing end-user traffic.

An end user downloads the Tor client, also known as an *Onion Proxy* (OP), which on start up downloads the *consensus* document listing all running relays as well as relay *descriptors* from the directory authority (or one of its mirrors). These documents contain the details of each relay that the OP can use to route traffic through the network. In order to protect clients against route bridging and fingerprinting attacks [DS08], these documents are updated hourly so as to provide a current and consistent picture of the network to all clients. Consensus documents are published precisely once per hour and descriptors are updated in real time as their contents change.

The directory authority also provides metadata in the consensus document that helps the OP route traffic more intelligently. In particular, the OP uses the consensus in the process of constructing a *circuit*—a path through the Tor network. By default, circuits consist of three ORs selected by the OP. We next describe the process of router selection that is performed by OPs.

Router selection. In the default setting, the OP selects ORs from a distribution that favours higher-bandwidth relays but also allows low-bandwidth relays to be utilized to some

extent. The three ORs in the circuit are termed the *entry*, *middle*, and *exit* ORs. The OP communicates directly with the entry OR, the entry communicates with the middle OR, and the middle communicates with the exit OR. Finally, the exit OR communicates directly with the destination Internet server.

Although the number of circuits constructed is governed by immediate and anticipated need, a general rule is that each circuit is used for ten minutes before the Tor client will begin using a fresh circuit.

The OP constructs the circuit as follows. The OP first picks a suitable exit relay—suitability being a function of the relay's configuration as an exit relay (which is communicated to clients with the `Exit` flag in the consensus document) and its exit policy. Next, the OP picks the entry OR while ensuring that all the relays have distinct /16 IP addresses and relay *families*.[2] (We provide more details on the constraints placed on entry selection in Section 5.2.2.) The middle node is then picked in a similar fashion.

Finally, the OP constructs the circuit using the three ORs in an incremental and telescoping manner. The OP negotiates cryptographic material with the entry OR and once an encrypted channel is established between them it asks the entry OR to *extend* the circuit towards the middle OR. The OP then negotiates cryptographic material with the middle OR—communicating through the entry OR—to establish an encrypted channel between them. The middle OR is then asked to extend the circuit to the exit OR and the process is repeated to establish a secure channel between the OP and the exit relay.

5.2.2 Entry Guard Relays

All Tor relays are donated and as such it is hard to know which ones can be trusted. It is easy, then, for the adversary to donate resources and participate in circuits. The danger is when the adversary controls both the entry and exit ORs on a single circuit. In this scenario the client address and destination address of the traffic are known to the adversary who, through tagging or traffic confirmation attacks [Dan04, MZ07, ES09], effectively deanonymizes the client. Following this previous work, Johnson *et al.* [JWJ+13] show that these attacks are in fact easy to carry out.

Given enough time and the presence of adversarial ORs, the OP will eventually construct circuits that have malicious entry and exit ORs. Since Tor picks relays weighted according to bandwidth, a sufficiently resourceful adversary can deluge the network with high-bandwidth relays and increase the rate at which it can compromise circuits.

[2] Operators of multiple Tor relays can voluntarily mark all the ORs they control as being in a common family.

To mitigate this and related threats such as the predecessor attack [WALS02] and locating hidden services [ØS06], entry guards were introduced. They limit the impact an adversary can have on Tor's user base by effectively reducing the number of times each client selects its entry relays, thus slowing the rate of compromise and limiting the SoI of the adversary.

Instead of picking a new entry every time a circuit is constructed, the OP maintains a *guard list* of a handful of pre-selected entry relays. When the Tor client constructs this list, it selects an expiry time for each of the guards in the list uniformly at random from the range of 30–60 days; after that time, the guards will be dropped and repopulated, as described in detail below. When circuits are constructed, the entry relay to be used is selected uniformly at random from the client's guard list. The rest of the circuit building process remains the same. The effect of this change is that if no malicious guard relays have been picked, the user is uncompromisable by the adversary until she picks new guards. The disadvantage is that if a client does pick a malicious guard then she has a higher probability of being compromised for the next 30–60 days. It is debatable if it is better to a) be compromised with some probability all the time or to b) be either completely safe, or else compromised with higher probability. Øverlier and Syverson [ØS06] provide analysis that the latter is preferable and hence the guard mechanism is embedded in the Tor client code.

Moreover, since entry guards have the potential of negatively affecting the performance of the Tor network and security of its users, they need to be carefully selected. The main mechanisms in place are the directory authority, which assigns guard status to relays, and the guard selection algorithm executed by the Tor client. We next explain how the guard flag is obtained by ORs and how the guard selection algorithm is carried out.

Guard flag. All ORs in the Tor network are monitored for availability and bandwidth capacity by the directory authority. Relays deemed stable[3] and providing bandwidth above a certain threshold (currently the median of all relay bandwidths, or 250 KB/s, whichever is smaller [DM06b]) are selected to receive the *guard flag* in the consensus document; this flag marks a relay as eligible to be included in guard lists. This criterion promotes ORs that will most likely be around for a long time and provide a level of bandwidth that will not likely cause bottlenecks. However, we find that there is large variance in actual guard bandwidth and stability. At the time of our experiments there were, on average, 800 routers with the guard flag. An important tension to note is that if the criteria are too selective, then few guards will be available, forcing more traffic through fewer nodes,

[3]The Guard flag aims for high availability, not to be confused with the Stable flag from the consensus document, which is given to relays with above-median mean-time-between-failures.

Algorithm 1: Tor's approach to retrying unavailable entry guards

Input: Current time T, last attempt at time E_ℓ to contact entry guard E, E has been unreachable since time E_u

Output: Return true if we should try to contact E, false otherwise

```
1 d ← T − E_u
2 if E_ℓ < E_u then return true
3 else if d < hours 6 then return T > (E_ℓ + hour 1)
4 else if d < days 3 then return T > (E_ℓ + hours 4)
5 else if d < days 7 then return T > (E_ℓ + hours 18)
6 else return T > (E_ℓ + hours 36)
```

at a cost to both network utilization and security. At the same time, if the criteria are too lenient, then less stable guards are likely to churn more often, leading to larger guard lists, and an increased likelihood of selecting a malicious guard. This paper investigates this balance in detail.

Guard selection algorithm. Each client ensures that the number of guards—both online and offline—in its guard list is at least the default number at all times. If a guard goes offline, either temporarily or permanently, and there are fewer than two *online* guards in the guard list, a new entry guard is picked, but each previous guard is retried periodically, with an increasing back-off period,[4] according to algorithm 1. In addition, each of the relays in a client's guard list expires in 30–60 days as a *guard rotation event* occurs. The algorithm for picking a guard, in either scenario, is as follows:

- Read the consensus to find the set of relays with the guard flag set.
- Exclude guards already in the client's guard list, if any.
- Exclude guards in the same /16 IP block or family as any of the guards in the client's guard list.
- Select a guard at random from the remaining list of relays, weighted by the relays' *adjusted bandwidths* (see below).
- Assign a random expiration time 30–60 days hence.
- Repeat until the guard list contains the required number of guard relays.

[4]While we do not analyze the effects of changing the back-off periods—currently believed to be orthogonal to Tor's guard design—*COGS* provides us the ability to do so in the future.

The adjusted bandwidths used as weights in the above algorithm are based on values reported in the consensus for each relay, further adjusted by *utility weights*. Since Tor's bandwidth capacity is at a premium, and exit bandwidth capacity more specifically, this weighting mechanism is in place to make the most of these resources, so that the network as a whole does not suffer from overly poor performance. These weights are a function of the total bandwidth of each relay type, the total network bandwidth, and the relative bandwidths and relay flags of individual relays. As the bandwidth and relay composition of the network changes, the bandwidth weights of individual relays also change. Note that the weighted consensus bandwidths are scalars *without units*; it is best to think of them as "points", where relays with more points are more likely to be chosen in circuits. They are not actual bandwidth measurements, and so it becomes difficult to translate this metric to real-world client experiences. We will refer to these "points" as weighted bandwidth units (WBU).

In general, exit bandwidth is protected such that relays with the `Exit` flag are chosen in the exit position more than in other roles. In particular, guards that are also exits will find themselves used more often as exits and less often as guards. This design choice will have implications we will discuss later on.

Threat model. Tor provides anonymity properties against an adversary that has a limited visibility (SoV) of the network. The adversary may operate malicious relays in the network and attain guard and exit flags by meeting the thresholds set out by the Tor specification. The goal of the adversary is to have relays under its control selected as the guard and exit relays on the same circuit, thus compromising the Tor user. The adversary does not have unlimited bandwidth and we count any relays it compromises as its own (*i.e.* within its SoI).

Our investigation of guards is concerned with the choices for parameters made by the Tor community. These parameters are the guard rotation duration, which at present is set to a uniformly random time between 30 and 60 days, and the number of guards, which at present defaults to three.

5.3 *COGS* Framework

The design of the *COGS* framework is guided by Tor's guard path selection design, its governing parameters, the historical data sets available, and the research questions that we would like to answer. The design is extensible in that future research questions pertaining

to guard and path selection can also be investigated using the same framework with minimal effort.

The framework encompasses a) researcher-defined observables or run-time measurements, b) the data sets available from the Tor data-collecting service, c) a Tor client simulator with hooks into the internal running state of thousands of simulated clients, d) configuration files that instrument the simulator for each experiment, and e) log parsers for data aggregation and statistics. Figure 5.1 provides a graphical representation of the framework. We will describe each in turn next.

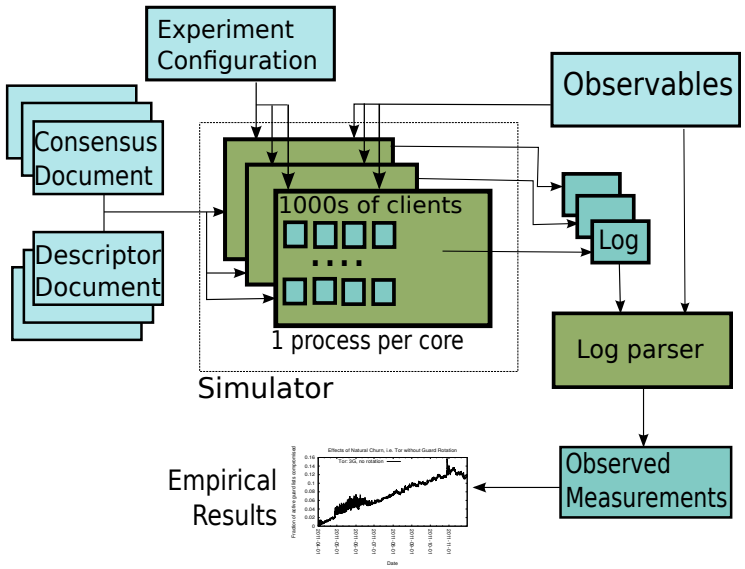


Figure 5.1: *COGS* framework

Observables. In order to drive the analysis and produce justifiable answers to the questions posed earlier, we define the following *observables*—metrics, attributes and effects that we want to measure. It is possible to introduce more observables for further research, some of which are outlined in Section 5.8.

From the historical consensus and descriptor documents we pick observables that will shed light into the behavior of guards. We focus on client compromise and how it is affected by natural churn and the operational parameters chosen by the Tor community.

The pattern of up time and down time for each relay provides insight into its *stability*. Using the consensus history we measure the consecutive down times of each relay; the same is done for up times. From this we calculate the mean time to recover between two runs

of up times as well as the mean time between failure between two runs of down times. Statistical analysis provides the average case for the general population of relays and that of guards.

To measure the impact of guard selection we also record the *number of guards that observe each client during the simulation*. This indicator is useful since it establishes the high water mark of potential compromise for each client. Even though each guard may have only been an active guard to a client for a short period of time, it is not safe to assume that the short period afforded limited impact on the client's privacy, since that short period may have been very sensitive in nature.

Additionally, we measure the number of clients at each consensus for whom at least one malicious guard is in the *active guard list*; we term this event *guard list compromise*. The active guard list is the first N online relays in the client's guard list ordered by age, where N is the number of entry guards being utilized by the client. This metric provides a view from the adversary's perspective of how many clients it could potentially compromise at any given time. Whereas Tor will always maintain a minimum of two online guards, we experiment with active guard lists that at times shrink to one in Section 5.4.

Finally, to evaluate the effect on performance of reduced active guard lists that may occur due to changes to Tor's default behavior, we measure the occurrences of active guard lists whose *average bandwidth falls below a certain threshold*. The number of active guards is not as important here as the average of their bandwidths, since this value can directly influence the client's expected performance. We measure the average active guard list bandwidth as an indicator of the end user's experience and not as an expectation of the performance of any particular circuit. Recall also, from Section 5.2.2, that the weighted consensus bandwidths do not represent absolute bandwidths; nonetheless, we can meaningfully compare the schemes against each other to find the relative merits of each.

Data sets. The Tor data-collecting service provides hourly snapshots of publicly downloadable Tor relay descriptors and actual published consensus documents from mid-2007 to the present.[5] This data offers a glimpse into the state of the Tor network over the years in terms of the total number of relays, their flags, and their bandwidths. In addition, the presence (or absence) of any particular relays enables us to analyze relay stability over time.

[5]https://collector.torproject.org/archive/relay-descriptors/

Configuration files of run-time options. We can change the behavior of Tor clients, the adversary's attributes, and the network characteristics by passing parameters at run-time through configuration files. Many experiments can be run simultaneously and independently—contingent on compute and storage resources—to provide insights into the behavior of stock Tor and the many interesting variations that research questions introduce. This mechanism allows us to attain answers in an efficient and reproducible manner. We discuss our parameter choices below in more detail.

Tor path selection simulator. Using the publicly available data sets and our selected observables, we constructed a Tor path selection simulator that selects guard relays and generates paths for a large number of simulated Tor clients. The simulator takes two pieces of data and a configuration file as input:

1. *Consensus documents*: The simulator reads unmodified consensus documents, one at a time, over the course of the time period desired. The consensus provides information such as each relay's bandwidth weighting and its flags.

2. *Relay descriptors*: The simulator also reads in relay descriptors that correspond to each relay listed in a particular consensus to allow correct Tor client behavior.

3. *Run-time options*: The simulator takes run-time parameters to introduce malicious relays (if an adversary is modelled), augment the behavior of clients (if required), choose the number of clients to be simulated, and produce logs of the observables.

In order to ensure the highest possible level of fidelity to Tor's design, our simulator is based on Tor's original source code (version 0.2.2.33). For each consensus period, the simulated clients select or update their guard lists, following all of the Tor rules for guard replacement as described in Section 5.2.2.

Our simulator allows us to control the guard rotation mechanism built in to Tor to test the effects of various guard rotation durations (or lack of them) on client compromise and also allows us to investigate the effects of client guard list size.

The granularity of our simulations is one hour, which corresponds to the granularity of the consensus documents. Every consensus lists the relays that were available at the time; they are loaded into the memory of our simulator, which then proceeds to select guard relays according to Tor's procedure for every client. These guards are written to a log file for later processing. Each consensus is fed into the simulator as a means to walk through time and produce guard selection scenarios. It uses parameter settings provided

by us to simulate different network characteristics such as the number of guards, guard rotation period, and others. Where consensuses are missing from the Tor Metrics dataset, the simulator skips that hour of history but all time-sensitive rules and operations are followed and are reflected in the simulation results.

We can simulate an adversary with a fixed budget of relay bandwidth by injecting it into the list of routers in each consensus period. The adversary is modeled by the amount of bandwidth it owns and the number of nodes it controls.

We also instrument the Tor client code to log client state to disk for all observables we are interested in. We refrained from logging all state changes due to storage constraint considerations.

We have made *COGS* available as open-source software and it is available from https://crysp.uwaterloo.ca/software.

Simulation setup and parameter choices. Our simulations were run on multi-core servers to take advantage of parallelism in the experiments. Each simulation run introduced 80,000 clients.[6]

It is not yet clear how to best model the client behavior as there is yet no consensus within the Tor community on real-world client behavior. Indeed, this is a research problem in itself and out of the scope of this work. Therefore, we model the user base size as constant with no new clients joining the network, since our simulations focus on long-term effects that are not sensitive to user churn. For simplicity the simulated clients are always online, which is a worst-case scenario since live clients do not use Tor continuously.

We choose the duration of our simulation by providing the starting and ending epoch times. We chose Apr 2011–Nov 2011 as our target time slice since it has relatively stable bandwidth characteristics and a consistent consensus version number.

COGS allows the injection of malicious routers into the network at run time through a configuration parameter.[7] We have chosen to introduce the malicious relay one consensus period, i.e. one hour, after the simulation has begun and all clients already have honest guards in their lists. This simulates an adversary attacking Tor after clients have already started using it and also establishes more conservative compromise rates—effectively a lower bound. For our simulations we assigned our malicious relay the guard flag only since

[6]This sample size was chosen by conducting experiments of increasing sizes and finding the point at which the resulting distributions stabilize, according to the Kolmogorov-Smirnov distance.

[7]While this paper investigates only one value of this parameter, it is simple to instantiate other behaviors through other values.

also having the exit flag reduces the probability of a router being picked as a guard relay and would confound our results. Note that the choice to operate an exit node is with the relay operator and is not controlled by any authority.

The bandwidth assigned to this relay, approximately in the top 20% of guards, is incorporated into the network using the same rules and bandwidth weightings as the normal routers. Using the results from Murdoch and Watson [MW08], we only introduce one malicious relay because Tor's guard selection algorithm chooses guards in proportion to their bandwidth; this design means that an adversary operating one high-bandwidth relay is equivalent to one operating many low-bandwidth relays as long as the total bandwidths are the same. Since we consider the adversary to be intelligent and capable of leveraging any and every advantage, we consider a client to be compromised if even one malicious guard exists in her active guard list.

It should be noted that while we initially set malicious bandwidth as a proportion of the total bandwidth, this proportion changes over time along with the total network bandwidth. We reason that keeping this value constant does not harm the experiments since i) a real adversary would not measure bandwidths on the entire network to keep malicious bandwidth proportions constant and ii) the bandwidth variance is small in our selected time period.

Log parsers and data visualization. The log parsers—there are several variants depending on the observables—extract the data we are most interested in and compile it into a format that can then be fed into data visualization programs. The data can be processed by a variety of parsers in order to gain insight into various aspects of guard design.

Our existing parsers process the raw logs to provide data on compromise rates, total guard exposure over the experiment run and expected client performance.

While *COGS* is rooted in guard analysis, it can also be used to simulate other Tor-related phenomena that do not involve actual network traffic. Examples include the analysis of client circuit diversity, the effects of introducing exit guards, and assessing whole-network effects of heterogeneous client configurations.

5.4 Measurements and Evaluation

We next use *COGS* to collect the empirical data that will be used to answer the four open research questions introduced earlier. The main aim is to understand the effects of various guard design choices on compromise rates. We measure the frequency with which a client

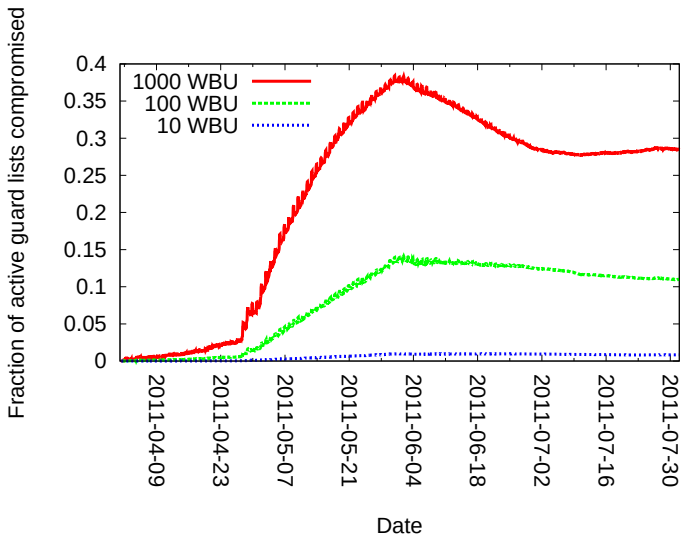


Figure 5.2: Client compromise rates at various adversarial bandwidths, where *WBU* is the amount of weighted bandwidth units assigned to the malicious relay.

picks new guards, since the more often guards are picked the more often a malicious relay has the chance to be placed in the client's guard list. The two main influences on the frequency of guard selection—other than a new client joining the network—are *natural churn* and *guard rotation*. We measure and evaluate characteristics of each in order to better understand the threats to client privacy.

Adversarial bandwidth and compromise rates. We base the subsequent analysis on the assumption that malicious bandwidth is directly related to compromise rates, albeit in complex ways. As the adversary increases their bandwidth contribution they are able to compromise more clients. This result is by design, as the Tor guard selection algorithm favors relays with higher bandwidth. We confirm this assumption in Figure 5.2, which shows that as the malicious bandwidth increases the compromise rates also increase.

Since relay bandwidth is independent of the other variables under study, we keep the malicious relay's bandwidth constant at 100 *WBU* for the rest of our experiments.

Table 5.1: Up and down times, in hours, of guard relays for Apr–Nov 2011. Guard down or up means the number of consecutive hours a guard relay was offline or online respectively. All down or up means the number of consecutive hours any type of relay was offline or online respectively.

	Min	1st Qu.	Median	3rd Qu.	Mean	Max
Guard Down	1	1	3	11	42.17	4978
Guard Up	1	7	20	127	156.7	3829
All Down	1	3	10	20	45	5454
All Up	1	1	4	11	19.82	3829

5.4.1 Natural Churn

To measure the effect of natural churn, we start by first analyzing the consensus data and establishing the pattern of churn (e.g., up and down times) for each relay over time. The subsequent statistical analysis provides the results in Table 5.1. Note that we allow for the effects of relays that had a high frequency of up/down events, and that only relays that were available April to November 2011 were included in the data set.

The distance between the leftmost and rightmost curves in Figure 5.3 indicates that guards are more stable compared to the general router population, due to their longer up times and shorter down times.

Next, we measure the effect of natural churn on guard list compromise and present the results in Figure 5.4 as the lower curve. For this analysis we have removed the normal guard rotation mechanism in the Tor client to isolate the effects of natural churn. We note that natural churn occurs frequently and also has a large effect on the network as indicated by the large uptick in compromised guard lists over time. The sharp peaks and valleys between May 1, 2011 and June 30, 2011 are indicative of honest guards that go down briefly—during which time our malicious guard has an opportunity to move into the active guard list—and then return—which bumps the malicious guard out of the active list again. These characteristic short guard down times concur with both Table 5.1 and Figure 5.3.

From the upward trend of the curve we now know that natural churn has a real and lasting effect on client security and increases with time. Given enough time a long-lived adversary will appear in all clients' guard lists. This risk can be mitigated with periodic guard rotation, which is presented next.

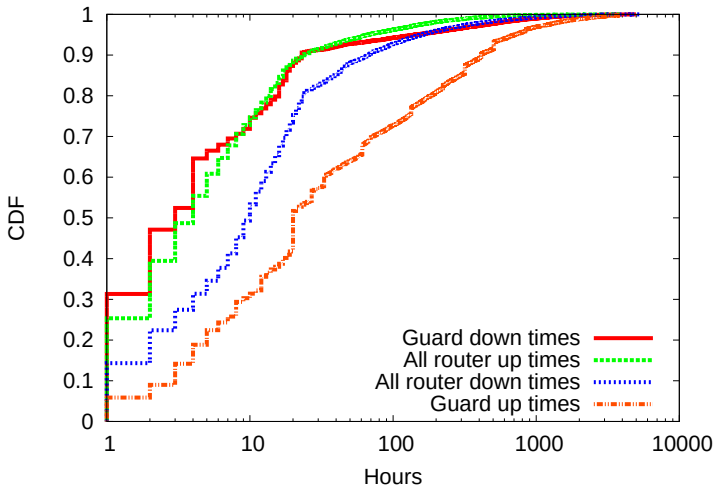


Figure 5.3: Router up and down times for all routers and for guards alone.

Figure 5.4: Effects of natural churn and guard rotation on active guard list compromise.

5.4.2 Guard Rotation

The second factor to guard list compromise is the mechanism to rotate each client's guards after defined periods of time. By default a Tor client drops its guards that are between 30–60 days old in the guard list. There are two major reasons: to limit the number of clients a single well-resourced guard can service, and hence compromise, at any given time and to balance the load so that long-serving guards do not potentially end up bearing the load of more clients over time. A negative effect is that clients with all honest guards are exposed to potentially selecting a malicious guard upon rotation, thus ensuring that after enough time all clients will have been compromised at some point.

It is difficult to isolate the effects of guard rotation from those of natural churn under simulation with real data. We can, however, analyze the effects of guard rotation in closed form and also analyze the empirical results of the additional effect of guard rotation to natural churn in simulation.

During our target time slice of eight months, we expect that every client will rotate their guards at most as often as every 30 days and at least as often as every 60 days. The maximum number of potentially unique guards that a client selects in those eight months is therefore 24, the minimum is 12, and the average is 17. This value is the number of guard relays that can potentially compromise the client. Note that without guard rotation, the least number of guards per client would be three.

The upper curve in Figure 5.4 shows the additional effect that guard rotation has on compromise rates. In the first 30 days we see a steady increase on both curves in compromise rates as only natural churn is in effect. Then between 30–60 days the guard rotation really begins to show its effects in the upper curve, peaking at the end of May after which point a steady state seems to have been reached, where the amount of new compromised active lists is offset with losses in compromised active lists. The upward and downward trends are in part due to the malicious relay being pushed out of the active guard list by honest relays returning from a downtime.

It is obvious that guard rotation increases the chances of active guard list compromise substantially. This result implies that guard rotation has a larger effect on compromise than does natural churn alone. Although it is difficult to isolate the interplay of natural churn and guard rotation it is simple to see that guard rotation does have negative effects.

A key takeaway here is that the nature of guard rotation and natural churn are different, which explains the disparity between the curves. Guard rotation *replaces* a guard, while natural churn only provides a *backup* guard. If the client picks no malicious guards (as is the case initially), then with only natural churn in effect the malicious relay can only hope to be picked once a client's guard goes offline. However, it will *never* be at the top of the

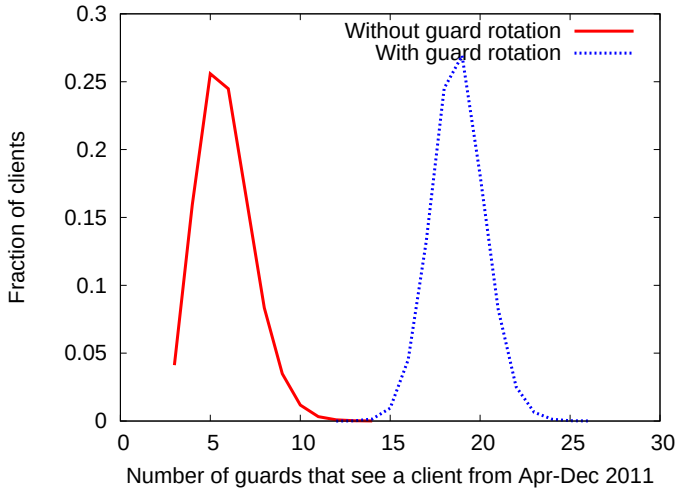


Figure 5.5: Comparison of natural churn and guard rotation effects on clients' exposure to guards.

list and will be bumped out of the active list once the original guard returns. On the other hand, when guard rotation is used, every 30–60 days the malicious relay has a chance to be picked as one of the first three guards, thus cementing its place in the active guard list and thereby enabling potential compromise whenever it is used.

Figure 5.5 shows the fraction of clients that have been seen by various numbers of guards for Tor with and without guard rotation. Guard rotation increases the visibility of each client on average to 19 guards. Recall that rotation causes at least 15 guards to see the client at minimum, so coupled with natural churn this effect is amplified. The effects of natural churn alone are small according to this metric: the mean increases to five versus the minimum three guards per client as indicated by the left curve.

As a counterpoint we observe that guard rotation does serve a beneficial purpose. As mentioned earlier, it reduces the likelihood that certain long-lived guards will accumulate a large set of clients and hence potentially compromise them. This self-limiting nature means that it is not desirable to remove guard rotation as a mechanism without a suitable alternative; we are actively exploring this area as ongoing work.

5.4.3 Guard List Size

Next, we investigate the effects of the size of the client's guard list and provide results and analysis for various values. We include results both with and without guard rotation enabled. For these experiments we run independent simulations for each of the guard list size settings so the clients are homogeneous within each run.

Recall that the client will only replace a guard if guard rotation dictates it (if in effect) or supplement it when there are fewer than two guards online from the client's guard list. Figure 5.6 shows client compromise rates with guard rotation when the size of the client's guard list is 1, 2, 3, 5 and 10 guards, where the 'G' stands for guards. From this analysis we discover that increasing the size of the guard list increases the client compromise rates.

However, compare these rates to the results without guard rotation in Figure 5.7, where the absolute compromise rates are far lower but steadily increase over time. Also note that with guard rotation off, increasing the guard list size beyond 3 guards has the reverse effect of decreasing client compromise (curves for 5 and 10 guards). However, this effect does not last. We see the curve for 5 guards crossing over the 1 guard curve, with all indications of eventually crossing over the 2 and 3 guard curves as well if the upward trend continues. The same trend occurs for the 10 guard curve. The reason behind this trend is that initially the pool of possible guards is large and all are online; as guards fail, the client does not take any steps to replace them since the size of the guard list is still large enough and at least two of them are online. As the guards that failed are removed from the list, more guards are picked to maintain the overall size of the client's guard list. This last effect slowly erodes the advantage of starting off with a large pool of guards.

We now consider the number of guards seen over time for different starting guard list sizes. Figure 5.8 shows the effect of increasing guard list size on clients' guard exposure. It is apparent that increasing the guard list size increases the client's guard exposure.

Figure 5.9 provides results for when guard rotation is turned off. While the overall guard exposure is far less than when guard rotation is in effect, we see the same trend where larger starting guard list size equates to more guard exposure. We observe that as the client guard list size increases, the probability of more guards ever being added to the list decreases. This effect is particularly striking for the 10 guard curve, and also evident for the 5 guard curve. This result is due to relative guard stability and also to the condition that fewer than two guards be present before a new guard is added. Comparing both figures we see a more general trend that without guard rotation the value of guard exposure is close to the starting guard list size (more pronounced for higher values) whereas with guard rotation the values of guard exposure are many times larger.

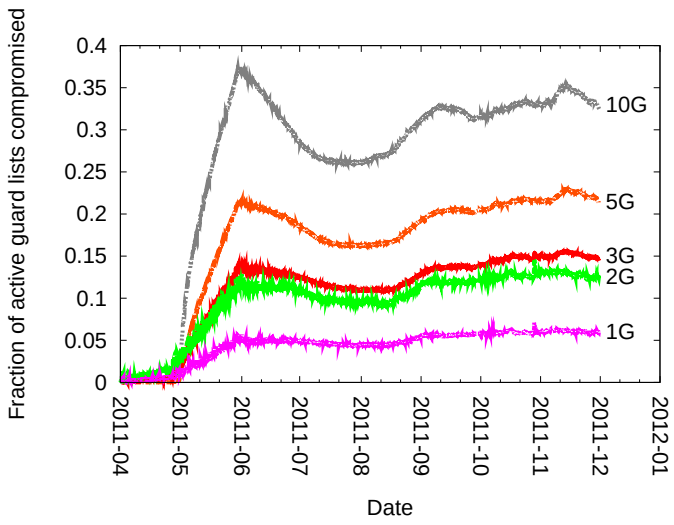


Figure 5.6: Client compromise rates at various client guard list sizes, with guard rotation.

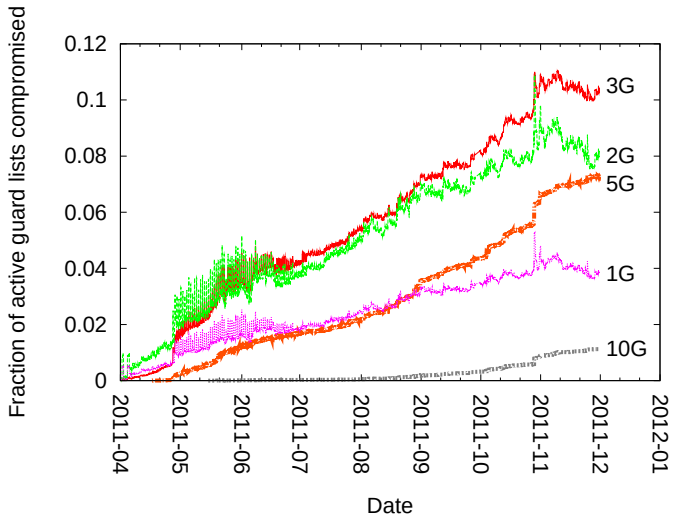


Figure 5.7: Client compromise rates at various client guard list sizes, without guard rotation.

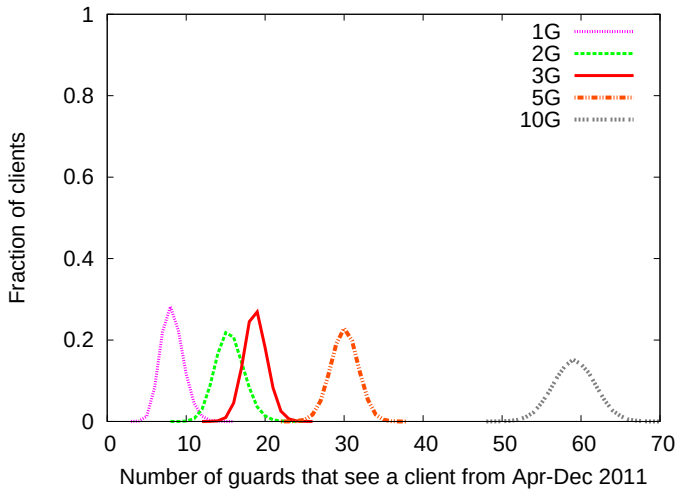


Figure 5.8: Client guard exposure with guard rotation at various guard list sizes.

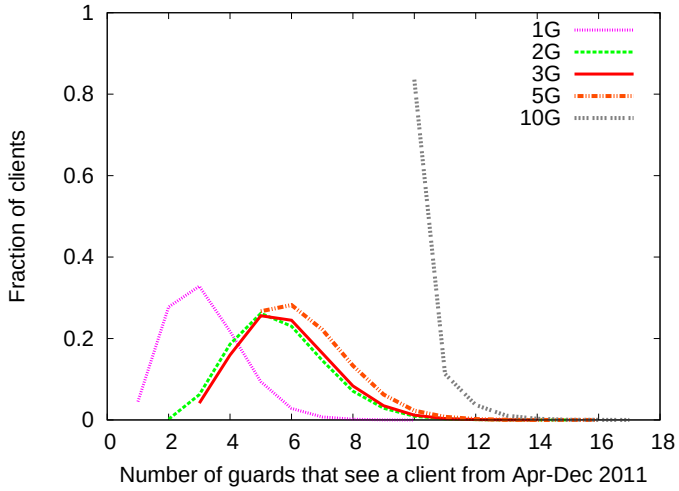


Figure 5.9: Client guard exposure without guard rotation at various guard list sizes.

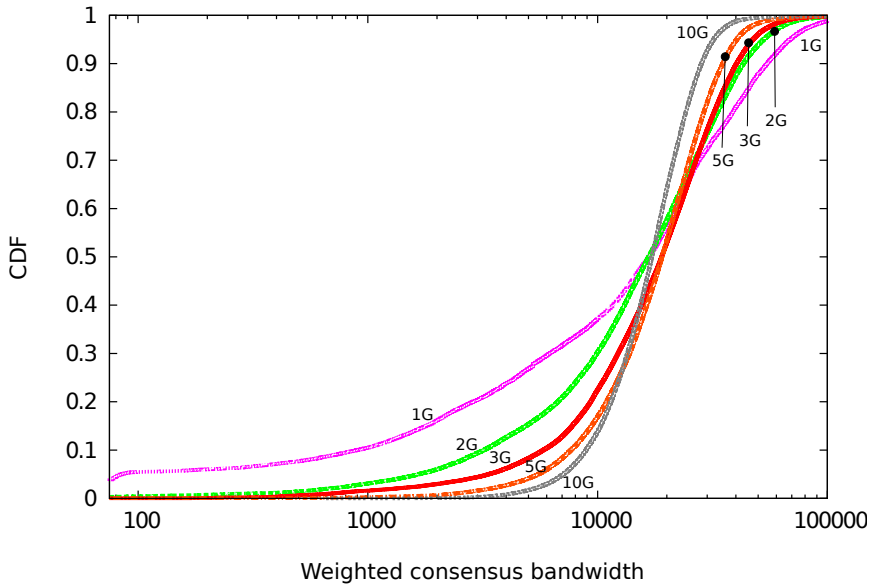


Figure 5.10: Client's expected circuit performance *with* guard rotation at various guard list sizes. Performance results *without* guard rotation are nearly identical.

5.4.4 Available Bandwidth

Before we can make any conclusions we must look at the effects of these parameters on the average available bandwidth a client's guards provide it. Figure 5.10 shows the expected bandwidth for a client circuit. Results with and without guard rotation are nearly identical with negligible variations meaning that average performance is independent of guard rotation.

Recalling that higher-bandwidth guards are more likely to be selected for spots in a client's guard list, poor guard bandwidth availability happens when *all* of a client's active guards have low bandwidth. This situation occurs with decreasing probability as the number of active guards increases, as is reflected in the dramatic decrease in the long left tail in Figure 5.10 as the number of guards increases from 1 to 3. Above 3, however, the improvements are less pronounced.

5.5 Discussion

Next we discuss the implications of our findings and address Dingledine's open research questions.

Table 5.2: Median guard bandwidth (WBU) from Apr–Nov 2011

Min	Median	Mean	Max
40	67	68.31	113

Guard stability and selection. As guards are the first hop on circuits, all of Tor's functionality is contingent on their availability. Section 5.4.1 shows that on the whole guards are quite stable. Compared to the general population of relays, guards are generally available for longer stretches of time and offline for shorter durations. This stability is a consequence of the guard flag assignment process governed by the directory authority and is as designed.

However, this process is not perfect, as we see that there are a large quantity of guards with a wide variety of stability characteristics that deviate from the intended entry guard design—recall Table 5.1 for the range of downtimes and uptimes for guards. We note that the incidence of active guard lists with low average bandwidth in general is not prevalent; note that the curves for 3–10 guards in Figure 5.10 do not have long tails to the left of the median as compared to the 1 guard curve—meaning occasions where every guard in a client's active list has low bandwidth are rarer—and that perhaps the guard flag allocations could be more selective. Indeed, Table 5.2 provides statistics on the median guard bandwidth during our 8-month time slice; it is calculated by finding the median WBU amongst all the guards in the consensus and then calculating statics based on those medians across all the consensuses. It shows that the greatest median guard relay bandwidth across all consensuses during that time slice is just 113 WBU, a level of active guard bandwidth which is surpassed by all clients with "3G" or more and only suffered by 5% of "1G" clients (Figure 5.10).

We reason that since low-bandwidth guard lists are rare, low-bandwidth guards are not depended upon by end users and so removing them from guard lists will not have a big impact from a performance perspective.

However, it can be argued that for the sake of load balancing these low-bandwidth relays provide relief whenever the end user chooses them from their guard list instead of one of their higher-bandwidth guards. These nodes may also provide added security through additional relay diversity. We shall see in Section 5.7, where we discuss the impact of *COGS*, that there is support to increase the minimum bandwidth of relays which may mitigate these performance issues.

Natural churn and its effects on client compromise. In the lower curve in Figure 5.4, it is clear that natural churn provides an adversary increased opportunities to

compromise guard lists. We also note that although there is some downward pressure due to returning honest guards, the trend is upwards over time. If not for guard rotation, after a sufficient length of time a malicious relay should be able to compromise all client lists. Recall that while guard rotation speeds up the adversary's accumulation of clients initially, it is self limiting as the rate of clients gained equals the rate of clients lost due to guard rotation.

Furthermore, when reasoning about the impact of natural churn it is difficult to know beforehand when a guard is likely to return, if ever. It is due to this uncertainty that Tor uses such sensitive guard replacement policies and sophisticated retry mechanisms.

Putting natural churn in perspective, we can reason that it is an artifact that cannot be removed from the network, and it has a large effect on the security and performance of the network. Therefore, the best policy may be to avoid situations that lead to churn in the first place by selecting guards more cautiously and mitigating the effects of churn when we do find one of our active guards offline.

Guard rotation. Long-lived relays tend to accumulate clients over time, and malicious relays will remain online to take advantage of this effect. Rotating guards does in fact mitigate that eventuality.

We note that guard rotation does, however, increase the chance of compromise: see the sudden increase in May 2011—when guard rotation began to take effect in our experiment, 30 days after its beginning—of the curve in Figure 5.4. We also note that in Figure 5.5 the number of guards a client is serviced by, and can hence potentially be compromised by, is much larger when guard rotation is in effect. Furthermore, Figure 5.9 indicates that all schemes expose clients to fewer guards when guard rotation is not enabled. As mitigation of the above, we could increase the minimum and maximum durations of guard rotation from the current 30–60 days and see a reduction in both metrics, since the frequency of rotation events would decrease.

In order to reason about rotation durations and pick better ones we plot in Figure 5.11 the CDF of guard longevity for Apr–Nov 2011. We note that only about 9% of guards remained part of the Tor network for the entire 8-month duration of our experiments. Also, the distribution is skewed towards shorter-lived routers with the median at 57.125 days. The current rotation period is between 30–60 days which means that the majority of guards undergo guard rotation. Since most guards are not long lived and leave the network of their own accord, guard rotation occurring as frequently as it currently does is both unnecessary and undesirable. We would prefer to target only those guards that are truly longer lived and thus are the cause for our concern, and ignore those that simply do

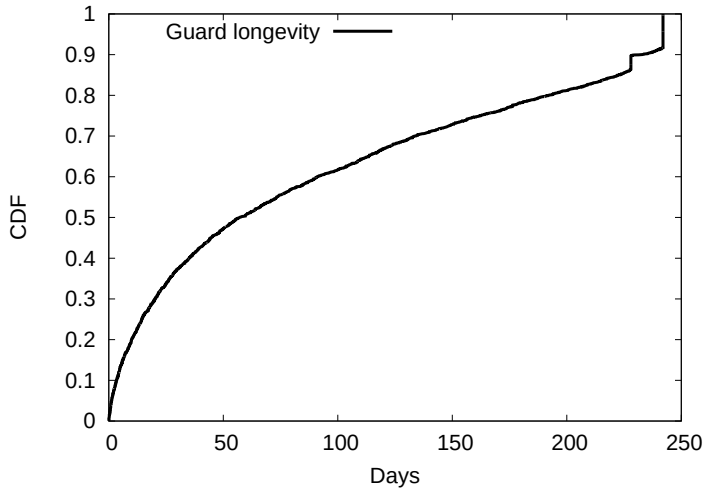


Figure 5.11: Guard longevity during Apr–Nov 2011.

not exist long enough to be worried about. Unfortunately, there is not an upward inflection point, apart from the two small ones at the extreme end of the time slice, which would indicate that longer-lived guards stand apart from the others and thus can be dealt with using a more appropriate rotation duration.

Perhaps as an alternative for longer-lived, and potentially more-utilized guards, Tor ought to adjust their probabilities of being selected according to how long they have been part of the network, in addition to their bandwidths. Indeed, we are now seeing that these alternatives are being explored and implemented by the Tor community. [Din14]

Tor with one guard. Intuitively, it seems that one guard ought to provide the best security but that perhaps performance would suffer. We revisit the results in Section 5.4 to evaluate this intuition. From a circuit compromise perspective, we see in Figure 5.6 that Tor with one guard offers the least likelihood of compromised guard lists. We also note that fewer guards participate in an end user's guard list in that case (Figure 5.8). However, from a performance perspective we note in Figure 5.10 that compared to Tor with three guards, Tor with one guard suffers from 60% worse performance 50% of the time but is better 50% of the time where it provides 25% more average guard list bandwidth. This outcome can be explained with guard lists that have a combination of slow and fast guards,

which causes the average to be lower than the fastest guard. In the case of Tor with only one guard, when a fast guard is selected, the client can expect to receive fast service, provided that the middle and exit nodes are not slow. It is important to note, however, that Tor with one guard is superior to the other schemes evaluated in Section 5.4 when the bandwidths are already at acceptable levels, whereas it provides far slower performance at the lower ends of the bandwidth spectrum. As we mentioned above, due to the effort to raise the minimum bandwidth for relays these performance issues can be mitigated.

Hence, the number of guards is a parameter that needs careful adjustment: our present results suggest that too few may lead to performance degradation, while more can have unnecessary security implications. Indeed, an optimal solution seems to be a single guard where all guards are required to have a higher minimum bandwidth. This idea has been taken up by the Tor community; see Section 5.7.

5.6 Related Work

Entry guards were first proposed by Wright *et al.* [WALS03] (there called "helper nodes") to mitigate the threat of the predecessor attack [WALS02] in low-latency anonymity networks. In the predecessor attack, an adversary who deploys relays into the anonymity network can passively link possible senders with possible receivers. If clients choose their paths through the anonymity network by uniformly random selection, the predecessor attack predicts that an adversary that controls c out of n nodes has the expectation of successfully observing a given client after c/n rounds; the same adversary has a $(c-1)/(n-1)$ probability of observing the corresponding destination server and a $(c/n)((c-1)/(n-1))$ probability of linking the two. To eliminate the predecessor attack, Wright *et al.* propose that the first node in a path be fixed. Clients who have the misfortune of choosing a malicious entry node are guaranteed to have a compromised first hop, while all other clients are protected from this threat.

Entry guards for modern onion routing networks (like Tor) were proposed by Øverlier and Syverson [ØS06]. Since Tor does not choose circuits with uniform selection over the available nodes (but instead, in proportion to each node's bandwidth capacity), the details of the analysis of the predecessor attack are more complicated. However, Øverlier and Syverson found that an adversary who artificially inflates his perceived bandwidth capacity will be selected more often and can launch a powerful predecessor attack. To mitigate this threat, they propose that Tor clients choose a small, fixed number of Tor relays to always use as entry points into the anonymity network.

Extending Øverlier and Syverson's predecessor attack, Bauer *et al.* [BMG+07] showed that an adversary who controls a large number of nodes can launch a Sybil attack that has the effect of replacing all non-malicious entry guards with malicious ones (potentially all running on the same machine). The attack works by deploying enough malicious nodes that advertise high bandwidth and uptimes to effectively raise the criteria for the guard flag so that only malicious nodes can be used as entry guards. This attack was dangerously easy to launch, due to the fact that Tor's authoritative directory authorities relied solely on self-reported (and potentially inflated) bandwidth and uptime claims. In part due to this attack, the directory authorities now track each router's bandwidth and uptime [BM07a, Per09], and ensure that no one can launch too many malicious nodes from the same machine (or network) [BM07b].

Borisov *et al.* [BDMT07] describe the effects of entry guards on the selective denial of service (DoS) attack. They argue that while the selective DoS attack will never be effective on a client that uses honest entry guards, the attack becomes more powerful when a client uses malicious entry guards. The authors also suggest that the choice of three entry guards results in the highest number of compromised circuits, and they suggest fixing both the entry and exit ORs as suggested by Wright *et al.* [WALS02].

Abbott *et al.* [ALLP07] describe a browser-based attack on Tor where a malicious exit injects a signal generator to the user's traffic. A malicious entry guard is required to perform traffic analysis on its clients' circuits to identify if a circuit carries the injected signal. If such a circuit is identified, then the attacker is able to link the client to its destination. A strong point of this attack is that it does not require both entry and exit to compromise a circuit at the same time, as it only requires that a malicious entry guard detect a specific signal encoded by a malicious web service. The authors argue that using three entry guards helps to protect clients that use honest entry guards. However, the attack becomes more effective for unlucky clients who use malicious entry guards.

Since its initial proposal for Tor, the entry guard design has become more sophisticated, including the many minute details described in Section 5.2. However, to date, there has been no thorough investigation into the security and performance implications of Tor's entry guard design. This work serves to fill this gap.

The major next step is to use the results presented here coupled with further *COGS*-driven analysis to answer the final question posed by Dingledine [Din11b]: how should Tor assign guard flags to find the right balance between assigning the flag to as many relays as possible (for diversity) and minimizing the chance that a client will use the adversary's relay as a guard?

A related research problem currently under way is the Tor client model. We noted in Section 5.3 that it is unclear how to model the Tor client base and the adversary's insertion

strategy. We have presented results where the adversary arrives after all clients have picked their guards, and no client leaves the Tor network or joins it. Counterintuitive properties—like those in Figure 5.7 where increasing guard list size actually reduces compromise rates—may not hold for other conditions. We need better models that accurately reflect user and adversary behavior in the Tor network in order to properly resolve these questions.

We are also presently considering alternative guard selection algorithms that have desirable properties. As an example of one possible direction, we note that in Section 5.5 the consensus bandwidth weightings currently utilized to control the guard selection process could be augmented with an age-related weighting that would affect the probabilities of a guard's selection. Also being examined, and closely related, is Tor's "weighted-fractional-uptime" metric—a component in ensuring that the guard flag is given to a relay with little churn—which could be replaced with an alternative calculation that better predicts relay churn behavior. Another example is a trust-based [ØS06] guard selection scheme such that clients pick guards according to how much they trust them. One final example is to investigate the condition that guards are only added to a client's guard list when fewer than two online guards remain in the list; further analysis is required to learn how this strategy may interact with various guard selection algorithms.

5.7 Impact

Our contributions have spurred the Tor research and development communities to replicate and verify our results [BPW13, JWJ+13] and to consider how to adopt our recommendations [Din13a, Din13b]. To assess the implications, Kadianakis [Kad14] and Hopper [Hop14] investigated the performance impact of limiting clients to a single guard and conclude that by raising the bandwidth requirements for guard relays to 2 Mb/s the likelihood of degraded performance would be reduced. Incorporating the findings above, Dingledine *et al.* [DHKM14] presented a more formal security and performance analysis to assess the implications of adopting our recommendations. Their findings were in favor of adoption and thus the authors also produced an implementation proposal [KHM14] for the Tor development community as a reference document.

Since 2014, the guard rotation period has been increased from 2–3 months to 9–10 months and the default guard count has been decreased from three to one. [Mat14] The minimum bandwidth requirement has been extended to all types of relays and the Tor project now officially encourages at least 2 Mb/s to new relay operators. [Tor14] Figure 5.12 shows a timeline of these developments.

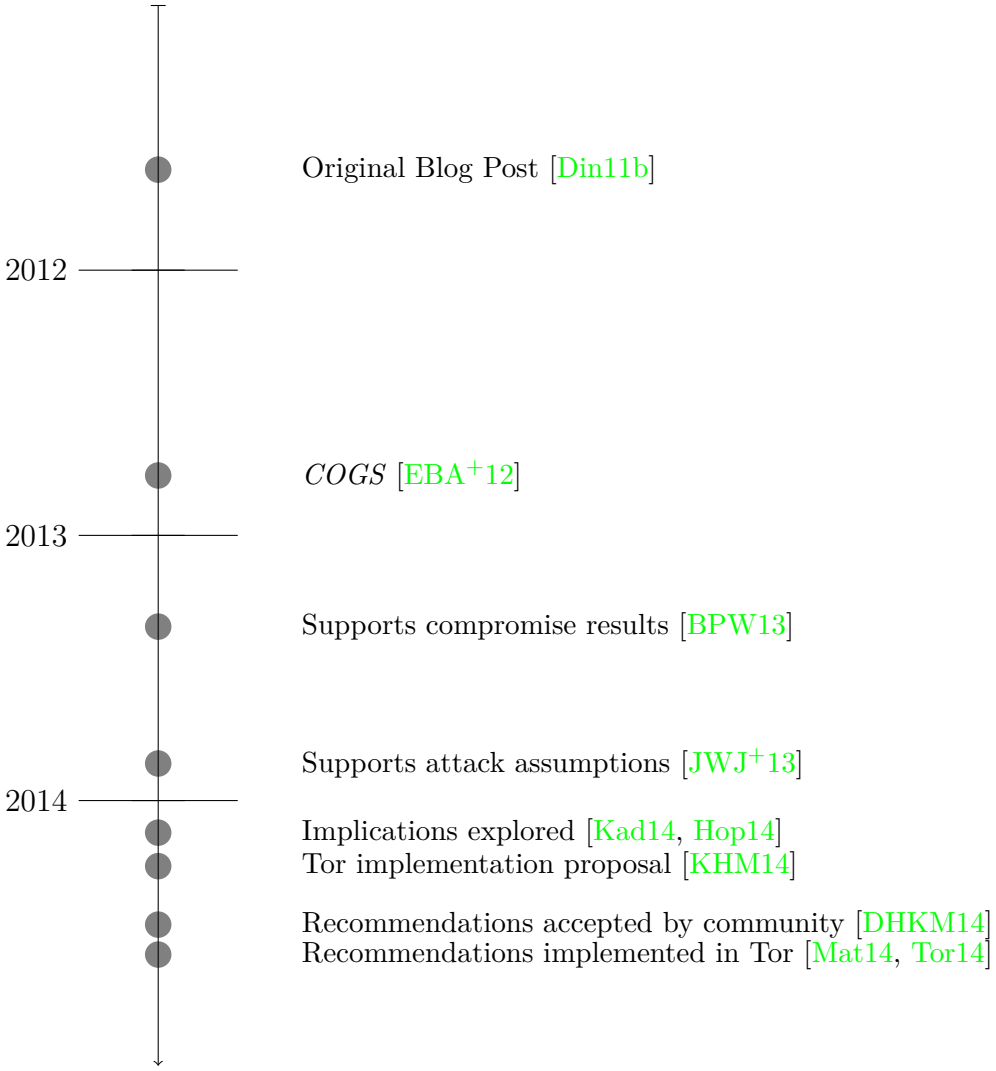


Figure 5.12: Timeline of subsequent research and developments to the Tor network due to *COGS*.

5.8 Conclusion

We constructed *COGS*, a flexible simulation framework, and used it to investigate open research questions relating to Tor's entry guard design.

We now tie the results of our investigation to the questions posed in Section 5.1, and see how much progress has been made and what remains to be answered.

We found that adversarial bandwidth is directly related to client compromise rates and that this is an unavoidable effect of favoring higher-bandwidth relays—recall that this is a design choice for the sake of better performance. What is interesting is that, as seen in Figure 5.2, bandwidth and compromise rates are not linearly related. While more research is required to establish the exact relationship, it is clear that performance-enhancing measures have led to higher client compromise rates in this regard.

We found that users achieve greater security if they reduce the number of entry guards they use. They can further improve their security by eliminating or reducing the process of guard rotation. However, we also found that the security improvements through the reduction of guards and guard rotation come at the expense of performance degradation. We also found that natural churn, while inherent in the network and a source of compromise, works to amplify the compromise rate, but is not a dominating factor in the present Tor network.

From the perspective of CRS strategies we see that obfuscated values coupled with rate limiting is a viable strategy but one that requires careful thought and care when it comes to picking security parameters. Taking all of our findings about parameters together we find in general that if a suitable alternative to guard rotation can be found and smaller guard lists used, then the security of Tor's users will increase significantly while the impact to performance for clients with slower-than-average guards will degrade only slightly. The risks could be further mitigated by making the guard flag more selective and thus removing low-bandwidth guards, which would raise the average guard bandwidth for all clients. These recommendations have been accepted by the Tor community and have been implemented in the live network.

Chapter 6

Conclusion

6.1 Progress on Thesis

Our thesis suggests that better CRS designs can come from more formal and empirically driven analysis.

We provide support for this by first carrying out a systematization of the CRS space that exposed gaps and problem areas. The most glaring of these gaps was the lack of research into the censor's decision function. We fill it by introducing a game-theoretic framework, supported by theory and simulation. It allows us to analyze the censorship game and shows how CRS designers can manipulate the base rate and cause the adoption of favorable censor behaviors.

We also present two frameworks for collecting empirical data on CRS networks. The first, *PrivEx*, provides a privacy-preserving general-purpose data-collection system for aggregate statistics. We use it to learn the base rate of CRS activity on the Tor network. The second, *COGS*, provides a data-driven simulator and we use it to empirically analyze rate limiting and obfuscating strategies as implemented by Tor's routing algorithm. We discovered suboptimal parameter settings and provided recommendations that have been since been accepted and adopted into the live Tor network.

6.2 Limitations

There are two main limitations of this work and the applicability of the frameworks we propose. The first concerns the need for empirical data and the second the scope of our

models and analysis.

Empirically driven CRS designs are contingent on the availability of data. There are two reasons there may be a paucity of data in the frameworks we propose. The first, as we noted in Section 3.5, is that certain parameters may be unobservable by their inherent nature. We worked our way around this problem by using properties we can observe as a means of inferring information about these hidden parameters. The predictive power of our models would be greater if we had access to the actual empirically collected values. The second, which is the *raison d'être* for *PrivEx*, is that collection of data in the censorship resistance field, due to the risks associated with it, is still not universally acceptable by either the designers of CRSs or their user bases. Here, even though the data *is* observable and hence a source of empirical evidence and useful for our models, the community has not come to a consensus about what is safe to collect and what is not. Until this consensus is reached, the value to be gained from our frameworks will be limited to what we can learn from partial and ad-hoc deployments, thus also limiting the progress of empirically driven CRS designs.

Our models, and consequently the analysis, limits its scope to the technological realities of the censorship and CRS landscape as recorded in the literature to date. In order to distill the common features and systematize the field we focus on the censorship apparatus' error rates and the base rate of circumvention traffic. We use these as the basis for the analysis of the CRS strategies and the techniques that are employed. This is a useful abstraction, which yields the insights in Chapter 2 and Chapter 3; however, this perspective only provides a partial view of censor and CRS dynamics. The first reason is that there may be other parameters that are also relevant and that may yield further insights that our frameworks presently fail to model. The second reason is that the area of censorship and its resistance is concerned with more than the technological realities of the censorship apparatus and CRS techniques. Indeed, censorship and the struggle against it are human endeavors, which the social and political sciences can better reason about and provide the missing context to the purely technical treatment given in this thesis.

6.3 Future Work

We have described a number of avenues of context-specific future work in each of the preceding chapters. We now consolidate these avenues and provide a high-level research agenda that we believe are worth pursuing.

A major goal of this thesis is to help produce better CRS designs through a principled approach. It would be useful to understand the extent to which combining previous ap-

proaches to CRS design is feasible, and its limitations. Another related avenue is to explore whether certain designs can be avoided, such as the use of CRS-agnostic infrastructure. We proposed diffusion as a general idea, and it would be fruitful to see how it can be realized in a manner that is both better from a security and performance perspective as well as practical from the deployment perspective.

We explored the role of error rates and circumvention traffic volume as the basis of our analyses. As we pointed out above, this is a limitation of our work; it would be interesting to explore other parameters and how these can be modeled and consolidated within the frameworks we have proposed. These parameters should influence the security and performance properties that we have defined or the equilibria points that we identify. Extending the analysis to incorporate temporal dynamics would also be interesting since it would reveal how the strategy and technology space evolves.

Another avenue is to apply the kind of game-theoretic analysis performed in Chapter 3 to the strategies we identified in Chapter 2. The aim would be to see 1) if there are relationships that exist between the strategies and 2) to see how—perhaps along with temporal modeling—they can be combined together and if any meta-strategies emerge.

Finally, there is a dearth of empirical data that CRS research can leverage, and this partly stems from the limitations we have mentioned above. To study and overcome these limitations we propose the following related avenues of research:

1. Discovering and describing the CRS participant threat models as they relate to the information leaked by gathered statistics.

2. Identify the set of statistics that are key to learning about the important security and performance characteristics of a CRS. Here the set should be as small as possible while yielding the greatest amount of information, either directly or being derived from it.

3. Identify use-cases and investigate the level of granularity that is necessary to achieve useful statistics for each.

4. Explore topologies of the statistics collection system, such as centralized or laissez-faire for example. A central model allows more control while the other allows more autonomy for the collector and perhaps a more realistic model since we can be sure that statistics are being collected even today without any central oversight.

5. Engage the community and build consensus on 1) the whole idea of persistent and pervasive statistics collection and 2) what the procedure is for (research or otherwise) deploying data-collection systems on the Tor network.

By making progress on these fronts, and coupled with the frameworks we have proposed, we believe that empirically driven CRS designs will be more likely to be successful.

In summary, the contributions of this thesis, which are based on formal and empirical analysis frameworks, have led to a better understanding of CRS designs and the arms race between the censor and circumventor. We are mindful of the limitations of our work and propose that with continued research effort we will provide further insights about upgrading, building, and deploying CRSs in the future.

References

[AAH13] Simurgh Aryan, Homa Aryan, and J. Alex Halderman. Internet Censorship in Iran: A First Look. In *Proceedings of the USENIX Workshop on Free and Open Communications on the Internet*. USENIX, 2013.

[AB10] Tansu Alpcan and Tamer Başar. *Network Security: A Decision and Game-Theoretic Approach*. Cambridge University Press, 2010.

[ABEG13] Mashael AlSabah, Kevin Bauer, Tariq Elahi, and Ian Goldberg. The Path Less Travelled: Overcoming Tor's Bottlenecks with Traffic Splitting. In *Proceedings of the 13th Privacy Enhancing Technologies Symposium*, pages 143–163. Springer, July 2013.

[ALLP07] Timothy G. Abbott, Katherine J. Lai, Michael R. Lieberman, and Eric C. Price. Browser-based Attacks on Tor. In *Proceedings of the 7th International Conference on Privacy Enhancing Technologies*, pages 184–199. Springer-Verlag, 2007.

[AM06] Ross Anderson and Tyler Moore. The Economics of Information Security. *Science*, 314(5799):610–613, 2006.

[AMNO07] Ross Anderson, Tyler Moore, Shishir Nagaraja, and Andy Ozment. Incentives and Information Security. *Algorithmic Game Theory*, pages 633–649, 2007.

[And96] Ross Anderson. The Eternity Service. In *Proceedings of Pragocrypt*, 1996.

[Ano13] Anonymizer Inc. Anonymizer. https://www.anonymizer.com/index.html, 2013. Retrieved May 2015.

[Aum59] Robert J Aumann. Acceptable Points in General Cooperative n-Person Games. *Contributions to the Theory of Games*, 4:287–324, 1959.

[Bac97] Adam Back. Usenet Eternity. *Phrack Magazine*, http://www.cypherspace.org/eternity/phrack.html, 1997. Retrieved May 2015.

[BCK12] Richard Barnes, Alissa Cooper, and Olaf Kolkman. Technical Considerations for Internet Service Filtering. IETF-Draft, http://tools.ietf.org/html/draft-iab-filtering-considerations-01, 2012.

[BDG+13] Gilles Barthe, George Danezis, Benjamin Grégoire, César Kunz, and Santiago Zanella-Béguelin. Verified Computational Differential Privacy with Applications to Smart Metering. In *26th IEEE Computer Security Foundations Symposium*, pages 287–301, 2013.

[BDMT07] Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. Denial of Service or Denial of Security? In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pages 92–102. ACM, 2007.

[Ben94] Josh Benaloh. Dense Probabilistic Encryption. In *Proceedings of the Workshop on Selected Areas in Cryptography*, pages 120–128, 1994.

[BF13] Sam Burnett and Nick Feamster. Making Sense of Internet Censorship: A New Frontier for Internet Measurement. *ACM SIGCOMM Computer Communication Review*, 43(3):84–89, July 2013.

[BFJ+12] Robert M. Bond, Christopher J. Fariss, Jason J. Jones, Adam D.I. Kramer, Cameron Marlow, Jamie E. Settle, and James H. Fowler. A 61-Million-Person Experiment in Social Influence and Political Mobilization. *Nature*, 489(7415):295–298, 2012.

[BFK00] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: A System for Anonymous and Unobservable Internet Access. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 115–129. Springer-Verlag, LNCS 2009, July 2000.

[BFV10] Sam Burnett, Nick Feamster, and Santosh Vempala. Chipping Away at Censorship Firewalls with User-Generated Content. In *Proceedings of the 19th USENIX Security Symposium*, 2010.

[BHS14] Chad Brubaker, Amir Houmansadr, and Vitaly Shmatikov. CloudTransport: Using Cloud Storage for Censorship-Resistant Networking. In *Proceedings of 14th Privacy Enhancing Technologies Symposium*. Springer, 2014.

[BM07a] Kevin Bauer and Damon McCoy. Tor specification proposal 107: Uptime Sanity Checking. https://gitweb.torproject.org/torspec.git/blob_plain/HEAD:/proposals/107-uptime-sanity-checking.txt, March 2007.

[BM07b] Kevin Bauer and Damon McCoy. Tor Specification Proposal 109: No More Than One Server per IP Address. https://gitweb.torproject.org/torspec.git/blob_plain/HEAD:/proposals/109-no-sharing-ips.txt, March 2007.

[BMG+07] Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Low-Resource Routing Attacks against Tor. In *Proceedings of the Workshop on Privacy in the Electronic Society*, pages 11–20, October 2007.

[BNO08] Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed Private Data Analysis: Simultaneously Solving How and What. In *Advances in Cryptology–CRYPTO 2008*, pages 451–468. Springer, 2008.

[BPW13] Alex Biryukov, Ivan Pustogarov, and Ralf-Philipp Weinmann. Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, May 2013.

[BSMD10] Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas Dimitropoulos. SEPIA: Privacy-preserving Aggregation of Multi-domain Network Events and Statistics. In *19th USENIX Security Symposium*, August 2010.

[Cha81] David L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[CMW06] Richard Clayton, Steven J. Murdoch, and Robert N. M. Watson. Ignoring the Great Firewall of China. In George Danezis and Philippe Golle, editors, *Proceedings of the Sixth Workshop on Privacy Enhancing Technologies*, pages 20–35. Springer, June 2006.

[CSWH00] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, July 2000.

[DA04] George Danezis and Ross Anderson. The Economics of Censorship Resistance. *Proceedings of the 3rd Annual Workship on Economics and Information Security*, 2004.

[Dan04] George Danezis. The Traffic Analysis of Continuous-Time Mixes. In *Proceedings of Privacy Enhancing Technologies workshop*, volume 3424 of *LNCS*, pages 35–50, May 2004.

[DB15] Eva Dou and Alistair Barr. U.S. Cloud Providers Face Backlash From China's Censors. *The Wall Street Journal*, http://www.wsj.com/articles/u-s-cloud-providers-face-backlash-from-chinas-censors-1426541126, 2015. Retrieved May 2015.

[DCRS13] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. Protocol Misidentification Made Easy with Format-Transforming Encryption. In *Proceedings of the 20th ACM conference on Computer and Communications Security*, November 2013.

[DCS15] Kevin P. Dyer, Scott E. Coull, and Thomas Shrimpton. Marionette: A Programmable Network Traffic Obfuscation System. In *Proceedings of the 24th USENIX Security Symposium*, pages 367–382. USENIX Association, August 2015.

[DHKM14] Roger Dingledine, Nicholas Hopper, George Kadianakis, and Nick Mathewson. One Fast Guard for Life (or 9 Months). In *7th Workshop on Hot Topics in Privacy Enhancing Technologies*, 2014.

[Din11a] Roger Dingledine. Iran blocks Tor; Tor Releases Same-Day Fix. *Tor Blog*, https://blog.torproject.org/blog/iran-blocks-tor-tor-releases-same-day-fix, September 2011. Retrieved May 2015.

[Din11b] Roger Dingledine. Research Problem: Better Guard Rotation Parameters. *Tor Blog*, https://blog.torproject.org/blog/research-problem-better-guard-rotation-parameters, August 2011. Retrieved May 2015.

[Din11c] Roger Dingledine. Research Problems: Ten Ways to Discover Tor Bridges. *Tor Blog*, https://blog.torproject.org/blog/research-problems-ten-ways-discover-tor-bridges, October 2011. Retrieved May 2015.

[Din12] Roger Dingledine. Obfsproxy: The Next Step in the Censorship Arms Race. *Tor Blog*, https://blog.torproject.org/blog/obfsproxy-next-step-censorship-arms-race, February 2012. Retrieved May 2015.

[Din13a] Roger Dingledine. Brainstorm Tradeoffs from Moving to 2 (or Even 1) Guards. *Tor Bug Tracker* https://trac.torproject.org/projects/tor/ticket/9273, July 2013. Retrieved June 2015.

[Din13b] Roger Dingledine. Raise our Guard Rotation Period, if Appropriate. Ticket on *Tor Bug Tracker* https://trac.torproject.org/projects/tor/ticket/8240, February 2013. Retrieved June 2015.

[Din14] Roger Dingledine. Load Balance Right when we have Higher Guard Rotation Periods. Ticket on *Tor Bug Tracker* https://trac.torproject.org/projects/tor/ticket/9321, July 2014. Retrieved June 2015.

[DKM+06] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our Data, Ourselves: Privacy via Distributed Noise Generation. In *Advances in Cryptology-EUROCRYPT 2006*, pages 486–503. Springer, 2006.

[DM06a] Roger Dingledine and Nick Mathewson. Design of a Blocking-Resistant Anonymity System. Technical Report 2006-1, The Tor Project, November 2006.

[DM06b] Roger Dingledine and Nick Mathewson. Tor Directory Protocol, Version 3. https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=dir-spec.txt, January 2006. Retrieved May 2015.

[DMS04a] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[DMS04b] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th conference on USENIX Security Symposium-Volume 13*, pages 303–320. USENIX Association, 2004.

[DRV10] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and Differential Privacy. In *51st IEEE Symposium on Foundations of Computer Science*, pages 51–60. IEEE, 2010.

[DS08] George Danezis and Paul Syverson. Bridging and Fingerprinting: Epistemic Attacks on Route Selection. In Nikita Borisov and Ian Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies*, pages 133–150. Springer, July 2008.

[DSA+11] 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. ACM, 2011.

[DSD04] Claudia Diaz, Len Sassaman, and Evelyne Dewitte. Comparison between Two Practical Mix Designs. In *Proceedings of the 9th European Symposium on Research in Computer Security*, pages 141–159. Springer, 2004.

[Dwo06] Cynthia Dwork. Differential Privacy. In *Automata, languages and programming*, pages 1–12. Springer, 2006.

[EBA+12] Tariq Elahi, Kevin Bauer, Mashael AlSabah, Roger Dingledine, and Ian Goldberg. Changing of the Guards: A Framework for Understanding and Improving Entry Guard Selection in Tor. In *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*, pages 43–54. ACM, 2012.

[EDG14] Tariq Elahi, George Danezis, and Ian Goldberg. PrivEx: Private Collection of Traffic Statistics for Anonymous Communication Networks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1068–1079. ACM, 2014.

[ES09] Matthew Edman and Paul F. Syverson. AS-awareness in Tor Path Selection. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *Proceedings of the 2009 ACM Conference on Computer and Communications Security*, pages 380–389. ACM, 2009.

[EW11] Nahed Eltantawy and Julie Wiest. The Arab Spring | Social Media in the Egyptian Revolution: Reconsidering Resource Mobilization Theory. *International Journal of Communication*, 5(0), 2011.

[FA12] Arturo Filasto and Jacob Applebaum. OONI: Open Observatory of Network Interference. In *Proceedings of the USENIX Workshop on Free and Open Communications on the Internet*. USENIX, 2012.

[FBH+02] Nick Feamster, Magdalena Balazinska, Greg Harfst, Hari Balakrishnan, and David Karger. Infranet: Circumventing Web Censorship and Surveillance. In *Proceedings of the 11th USENIX Security Symposium*, August 2002.

[FBW+03] Nick Feamster, Magdalena Balazinska, Winston Wang, Hari Balakrishnan, and David Karger. Thwarting Web Censorship with Untrusted Messenger Delivery. In Roger Dingledine, editor, *Proceedings of Privacy Enhancing Technologies workshop*, pages 125–140. Springer-Verlag, LNCS 2760, March 2003.

[FHE+12] David Fifield, Nate Hardison, Jonathan Ellithorpe, Emily Stark, Roger Dingledine, Phil Porras, and Dan Boneh. Evading Censorship with Browser-Based Proxies. In *Proceedings of the 12th Privacy Enhancing Technologies Symposium*. Springer, July 2012.

[Fis05] Ken Fisher. The Death of SuprNova.org. *Ars Technica*, http://arstechnica.com/staff/2005/12/2153/, December 2005. Retreived Nov 2012.

[FLH+15] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. Blocking-resistant Communication through Domain Fronting. *Proceedings on Privacy Enhancing Technologies*, 2015(2):46–64, June 2015.

[FNB13] David Fifield, Gabi Nakibly, and Dan Boneh. OSS: Using Online Scanning Services for Censorship Circumvention. In *Proceedings of the 13th Privacy Enhancing Technologies Symposium*, July 2013.

[GMPS13] Vipul Goyal, Ilya Mironov, Omkant Pandey, and Amit Sahai. Accuracy-Privacy Tradeoffs for Two-Party Differentially Private Protocols. In *Advances in Cryptology–CRYPTO 2013*, pages 298–315. Springer, 2013.

[goa11] goagent. GoAgent. *Pseudonymously*, https://github.com/goagent/goagent, July 2011. Retrieved May 2015.

[Goo15] Dan Goodin. Massive denial-of-service attack on GitHub tied to Chinese government. *Ars Technica*, http://arstechnica.com/security/2015/03/massive-denial-of-service-attack-on-github-tied-to-chinese-government/, March 2015. Retrieved May 2015.

[GSH13] 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*, 2013.

[GW98] Ian Goldberg and David Wagner. TAZ Servers and the Rewebber Network: Enabling Anonymous Publishing on the World Wide Web. *First Monday*, 3(4), August 1998.

[HBS13] Amir Houmansadr, Chad Brubaker, and Vitaly Shmatikov. The Parrot is Dead: Observing Unobservable Network Communication. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, May 2013.

[HNCB11] 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*, October 2011.

[HNGJ15] Bridger Hahn, Rishab Nithyanand, Phillipa Gill, and Rob Johnson. Games Without Frontiers: Investigating Video Games as a Covert Channel. http://arxiv.org/pdf/1503.05904v2.pdf, 2015. Retrieved May 2015.

[Hop14] Nick Hopper. Implications of Switching to a Single Guard Node: Some Conclusions. Post to *[tor-dev]* mailing list https://lists.torproject.org/pipermail/tor-dev/2014-March/006563.html, March 2014. Retrieved June 2015.

[HR12] Moritz Hardt and Aaron Roth. Beating Randomized Response on Incoherent Matrices. In *44th Symposium on Theory of Computing*, pages 1255–1268. ACM, 2012.

[HRBS12] Amir Houmansadr, Thomas Riedl, Nikita Borisov, and Andrew Singer. IP over Voice-over-IP for Censorship Circumvention. *arXiv preprint arXiv:1207.2683*, 2012.

[Hsu00] Stephen Hsu. TriangleBoy. http://www.webrant.com/safeweb_site/html/www/tboy_whitepaper.html, 2000. Retrieved May 2015.

[HWS14] Amir Houmansadr, Edmund L. Wong, and Vitaly Shmatikov. No Direction Home: The True Cost of Routing Around Decoys. In *Network and Distributed System Security*. The Internet Society, 2014.

[ICA12] ICANN Security and Stability Advisory Committee. Impacts of Content Blocking via the Domain Name System. http://www.icann.org/en/groups/ssac/documents/sac-056-en.pdf, October 2012. ICANN SSAC security advisory.

[ifi13a] ifinity0. Complete Specification for Generalised PT Composition. *Pseudonymously*, https://trac.torproject.org/projects/tor/ticket/10061, October 2013. Retrieved May 2015.

[ifi13b] ifinity0. Composing Pluggable Transports. *Pseudonymously*, https://github.com/infinity0/tor-notes/blob/master/pt-compose.rst, October 2013. Retrieved May 2015.

[IKV12] Luca Invernizzi, Christopher Kruegel, and Giovanni Vigna. Message in a Bottle: Sailing Past Censorship. In *Privacy Enhancing Technologies Symposium*, 2012.

[JACF11] Nicholas Jones, Matvey Arye, Jacopo Cesareo, and Michael J. Freedman. Hiding Amongst the Clouds: A Proposal for Cloud-based Onion Routing. In *Proceedings of the USENIX Workshop on Free and Open Communications on the Internet*, August 2011.

[JBF+14] Ben Jones, Sam Burnett, Nick Feamster, Sean Donovan, Sarthak Grover, Sathya Gunasekaran, and Karim Habak. Facade: High-Throughput, Deniable Censorship Circumvention Using Web Search. In *Proceedings of the USENIX Workshop on Free and Open Communications on the Internet*. USENIX, 2014.

[JK12] Marek Jawurek and Florian Kerschbaum. Fault-tolerant privacy-preserving statistics. In *12th Privacy Enhancing Technologies Symposium*, pages 221–238. Springer, 2012.

[Jon13] JonDo Inc. JonDonym. http://anonymous-proxy-servers.net/, 2013. Retrieved May 2015.

[jra03] jrandom. Invisible Internet Project (I2P) Project Overview. *Pseudonymously*, https://geti2p.net/_static/pdf/i2p_philosophy.pdf, August 2003. Retrieved May 2015.

[JWJ+13] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries. In *Proceedings of the 20th ACM conference on Computer and Communications Security*, November 2013.

[Kad14] George Kadianakis. Implications of Switching to a Single Guard Node: Some Conclusions. Post to *[tor-dev]* mailing list https://lists.torproject.org/pipermail/tor-dev/2014-March/006458.html, March 2014. Retrieved June 2015.

[KDK11] Klaus Kursawe, George Danezis, and Markulf Kohlweiss. Privacy-Friendly Aggregation for the Smart-Grid. In *11th Privacy Enhancing Technologies Symposium*, pages 175–191. Springer, 2011.

[KEJ+11] 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 *Proceedings of the USENIX Workshop on Free and Open Communications on the Internet*, August 2011.

[KH04] Stefan Köpsell and Ulf Hillig. How to Achieve Blocking Resistance for Existing Systems Enabling Anonymous Web Surfing. In *Proceedings of the Workshop on Privacy in the Electronic Society*, October 2004.

[KHM14] George Kadianakis, Nick Hopper, and Nick Mathewson. The Move to a Single Guard Node. *Tor Proposal #236* https://gitweb.torproject.org/torspec.git/tree/proposals/236-single-guard-node.txt, March 2014. Retrieved June 2015.

[KNRS13] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing Graphs with Node Differential Privacy. In *Theory of Cryptography*, pages 457–476. Springer, 2013.

[KSM14] Sheharbano Khattak, Laurent Simon, and Steven J. Murdoch. Systemization of Pluggable Transports for Censorship Resistance. http://arxiv.org/pdf/1412.7448v1.pdf, 2014. Retrieved May 2015.

[Lew09] Andrew Lewman. Tor Partially Blocked in China. *Tor Blog*, https://blog.torproject.org/blog/tor-partially-blocked-china, September 2009. Retrieved May 2015.

[Lew11] Andrew Lewman. New Blocking Activity from Iran. *Tor Blog*, https://blog.torproject.org/blog/new-blocking-activity-iran, January 2011. Retrieved May 2015.

[Lew12] Andrew Lewman. Iran Partially Blocks Encrypted Network Traffic. *Tor Blog*, https://blog.torproject.org/blog/iran-partially-blocks-encrypted-network-traffic, February 2012. Retrieved May 2015.

[Lew14] Andrew Lewman. Thoughts and Concerns about Operation Onymous. *Tor Blog*, https://blog.torproject.org/blog/thoughts-and-concerns-about-operation-onymous, November 2014. Retrieved May 2015.

[Lip10] Helger Lipmaa. On the CCA1-Security of Elgamal and Damgård's Elgamal. In *Inscrypt 2010*, pages 18–35. Springer, 2010.

[LM11] Kirill Levchenko and Damon McCoy. Proximax: Fighting Censorship With an Adaptive System for Distribution of Open Proxies. In *Proceedings of Financial Cryptography and Data Security*, February 2011.

[LMP+12] Patrick Lincoln, Ian Mason, Phillip Porras, Vinod Yegneswaran, Zachary Weinberg, Jeroen Massar, William Allen Simpson, Paul Vixie, and Dan Boneh. Bootstrapping Communications into an Anti-Censorship System. In *Proceedings of the USENIX Workshop on Free and Open Communications on the Internet*, August 2012.

[Loe09] Karsten Loesing. Measuring the Tor Network. https://research.torproject.org/techreports/directory-requests-2009-06-25.pdf, 2009. Retrieved May 2015.

[LSH14] Shuai Li, Mike Schliep, and Nick Hopper. Facet: Streaming over Videoconferencing for Censorship Circumvention. In *Proceedings of the Workshop on Privacy in the Electronic Society*, November 2014.

[mar12] martin. China listening in on Skype - Microsoft assumes you approve. *Pseudonymously*, https://en.greatfire.org/blog/2012/dec/china-listening-skype-microsoft-assumes-you-approve, December 2012. Retrieved May 2015.

[Mat14] Nick Mathewson. Changes in version 0.2.4.23 - 2014-07-28. https://gitweb.torproject.org/tor.git/plain/ChangeLog, July 2014. Retrieved June 2015.

[MBG+08] Damon McCoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Shining Light in Dark Places: Understanding the Tor Network. In *8th Privacy Enhancing Technologies Symposium*, pages 63–76. Springer, 2008.

[McP08] Danny McPherson. When Hijacking the Internet. https://asert.arbornetworks.com/when-hijacking-the-internet/, November 2008. Retrieved May 2015.

[MD05] Steven J. Murdoch and George Danezis. Low-Cost Traffic Analysis of Tor. In *2005 IEEE Symposium on Security and Privacy*. IEEE, May 2005.

[Mir12] Ilya Mironov. On Significance of the Least Significant Bits for Differential Privacy. In *2012 ACM Conference on Computer and Communications Security*, pages 650–661. ACM, 2012.

[MLDG12] Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. SkypeMorph: Protocol Obfuscation for Tor Bridges. In *Proceedings of the 19th ACM conference on Computer and Communications Security*, October 2012.

[MMBY14] Jeroen Massar, Ian Mason, Linda Briesemeister, and Vinod Yegneswaran. JumpBox–A Seamless Browser Proxy for Tor Pluggable Transports. *Security and Privacy in Communication Networks. Springer*, pages 116–134, 2014.

[MMP+10] Andrew McGregor, Ilya Mironov, Toniann Pitassi, Omer Reingold, Kunal Talwar, and Salil Vadhan. The Limits of Two-party Differential Privacy. In *51st IEEE Symposium on Foundations of Computer Science*, pages 81–90. IEEE, 2010.

[MPRV09] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. Computational Differential Privacy. In *Advances in Cryptology-CRYPTO 2009*, pages 126–142. Springer, 2009.

[MW08] Steven J. Murdoch and Robert N. M. Watson. Metrics for Security and Performance in Low-Latency Anonymity Networks. In Nikita Borisov and Ian Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies*, pages 115–132. Springer, July 2008.

[MWD+15] Bill Marczak, Nicholas Weaver, Jakub Dalek, Roya Ensafi, David Fifield, Sarah McKune, Arn Rey, John Scott-Railton, Ronald Deibert, and Vern Paxson. China's Great Cannon. https://citizenlab.org/2015/04/chinas-great-cannon/, 2015. Retrieved May 2015.

[MZ07] Steven J. Murdoch and Piotr Zieliński. Sampled Traffic Analysis by Internet-Exchange-Level Adversaries. In Nikita Borisov and Philippe Golle, editors, *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies*. Springer, June 2007.

[MZA+13] Mohammad Hossein Manshaei, Quanyan Zhu, Tansu Alpcan, Tamer Bacşar, and Jean-Pierre Hubaux. Game Theory meets Network Security and Privacy. *ACM Computing Surveys*, 45(3):25, 2013.

[Nab13] Zubair Nabi. The Anatomy of Web Censorship in Pakistan. In *Proceedings of the USENIX Workshop on Free and Open Communications on the Internet*. USENIX, 2013.

[NFS14] Abhinav Narain, Nick Feamster, and Alex C Snoeren. Deniable Liaisons. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 525–536. ACM, 2014.

[ØS06] Lasse Øverlier and Paul Syverson. Locating Hidden Servers. In *Symposium on Security and Privacy*, pages 100–114. IEEE, May 2006.

[Pau13] Paul Farrell. History of 5-Eyes – Explainer. *The Guardian*, http://www.theguardian.com/world/2013/dec/02/history-of-5-eyes-explainer, December 2013. Retrieved May 2015.

[Per09] Mike Perry. TorFlow: Tor Network Analysis. https://research.torproject.org/techreports/torflow-2009-08-07.pdf, August 2009. Retrieved May 2015.

[Pol78] John M Pollard. Monte Carlo Methods for Index Computation (mod p). *Mathematics of computation*, 32(143):918–924, 1978.

[Pou13] Kevin Poulsen. Edward Snowden's Email Provider Shuts Down Amid Secret Court Battle. http://www.wired.com/2013/08/lavabit-snowden/, 2013. Retrieved May 2015.

[Psi] Psiphon Inc. Psiphon. https://psiphon.ca. Retrieved May 2015.

[RES+10] Sankardas Roy, Charles Ellis, Sajjan Shiva, Dipankar Dasgupta, Vivek Shandilya, and Qishi Wu. A Survey of Game Theory as Applied to Network Security. In *2010 43rd Hawaii International Conference on System Sciences*, pages 1–10. IEEE, 2010.

[Rif15] Roseann Rife. Opinion: The Chilling Reality of China's Cyberwar on Free Speech. *CNN (U.S. Edition)*, http://www.cnn.com/2015/03/24/opinions/china-internet-dissent-roseann-rife/, March 2015. Retrieved June 2015.

[San14] Sandvine. Global Internet Phenomena Report. https://www.sandvine.com/downloads/general/global-internet-phenomena/2014/2h-2014-global-internet-phenomena-report.pdf, November 2014. Retrieved May 2015.

[SCL+12] Will Scott, Raymond Cheng, Jinyang Li, Arvind Krishnamurthy, and Thomas Anderson. Blocking-Resistant Network Services using Unblock. http://unblock.cs.washington.edu/unblock.pdf, October 2012. Retrieved May 2015.

[SCR+11] Elaine Shi, T-H. Hubert Chan, Eleanor G. Rieffel, Richard Chow, and Dawn Song. Privacy-Preserving Aggregation of Time-Series Data. In *Network and Distributed System Security Symposium*, 2011.

[Ser02] Andrei Serjantov. Anonymizing Censorship Resistant Systems. In *Proceedings of the 1st International Peer To Peer Systems Workshop*, March 2002.

[SGTH12] Max Schuchard, John Geddes, Christopher Thompson, and Nicholas Hopper. Routing Around Decoys. In *Computer and Communications Security*. ACM, 2012.

[Sha71] Daniel Shanks. Class number, a theory of factorization, and genera. In *Prococeedings of the Symposium on Pure Math*, volume 20, pages 415–440, 1971.

[Sin07] Ryan Singel. Encrypted E-Mail Company Hushmail Spills to Feds. http://www.wired.com/threatlevel/2007/11/encrypted-e-mai/, 2007. Retrieved May 2015.

[SJP+11] Rob Smits, Divam Jain, Sarah Pidcock, Ian Goldberg, and Urs Hengartner. BridgeSPA: Improving Tor Bridges with Single Packet Authorization. In *Proceedings of the Workshop on Privacy in the Electronic Society*. ACM, October 2011.

[Sog11] Chris Soghoian. Enforced Community Standards for Research on Users of the Tor Anonymity Network. In *2nd Workshop on Ethics in Computer Security Research*, pages 146–153, 2011.

[TAPT14] Micheal Carl Tschantz, Sadia Afroz, Vern Paxson, and JD Tygar. On Modeling the Costs of Censorship. *arXiv preprint arXiv:1409.3211*, 2014.

[Tor10a] The Tor Project. Tor Mertics Portal: Network, Advertised Bandwidth Distribution. https://metrics.torproject.org/network.html, 2010. Retrieved May 2015.

[Tor10b] The Tor Project. Tor Mertics Portal: Users. https://metrics.torproject.org/users.html, 2010. Retrieved May 2015.

[Tor13] Yeganeh Torbati. Iranians Face New Internet Curbs Before Presidential Election. *Reuters*, http://www.reuters.com/article/2013/05/21/net-us-iran-election-internet-idUSBRE94K0ID20130521, May 2013. Retrieved May 2015.

[Tor14] The Tor Project. Configuring a Tor Relay. https://www.torproject.org/docs/tor-doc-relay.html.en, July 2014. Retrieved June 2015.

[TS14] Henry Tan and Micah Sherr. Censorship Resistance as a Side-Effect. In Bruce Christianson, James Malcolm, Vashek Matyáš, Petr Švenda, Frank Stajano, and Jonathan Anderson, editors, *Security Protocols XXII*, volume 8809 of *Lecture Notes in Computer Science*, pages 221–226. Springer International Publishing, 2014.

[VK15] Paul Vines and Tadayoshi Kohno. Rook: Using Video Games as a Low-Bandwidth Censorship Resistant Communication Platform. http://homes.cs.washington.edu/~yoshi/papers/tech-report-rook.pdf, 2015. Retrieved May 2015.

[WALS02] Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. An Analysis of the Degradation of Anonymous Protocols. In *Proceedings of the Network and Distributed Security Symposium*. IEEE, February 2002.

[WALS03] Matthew Wright, Micah Adler, Brian N. Levine, and Clay Shields. Defending Anonymous Communications Against Passive Logging Attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 28–41, 2003.

[WGN+12] Qiyan 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*, October 2012.

[Wil11] Brandon Wiley. Dust: A Blocking-Resistant Internet Transport Protocol. http://blanu.net/Dust.pdf, 2011. Retrieved May 2015.

[Wil12] Tim Wilde. Knock Knock Knockin' on Bridges' Doors. *Tor Blog*, https://blog.torproject.org/blog/knock-knock-knockin-bridges-doors, January 2012. Retrieved June 2015.

[Win13] Philipp Winter. Towards a Tor Censorship Analysis Tool. https://blog.torproject.org/category/tags/measurement, 2013. Retrieved May 2015.

[WLBH13] Qiyan Wang, Zi Lin, Nikita Borisov, and Nicholas J. Hopper. rBridge: User Reputation based Tor Bridge Distribution with Privacy Preservation. In *Network and Distributed System Security*. The Internet Society, 2013.

[WM01] Marc Waldman and David Mazières. Tangler: A Censorship-Resistant Publishing System based on Document Entanglements. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 126–135, November 2001.

[WPF13] Philipp Winter, Tobias Pulls, and Juergen Fuss. ScrambleSuit: A Polymorphic Network Protocol to Circumvent Censorship. In *Proceedings of the Workshop on Privacy in the Electronic Society*. ACM, November 2013.

[WRC00] Marc Waldman, Aviel Rubin, and Lorrie Cranor. Publius: A Robust, Tamper-Evident, Censorship-Resistant and Source-Anonymous Web Publishing System. In *Proceedings of the 9th USENIX Security Symposium*, pages 59–72, August 2000.

[WSH14] Eric Wustrow, Collen M. Swanson, and J. Alex Halderman. TapDance: End-to-Middle Anticensorship Without Flow Blocking. In *Proceedings of the 23rd USENIX conference on Security Symposium*, pages 159–174. USENIX Association, 2014.

[WWGH11] Eric Wustrow, Scott Wolchok, Ian Goldberg, and J. Alex Halderman. Telex: Anticensorship in the Network Infrastructure. In *Proceedings of the 20th USENIX Security Symposium*, August 2011.

[WWY+12] 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 19th ACM conference on Computer and Communications Security*, October 2012.

[WYH09] S. Wolchok, R. Yao, and J. Alex Halderman. Analysis of the Green Dam censorware system. *Computer Science and Engineering Division, University of Michigan*, 18, 2009.

[ZHCB13] Wenxuan Zhoun, Amir Houmansadr, Matthew Caesar, and Nikita Borisov. SWEET: Serving the Web by Exploiting Email Tunnels. *HotPETS*, 2013.

[ZPP+13] Tao Zhu, David Phipps, Adam Pridgen, Jedidiah R. Crandall, and Dan S. Wallach. The Velocity of Censorship: High-Fidelity Detection of Microblog Post Deletions. In *Proceedings of the 22nd USENIX Security Symposium*. USENIX, 2013.