# Privacy Infrastructure for Content and Communications

Anne Marissa Edmundson

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Department of
Computer Science
Adviser: Nick Feamster

June 2018

# Abstract

Citizens' privacy is coming under greater threat as an increasing number of entities can access user data. A powerful adversary, such as a nation-state, can gain access to user data using a broad range of techniques, from privately tapping wires and collecting traffic to serving warrants or subpoenas for user data. Protecting user privacy in the face of these types of activities is challenging. Existing protocol encryption such as TLS is not sufficient, since a wide range of data, from DNS lookups to server access logs, may be visible to eavesdroppers or subject to data requests. In this dissertation, I develop new techniques that demonstrate that three aspects of the existing Internet infrastructure, specifically routing, hosting, and naming, can be used to counter surveillance.

First, I study the current state of routing by measuring which countries are on the paths between users and popular websites. I then evaluate different methods for routing Internet traffic around unfavorable countries, and based on these findings, I design and implement RAN, a lightweight system that routes a client's web traffic around specified countries with no modifications to client software.

Second, I describe modifications to content hosting that prevent a powerful adversary such as a nation-state from gaining access to a user's requests for certain Web content. In today's Internet, Content Distribution Networks (CDNs) have rich information both about the content they are serving and the users who are requesting that content. Access to this type of information makes CDNs a target for requests for data about users' browsing activities. To counter this threat, I developed Oblivious CDN (OCDN), which hides from the CDN both the content it is serving and the users who are requesting that content.

In the last part of this dissertation, I explore how the naming infrastructure currently compromises client privacy by looking at conventional DNS as well as onion services. I highlight fundamental issues with both types of domain lookups, and

present Oblivious DNS (ODNS) as a new approach to protecting privacy by decoupling client identities from the domains they are looking up.

# Acknowledgements

Many people have helped make this work possible, none moreso than my advisor. I'd like to thank Nick Feamster, for his guidance and encouragement throughout the course of my studies. He taught me how to find problems grounded in reality and how to communicate my research in an impactful way. His influence will always be seen in my research tastes and methods.

I owe many thanks to my dissertation readers, Jen Rexford and Ed Felten, as well as my dissertation examiners, Prateek Mittal and Arvind Narayanan. Their advice and insightful feedback has strengthened this work immensely.

My time at Princeton has led me to work with wonderful collaborators in various research groups. I owe a particular debt to Jen Rexford, who always made time to discuss research, and who has made me a better researcher, writer, and person. I am grateful to have worked with Prateek Mittal, who showed me the value (and effort) in real-world adoption of novel solutions. I had the great fortune of having Ed Felten as an advisor for the first couple years of my studies. His support and faith in me allowed me to explore my academic interests freely.

At the Center for Information Technology Policy and in Princeton's Security & Privacy research group, I had the opportunity to collaborate across disciplines and work with some incredibly talented researchers. I am particularly thankful to those that I worked with: Anna Kornfeld-Simpson, Josh Kroll, Marcela Melara, Yixin Sun, Roya Ensafi, Paul Schmitt, Laura Roberts, and Philipp Winter.

I never would have pursued a graduate degree without early and honest advice from faculty at Berkeley and Cornell. I am particularly thankful to David Wagner, who helped me discover my passion for research. His mentorship set me on the path to graduate school. I am indebted to Cynthia Sturton for her continued support from the very start. Her advice has been invaluable. I'd also like to express gratitude to Fred Schneider, for his candid thoughts and always checking up on me.

And most importantly, I would like to thank my family—my parents, Ellen and Neil, and my sister, Lexi. There are no words to express how truly grateful I am for all of the opportunities they have given me.

The work in this dissertation was supported in part by the National Science Foundation under a CNS ANET Award (#1518882) and by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program.

To my parents and sister.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

The growth of the global Internet has led to innovative applications and solutions that it was never intended for, and consequently exposes troves of unprotected data about its users. Over the years, it has become common — and almost necessary — to conduct most of our activities via the Internet: communication, shopping, banking, education, news, e-commerce, research, and politics.

Internet users often inherently trust the structure of the Internet to keep their information private, whereas companies and organizations are typically (publicly) blamed for privacy malpractices. Unfortunately, the systems and protocols that allow the Internet to operate as it does today also allow a third party to learn information about Internet users. The Internet was originally designed for different purposes, and therefore privacy was not a priority; more recently, privacy has come to the forefront as an important issue that needs to be addressed — particularly with the growing number and type of applications the Internet is used for.

Data can be gathered from different parts of the Internet infrastructure by monitoring traffic or gaining access to stored data (such as logs or content). These methods can be used on multiple components of the Internet's architecture, allowing an attacker to learn rich information about an Internet user; they can learn a user's

traffic patterns, what content they requested, and sometimes personal information sent across the network or stored in the user's data on a provider's server.

This type of attacker has already been realized, which we discuss in more detail in the next chapter, yet there has been little work on allowing users to control who can monitor or access their data. As we explain below, the Internet infrastructure in practice today leaves a lot to be desired in terms of private communications and content — it forces the user to decide between privacy and being able to participate in online society, including all the applications that the Internet presently supports.

This dissertation presents new methods to improve the Internet infrastructure in a way that preserves user privacy, while allowing the components of the Internet to function as they currently do. The new systems we design are a first step in showing how technology may be able to shape privacy debates concerning data flows and data storage.

## 1.1 Internet Architecture & Associated Privacy Risks

The Internet consists of a variety of systems and protocols that allow users to generate and access content across the world. Here we break apart the different components that make up the Internet and describe the details of the components that are the focus of this dissertation. First, we take a look at what data Internet users may be wishing to access and how it is stored. Second, we explain the system used to identify the location of the data they are accessing. And third, we describe the protocols used to route both a user's request to the data they are accessing and the corresponding response back to the user.

**Hosting Content.** Oftentimes, people use the Internet to access information quickly and easily without concerning themselves with the information's geographic location. In this case, we assume this information is a web page or set of web pages. Each web page is stored on at least one web server; sometimes a web page is replicated across many web servers, such is the case in content distribution networks (CDNs). Additionally, a web page may contain a number of resources, such as images or scripts, that may be stored on different web servers. When a user wishes to access a web page, they are accessing content from a set of web servers potentially located anywhere around the world. Each web server is run by an operator, and each operator can learn information about the content that is hosted on his server as well as the IP address (and potentially, the identity) of the user accessing the content on his server. Even if the content is encrypted at rest (when stored on the web server), it is often decrypted by the operator before being sent (even if using TLS), allowing the operator to still learn information about both the content and the requestor.

**Naming Content.** For the Internet to work, there must be a way to locate the correct web server(s) corresponding to a user's request. This is done via the Domain Name System (DNS), which translates a domain name (`www.cs.princeton.edu`) to an IP address (128.112.136.51), and each web server has a unique IP address that allows traffic to be routed to the server. The translation of a domain name to an IP address is discussed in more depth in Chapter 5. It is important to note that DNS queries are very rarely encrypted and therefore reveal information about the client's IP address as well as what domain she is requesting. Additionally, even if DNS queries were encrypted, research has shown that a web page can be identified just based on the size and location of 3rd party resources [97].

**Routing to Content.** After learning where the requested content is located, the protocols that make up the Internet must determine the correct path of networks in

which to access the content. The Internet is a network of networks, and when content is requested, packets are sent through a set of these networks to reach the destination. There are many layers and protocols in Internet routing, but the one the that is most applicable to this work is the Border Gateway Protocol (BGP). BGP determines how to route a user's request from her local machine to the appropriate web server(s) by selecting the optimal path of networks to traverse. While BGP is still lacking an operational security mechanism (despite many proposals), security protocols have been adopted at other layers of the routing stack; for example, HTTPS has become more widespread, especially with the rise of the Let's Encrypt project [2]. While this is a major step in the right direction, there are still many sites that do not support HTTPS by default; moving a site to HTTPS is not an easy task — it requires that all third party domains also support HTTPS. Even sites that do support HTTPS can be compromised; ISPs sometimes terminate (or man-in-the-middle) TLS connections for network management purposes, and can therefore learn the content that a given client is requesting.

Routing protocols, DNS, and web servers are just a few of the necessary pieces that allow the Internet to operate; and despite the importance of these systems and protocols, they can leave an enormous amount of data at the hands of an adversary who conducts surveillance. Each component is susceptible to this type of adversary on its own, but because each component is vulnerable, the attack surface is even greater. And while proposals to secure and increase privacy of the Internet are not recent ideas, the largest obstacle in adopting secure and privacy-enhancing protocols is typically convincing all parties involved that: 1) the effort to adopting the protocol is worthwhile, 2) that the new protocol will provide useful protections, and 3) the new protocol will not significantly hinder functionality and performance. And while adopting security protocols in the infrastructure is difficult, it is, in a way, easier to quantify and justify the need for security in comparison to privacy; examples of

security protocols which have been adopted (at least partially) include DNSSEC and HTTPS.

Addressing the privacy issues of the Internet infrastructure is a challenging problem; each component — routing, hosting, naming — is susceptible to surveillance, either by maliciously monitoring traffic or by requesting data via a warrant or subpoena. Actors who have the capabilities to perform either of these activities include nation-state actors, (resolver or CDN) operators, or a combination of nation-states and operators; we detail a potential adversary more in Chapter 2.

Over time, attackers evolve, becoming more sophisticated, and having access to more and better resources. To defend against stronger adversaries, users need privacy-preserving technologies to protect their data as it traverses the Internet. In addition to evolving attackers, technology is rapidly changing — and will continue to change — and yet policy covering the Internet and privacy is not changing at a similar pace. We need to design systems, such as improvements to the Internet infrastructure, to help shape policy debates in the future.

## 1.2 Outline

The rest of this dissertation is as follows. Chapter 2 gives a primer on surveillance techniques that are addressed by this work, as well as some existing countermeasures and approaches taken by both end-users and technology companies. Additionally, we provide evidence of users' privacy expectations, in particular their level of concern about a powerful adversary that has the capabilities of conducting surveillance. Lastly, we discuss the current legal landscape as it pertains to surveillance and privacy law; we highlight past influential cases as well as ongoing debates.

We explore the current state of Internet routing at a country-level granularity in Chapter 3 as a first step in determining which countries are in the position to con-

duct surveillance on their domestic infrastructure. After describing our measurement methods for analyzing country-level paths to popular destinations, we propose two different methods that an end-user can use to control the country-level paths of their Internet traffic. Then we develop RAN, Routing Around Nation-states, a system that uses measurements to allow end-users to easily route around a given country without any changes to the underlying routing protocols.[1]

In Chapter 4, we analyze how certain hosting providers are susceptible to — and often targets of — surveillance. In particular, we look at CDNs, content distribution networks, from a privacy perspective. We design OCDN, Oblivious Content Distribution Networks, an ecosystem in which current CDNs can operate as normal, while providing protections for end-users and the CDN itself. [2]

The naming component of the Internet is designed in a way that does not support the privacy of users' data from surveillance actors; Chapter 5 discusses the privacy risks with DNS; in light of the vulnerabilities found in conventional DNS, we then introduce ODNS, Oblivious DNS, as a way to protect users from surveillance and/or data requests.[3]

We conclude and present avenues for future work in Chapter 7.

[1] This is joint work with Roya Ensafi, Nick Feamster, and Jennifer Rexford; parts of this work have been published [49, 51, 53] and presented [50, 52].

[2] This is joint work with Paul Schmitt, Nick Feamster, and Jennifer Rexford; parts of this work have been published in [55].

[3] This is joint work with Paul Schmitt, Nick Feamster, and Jennifer Rexford; parts of this work have been presented [6, 54].

# Chapter 2

# Background

In this chapter, we discuss how a third party can collect data, provide evidence that users care about an adversary learning information about them, and highlight some existing, yet insufficient countermeasures to this type of surveillance. The last part of this chapter provides a brief overview of the ongoing legal and policy issues regarding data privacy.

## 2.1 Data Capture & Collection Methods

This dissertation addresses an adversary who attempts to learn information about a user's traffic, content requests, or browsing patterns. There are two primary ways that an adversary may do so: 1) traffic collection and 2) data requests.

**Traffic Collection.** The methods used by an adversary to conduct surveillance and gather traffic information can include tapping wires or monitoring traffic. This method has been realized, as evidence by the Snowden Revelations in 2013; a joint NSA and GCHQ project called MUSCULAR involved secretly wiretapping communications links in both Yahoo and Google's internal networks [125]. This is just a singular example of surveillance, and there could be many other instances that we are not yet aware of. Recently, it has been made public that the German govern-

ment has been wiretapping one of the largest IXPs in the world, DE-CIX, since 2009 [179]. In addition to privately tapping wires, we have seen governments propose (and sometimes adopt) legislation allowing wiretapping. For example, in November 2016, Great Britain passed the Investigatory Powers Act, which allows the British government to intercept and collect a user's data without the user's knowledge [94]. Similarly, the Netherlands started enforcing the Intelligence and Security Services Act in January of 2018; this law gives the Dutch government the power to conduct un-targeted surveillance and large-scale interception of and analysis of communication [48]. While these examples showcase efforts to increase traffic collection in the West, many regimes across the globe have been, and continue to conduct surveillance; in 2013, major Indian telecom companies agreed to share real-time data on Blackberry calls and services with the Indian government [147]. In that same year, Pakistan ordered Blackberry to shut down all encrypted messaging services, and when Blackberry refused to comply, Pakistan ordered the ban of Blackberry's encryption services to Pakistani citizens [155]. China, a known censor, also monitors its citizens online activities; a recent study found that the Chinese government issues surveillance (and filtering) keywords for use on social video platforms, and found that many of the keywords are related to criticism of the government [106]. During the May 2014 coup d'etat in Thailand, researchers found highly dynamic information controls within the country—including a military order that required ISPs to monitor (and censor) the publication of online content which could lead to unrest in the country [150]. Another regime that has been increasing their surveillance (and censorship) capabilities is that of the Russian government; Russian Federal Security Services can conduct *lawful* surveillance by using an interception tool called SORM (System of Operative-Investigative Measures) [156]. Telecom operators must install SORM equipment and the Russian government does not need to supply a warrant to collect information from operators. As seen from this list of examples, despite the amount of traffic

monitoring increasing around the world, each regime conducts and addresses traffic collection and monitoring of their own citizens differently. It is also important to note that many governments also have the authority to monitor and collect traffic outside of their countries' borders, and the laws governing international traffic are often different from those governing domestic traffic [143].

**Data Requests.** An adversary can issue data requests by serving warrants or subpoenas to companies, providers, or operators that have access to user information. Oftentimes companies publish transparency reports that indicate how many data requests they receive, how many user accounts are affected by the data requests, and sometimes which government issued the request. Consistently across technology companies, the number of data requests being received and the number of accounts affected by the data requests has been increasing every year. For example, Google received about 13,000 data requests in 2011, but as of 2016 they have received about 40,000 data requests [139]. Most recently, Microsoft has published information for the first half of 2017, reporting that they received 25,367 requests, which affected 44,831 accounts/users [112]. While these are large numbers of data requests, they don't paint the full picture; these numbers do not include National Security Letters (NSLs), which are data requests that are accompanied by a gag order, requiring companies to keep the data request a secret.

## 2.2 Users' Expectations of Privacy Protection

While we have seen concrete examples of how an adversary can — and does — gain access to user data via both traffic capture/collection and data requests, building systems that preserve privacy will only be beneficial if users are willing to use them. The reasons that a user may want to use such a system are clear, she wants to keep her traffic, content requests, and browsing patterns private from third parties.

To understand if users are interested in Internet privacy and value it, we conducted a survey on users' expectations of privacy protections in their online activities.[1] We analyzed 527 survey responses submitted via an online form. For context, we asked respondents how knowledgeable they are about Internet privacy and security; about 59% said they were either an expert or highly knowledgeable, whereas 40% said they were moderately or mildly knowledgeable. Less than 1% responded by saying they knew nothing of Internet privacy and security. With this in mind, we asked respondents who they are attempting to protect themselves from online when using an anonymity system. Figure 2.1 shows the results, which highlight that the majority of respondents are concerned with a government-level adversary — the type of adversary that can conduct surveillance and issue data requests.

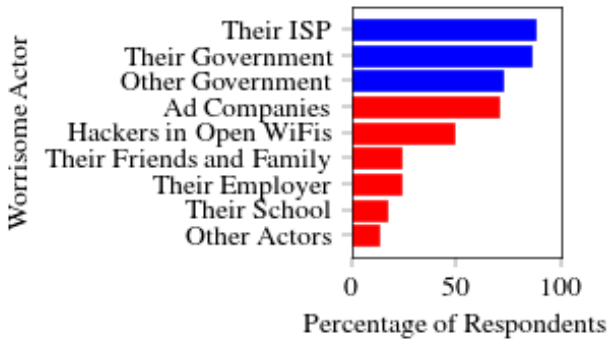



Figure 2.1: Responses from survey respondents, who were asked: "When using Tor, who do you want to protect yourself from?" The blue bars represent malicious actors that systems in this dissertation address.

In addition to the survey responses, we can see that Internet users are concerned about nation-state surveillance when they raise issues in the court systems. For example, an EU citizen, concerned about his online privacy, filed many complaints stating that Facebook is following European data protection laws, which eventually led to the nullification of the Safe Harbor agreement (an agreement made to protect EU citizens' data when transferred to the United States by U.S. companies). We

[1]This is joint work with Philipp Winter, Laura Roberts, Marshini Chetty, and Nick Feamster.

discuss this case in more detail in Section 2.4. Cases such as this give us reason to believe that improving the Internet infrastructure to increase privacy would be important to many Internet users around the world.

## 2.3 Existing Countermeasures

While encryption seems like the obvious solution to protecting the privacy of data in motion, there are many reasons why cryptography is not a panacea. Specifically, there are still a number of ways that a skilled adversary, such as a nation state, can learn information. First, the mere presence of communication between two parties may be revealing. Additionally, there has been a wide variety of research in the area of website fingerprinting, which can reveal information about the content based on size, content, and location of third-party resources. Some unencrypted communication may reveal information about the content of other (encrypted) communication; DNS traffic is extremely revealing and often unencrypted [174]. Finally, some ISPs terminate TLS connections, thereby conducting man-in-the-middle attacks on encrypted traffic for network management purposes; these purposes could be to block online video streaming (as done on Gogo's in-flight Internet [74]), or to issue copyright notices on websites (as Comcast has previously done [85]). Thus, existing encryption practices cannot protect data in motion from a nation state adversary. Additional systems are needed to conceal the nature or extent of communications in these scenarios.

One strategy is to *obfuscate* the existence of communications from the adversaries that may wish to discover it; a user could use an anonymity or censorship circumvention system. One such system, Tor, uses a series of relays and layered encryption [46]. Tor protects data in motion from surveillance, but is still susceptible to correlation and fingerprinting attacks, which allows the adversary to learn user data [65, 76, 152, 159].

Virtual Private Networks (VPNs) have also been used to circumvent censorship; a proposed system that uses VPNs is VPNGate [124]. While VPNs protect the data in motion on a portion of the path, they leave the data susceptible to snooping on the portion of the path that is not covered by the VPN. Additionally, using a VPN requires the client to trust the VPN provider, which a nation state could demand data from.

In recent years, companies have taken action to prevent a government from overreaching and requesting large amounts of user data. While companies fight data requests in court, they are now setting up separate infrastructure that provides potentially privacy-enhancing characteristics. In 2015, Microsoft set up a data center in Germany that is technically owned by a Deutsche Telekom subsidiary, T-Systems; this could potentially provide Microsoft with the plausible deniability necessary to refuse responding to a data request. As this is a new mechanism, it is not clear how this will play out in the courts when a government requests data stored in this specific data center. IBM has recently followed suit by establishing a data center (also in Germany) to give users more control over their data.

There has also been a recent push by governments to prevent surveillance conducted by foreign governments on their own citizens. For example, in light of the Snowden revelations, Brazil has taken extreme measures to avoid United States surveillance of Brazilian citizens; Brazil is laying an underwater fiber cable directly from Fortaleza, Brazil to Portugal in the hopes of avoiding traffic traversing the United States. Brazil has also replaced their government email system (which used to be Microsoft Outlook) with a home-grown system called Expresso. Germany has also made a push to keep traffic local so as to avoid foreign surveillance. They have established National Routing, which specifies that if an email's sender and receiver are located in Germany, then the email must not cross German borders. There are also a number of countries, such as Russia, that are pressuring large technology com-

panies (which are mostly located in the United States) to store their citizens' data locally (as opposed to in the United States) in hopes of avoiding U.S. surveillance.

## 2.4 Legal Battles

Policies and legal cases typically involve one of two types of data: data in motion or data at rest. *Data in motion* refers to communications or data that are en route between Internet endpoints, and users should thus be concerned about an adversary who conducts surveillance anywhere on the path to the content, such as nation-state adversaries operating in different jurisdictions (where the laws and policies may be drastically different than those governing the region where the client or data are located). *Data at rest* refers to stored data, such as the content stored on a web server. When describing the legal landscape of data privacy, we will discuss the two types of data as the policies governing them often differ.

### 2.4.1 Who gets access to stored data and does it depend on where it is located?

This question pertains specifically to stored data — *data at rest*. Our analysis of the current legal and policy decision on accessing stored data will focus on those decisions and laws made in the United States.

While some companies want to protect their customers' data from government access (such as Microsoft, mentioned in Section 2.3), not all cases side with the user. In April 2017, New York's highest court ruled that Facebook is not allowed to ask an appellate court to reject a search warrant ordering them to hand over information from hundreds of accounts [62]. Facebook's argument was that the search warrants were so broad that they were essentially an unconstitutional search. This blow to Facebook's attempt at expanding privacy protections for their users motivates the

general need for systems that preserve the privacy of data at rest. Additionally, a recently passed piece of legislation, the CLOUD Act, will allow: 1) foreign police to collect peoples' communications from U.S. companies without a U.S. warrant, 2) foreign countries to demand personal data stored in the U.S., 3) foreign police to collect data about a person without notifying the person, and 4) empower U.S. police to demand any data regardless of the person's citizenship or the data's location [144].

The past few months have also seen a proposal for a United Nations Global warrant; this would allow the issuing of international surveillance warrants or international data access warrants (IDAWs) base on international law [138]. Complementary technology can prevent overreach that results in access to broad amounts of data, thus potentially also resolving ambiguities or laws that are unfavorable towards user privacy.

**Takeaways.** Companies have been issued data requests for years, but the number of requests has increased each year, resulting in more and more users being affected by the requests. The only way companies typically fought these data requests were in the courts, which led to a variety of outcomes; more recently, companies are trying new methods of fighting data requests. The work in this dissertation is complimentary to these approaches; Chapters 4 and 5 introduce new technical methods to fight and prevent data requests.

### 2.4.2 Where can data be transferred to and from?

This question primarily targets data flows — *data in motion*. Data in motion is subject to different data protection requirements and privacy legislation based on which region the data is traversing. The history of privacy of data flows primarily shows that privacy protection laws greatly vary between jurisdictions, and that the agreements and frameworks made are still very much in flux. It is still unclear what the future of EU-US data transfer will be, but technologists have a role among these

policy makers as well; technology can be designed to protect data privacy, as well as to prevent an overreaching government from access to a different jurisdiction's data.

While the frameworks governing data privacy are being negotiated between different nation states and governing entities, they are not the only ones that are concerned about data privacy protections. In fact, it was an EU citizen who brought a case against the Safe Harbor agreement, which eventually rendered it invalid; Max Schrems, an Austrian citizen, challenged the transfer of his data to the U.S. by Facebook [148].

The Privacy Shield addresses data transfers that start in the EU and end in the US, but the law does not mention intermediate countries on the path. For example, data transferred from the EU to the US may traverse Canada en route; once traffic enters a specific country (even if neither the client nor server are located in that country), it becomes subject to that country's policies on surveillance and censorship. As a result, data in motion protections can be compromised solely because the path taken to access information traverses an unfavorable jurisdiction. These issues go far beyond EU personal data being accessed by the U.S.; recent research has shown that Internet traffic suffers "collateral damage" simply because it is routed through a specific jurisdiction [7]. This study found that Korean traffic to German web sites (.de domains) often suffers "collateral damage" in the sense that it is censored because the path traverses an Autonomous System (AS) known for censorship within China. This case is generalizable to both nation state level surveillance and censorship, and recently certain countries, such as Brazil, have taken extreme measures to avoid routing their Internet traffic through the United States for this exact reason [22].

Several ongoing debates concern the privacy of data in motion. Some governments are demanding backdoors in encrypted communication applications, and other governments are trying to issue subpoenas for encrypted communications [56, 176]. On the other side, many companies are trying to keep their clients' data (in motion)

secure; one example of this is the increased use of end-to-end encryption in communication applications. End-to-end encryption is a step in the direction of protecting privacy using technology. It has become increasingly common for governments to criticize technologies that use end-to-end encryption; for example, the United Kingdom recently stated that they should have access to WhatsApp messages when necessary (this was in response to the recent terror attacks in London) [167]. The recent actions taken by many governments, not just the UK, should motivate technologists to design systems with cross- jurisdiction data flows in mind.

**Takeaways.** The policies governing data flows, particularly data flows between the European Union and the United States have been challenged and re-written due to privacy concerns. The most common approach to maintaining privacy of data in transit is using end-to-end encryption; Chapter 3 introduces an orthogonal approach of routing data around a given region (or jurisdiction).

# Chapter 3

# Routing: Nation-State Routing for Privacy

When Internet traffic enters a country, it often becomes subject to that country's domestic laws and policies. As a result, users, ISPs, and governments need to determine—and control—which countries their traffic is traversing. Discovering which countries an end-to-end path traverses and providing mechanisms to avoid certain countries may help users avoid the practices and laws of particular countries. One motivation for avoiding a certain geographic region is to evade surveillance and other types of interference. In some cases, avoiding certain countries may also lower costs or improve performance, where technologies that certain countries use (*e.g.*, firewalls, traffic shapers) throttle network traffic speeds.

An increasing number of countries have passed laws that facilitate mass surveillance of networks within their territory [67, 92, 103, 123]. While governments and citizens alike may want to divert their Internet traffic from countries that perform surveillance (notably, the United States [36, 45, 146]), this is a challenging problem with no known, effective solutions. Additionally, both users and ISPs may wish to prevent these international detours for performance and cost reasons; previous work

has shown that tromboning paths—paths that start and end in the same country, but also traverse a foreign country—are common [78, 151].

With the increasing pervasiveness of encryption (and the efforts of Let's Encrypt [2]), Internet security is improving, but defending against large-scale surveillance activities requires not only encryption, but also mechanisms for controlling where traffic goes in the first place: end-to-end encryption conceals some information content, but it does not protect all sensitive information. First, many websites do not fully support encrypted browsing by default; a recent study showed that more than 85% of the most popular health, news, and shopping sites do not encrypt by default [174]; migrating a website to HTTPS can be challenging, and doing so requires all third-party domains on the site (including advertisers) to use HTTPS. Second, even encrypted traffic may still reveal a lot about user behavior: the presence of any communication at all may be revealing, and website fingerprinting can reveal information about content merely based on the size, content, and location of third-party resources that a client loads [97]. Recent work studying Internet of Things (IoT) devices has shown that passive network observers can learn sensitive information about users even when traffic is encrypted [9]; this highlights the risks of large-scale surveillance in the IoT ecosystem. DNS traffic is also revealing and is almost never encrypted [174]. Additionally, ISPs often terminate TLS connections, conducting man-in-the-middle attacks on encrypted traffic for network management purposes [74]. And, of course, encryption offers no solution to interference, degradation, or blocking of traffic that a country might perform on traffic that crosses its borders. Finally, a nation-state may collect and store encrypted traffic; if the encryption is defeated in the future, a nation-state may be able to discover the contents of previous communications. This has already been realized, according to documents leaked from the National Security Agency (NSA) and Government Communications Headquarters (GCHQ): "A 10-year NSA program against encryption technologies made a breakthrough in

2010 which made 'vast amounts' of data collected through internet cable taps newly 'exploitable"' [140].

In this chapter, we study two questions: (1) Which countries do *default* Internet routing paths traverse?; (2) What methods can help governments (or citizens, ISPs, etc.) better control transnational Internet paths? We *actively measure* the paths originating in twenty countries to the most popular websites in each of these respective countries. Our analysis in this chapter focuses on five countries—Brazil, Netherlands, Kenya, India, and the United States—for a variety of reasons. For example, Brazil has made a concerted effort to avoid traversing certain countries such as the United States through extensive buildout of Internet Exchange Points (IXPs) [28]. The Netherlands has one of the world's largest IXPs and relatively inexpensive hosting. Kenya is one of the most well-connected African countries, but it is still thought to rely on connectivity through Europe and North America for many destinations, even content that might otherwise be local (*e.g.*, local newspapers) [34, 63, 64, 78]. We highlight many trends that are common across all of the countries we study; we have also released detailed statistics on all twenty countries that we measure on the project website and intend to update these regularly.

In contrast to all previous work in this area, we measure router-level forwarding paths, as opposed to analyzing Border Gateway Protocol (BGP) routes [100, 151], which can provide at best only an indirect estimate of country-level paths to popular sites. Although BGP routing can offer some information about paths, it does not necessarily reflect the path that traffic actually takes, and it only provides AS-level granularity, which is often too coarse to make strong statements about which countries that traffic is traversing. In contrast, we measure routes from RIPE Atlas probes [141] in each country to the Alexa Top 1000 domains for each country; we directly measure the paths not only to the websites corresponding to themselves, but also to the sites hosting any third-party content on each of these sites.

While using direct measurements provides these benefits, there are a number of challenges associated with determining which countries a client's traffic is traversing. First, performing direct measurements is more costly than passive analysis of BGP routing tables; RIPE Atlas, in particular, limits the rate at which one can perform measurements. As a result, we had to be strategic about the origins and destinations that we selected for our study. We study twenty geographically diverse countries, focusing on countries in each region that are making active attempts to thwart transnational Internet paths. Second, IP geolocation—the process of determining the geographic location of an IP address—is notoriously challenging, particularly for IP addresses that represent Internet infrastructure, rather than end-hosts. We cope with this inaccuracy by making conservative estimates of the extent of routing detours, and by recognizing that our goal is not to pinpoint a precise location for an IP address as much as to achieve accurate reports of *significant* off-path detours to certain countries or regions. (Section 3.2 explains our method in more detail; we also explicitly highlight ambiguities in our results.) Finally, the asymmetry of Internet paths can also make it difficult to analyze the countries that traffic traverses on the reverse path from server to client; our study finds that country-level paths are often asymmetric, and, as such, our findings represent a lower bound on transnational routing detours.

We first *characterize the current state of transnational Internet routing detours* (Section 3.2). By analyzing the last hop of routing detours, we can determine where clients' are retrieving popular content from. To put the measured location of content into context, we explore hosting diversity by first measuring the Alexa Top 1000 domains and comparing the location of path endpoints to that of the Alexa Top 100 domains. We find that there is no significant difference between the results in the two domain sets, and therefore focus on the Alexa Top 100 domains *and all associated third party domains*. We find that only 45% of the Alexa Top 100 domains in Brazil are hosted in more than one country (other countries studied showed similar results);

in many cases, that country is one that clients may want to avoid. Second, even if hosting diversity can be improved, routing can still force traffic through a small collection of countries. Despite strong efforts made by some countries to ensure their traffic does not transit certain countries [21, 23–25, 88], their traffic still does so. For example, over 50% of the top domains in Brazil and India are hosted in the United States, and over 50% of the paths from the Netherlands to the top domains transit the United States. About half of Kenyan paths to the top domains traverse the United States and Great Britain (but the same half does not traverse both countries). Much of this phenomenon is due to "tromboning", whereby an Internet path starts and ends in the same country, yet transits an intermediate country; for example, about 13% of the paths that we explored from Brazil tromboned through the United States. Infrastructure building alone is not enough. ISPs in respective regions need better encouragements to interconnect with one another to ensure that local traffic stays local.

Next, we explore the extent to which clients can avoid certain countries to popular destinations (Section 3.3). We explore two techniques: using the open DNS resolver infrastructure and using overlay network relays to route Internet traffic around an unfavorable country. Our results demonstrate that these techniques can be effective for clients in certain countries; of course, the effectiveness of these approaches naturally depends on where content is hosted for that country and the diversity of Internet paths between ISPs in that country and the respective hosting sites. For example, our results show that clients in Brazil can completely avoid Spain, Italy, France, Great Britain, Argentina, and Ireland (among others), even though the default paths to many popular Brazilian sites traverse these countries. We also find that some of the most prominent surveillance states are also some of the least avoidable countries. For example, many countries depend on ISPs in the United States, a known surveillance state, for connectivity to popular sites and content. Additionally, overlay network

relays can increase performance by keeping local traffic local: by using relays in the client's country, fewer paths trombone out of the client's country.

Finally, we *design, implement, and deploy RAN, a system that allows a client to access web content while avoiding the traversal of a specified country* (Section 3.4). We implemented RAN for end-users, but ISPs can also deploy RAN proxies to provide country avoidance as a service to its customers. RAN uses a series of overlay network relays to automatically route a client's traffic around a specified country. We evaluate RAN to assess its ability to avoid certain countries, as well as the effect on end-to-end performance. We also discuss the usability and and scalability of the system. Our evaluation shows that RAN can effectively avoid many different countries and introduces minimal performance overhead.

## 3.1 State of Surveillance and Interference

We focused on traffic *originating* in five countries:

**Brazil.** Brazil is actively trying to avoid having their traffic transit the U.S. They have been building IXPs, deploying underwater cables to Europe, and pressuring large U.S. companies to host content within Brazil [20, 21, 23–27, 88]. These efforts to avoid a specific country led us to investigate whether they have been successful.

**Netherlands.** First, the Netherlands is beginning to emerge as a site where servers are located for cloud services, such as Akamai. Second, the Netherlands is where a large IXP is located (AMS-IX). Third, they are drafting a mass surveillance law [123]. Analyzing the Netherlands will allow us to see what effect AMS-IX and the emergence of cloud service hosting has had on traffic.

**Kenya.** Previous research on the interconnectivity of Africa [63,78] led us to explore the characterization of an African country's interconnectivity. We chose Kenya for three reasons: 1) it terminates many submarine cable landings; 2) it has relatively high Internet access and usage; and 3) it has more than one IXP [10, 161].

**India.** India has one of the highest number of Internet users in Asia, second only to China, which has already been well-studied [165, 172].

**United States.** We chose to study the United States because of how inexpensive it is to host domains there, the prevalence of Internet and technology companies located there, and because it is a known surveillance state.

When analyzing which countries Internet traffic *traverses*, we gave additional attention to countries that have known laws and practices inolving surveillance of or interference with Internet traffic. These countries include, the "Five Eyes" [61, 110] (the United States, Canada, United Kingdom, New Zealand, and Australia), as well as France, Germany, Poland, Hungary, Russia, Ukraine, Belarus, Kyrgyzstan, and Kazakhstan. Countries such as China, Iran, and Russia, are not only surveilling, but are also censoring, blocking, and interfering with traffic that crosses their borders. We have studied surveillance states in more detail, and that information is available in our technical report [51].

## 3.2 Characterizing Transnational Detours

In this section, we describe our measurement methods, the challenges in conducting them, and our findings concerning the transnational detours of default Internet paths.

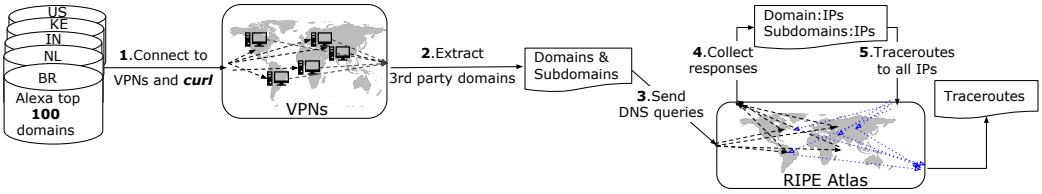



Figure 3.1: Measurement pipeline to study Internet paths from countries to popular domains.

### 3.2.1 Measurement Approach and Challenges

**Overview of approach** Figure 3.1 shows the process that we use to discover end-to-end Internet paths from our respective vantage points to various domains. We first use VPNs to establish various vantage points in the countries of interest; then, we use `curl` to download corresponding webpages for each of those popular domains, including all subdomains that are embedded in the site's top-level webpage (1,2). We extract all of these domain names (3) and resolve them to their corresponding IP addresses (4); we then perform traceroutes to each of those IP addresses (5). Figure 3.2 describes how we translate an IP-level traceroute to a country-level path. We geolocate each IP address, removing unknown hops; we then de-duplicate the country-level path. Although it is seemingly straightforward, this approach entails a number of limitations and caveats, which we describe in the rest of this section.

#### Resource Limitations

We currently focus our measurements on five countries due to resource limitations. The iPlane [118] and Center for Applied Internet Data Analysis (CAIDA) [30] projects maintain large repositories of traceroute data, neither of which are suitable for our study. iPlane has historical data as far back as 2006. Unfortunately, because iPlane uses PlanetLab [132] nodes, which are primarily hosted on the Global Research and Education Network (GREN), iPlane measurements may not be representative of typical Internet users' traffic paths [14]. CAIDA runs traceroutes from different vantage

points around the world to randomized destination IP addresses that cover all /24s; in contrast, we focus on paths to popular websites from a particular country.

We run active measurements that better represent paths of a typical Internet user. To do so, we run DNS and traceroute measurements from RIPE Atlas probes, which are hosted all around the world in many different types of networks, including home networks [141]. RIPE Atlas probes can use the local DNS resolver, which provides the router-level path to a destination that a user is likely to see in that country.

Conducting measurements from a RIPE Atlas probe costs a certain amount of "credits", which restricts the number of measurements that we can run. RIPE Atlas also imposes rate limits on the number of concurrent measurements and the number of credits that an individual user can spend per day. We address these challenges in two ways: (1) we reduce the number of measurements we must run on RIPE Atlas probes by conducting traceroute measurements to a single IP address in each /24 (as opposed to all IP addresses returned by DNS) because all IP addresses in a /24 belong to the same AS, and should therefore be located in the same geographic area; (2) we use a different method—VPN connections—to obtain a vantage point within a foreign country, which is still representative of an Internet user in that country. We are forced to use an alternative vantage point to RIPE Atlas probes because these probes do not support all operations that our methods require (such as requesting the webpage). While VPN connections provide the necessary functionality in the correct country, RIPE Atlas probes are more representative of typical Internet users, as they are often hosted in home networks, therefore we decide to use RIPE Atlas probes when possible and VPN connections when necessary.

### Path Asymmetry

The reverse path (i.e., the path from the server to the client) is just as important as (and often different from) the forward path. Previous work has shown that paths

between Internet endpoints are often asymmetric [80]. Most work on path asymmetry has been done at the AS level [68, 79, 80, 129], but not at the country level; our measurements can consider only the forward path (from client to domain or relay), not the reverse path from the domain or relay to the client.

We also (separately) measured path asymmetry at the country granularity. If country-level paths were symmetric, then the results of our measurements would be representative of the forward *and* reverse paths. If the country-level paths are asymmetric, then our measurement results only provide a lower bound on the number of countries that traffic between two endpoints may traverse. Using 100 RIPE Atlas probes and eight Amazon EC2 instances, we ran traceroute measurements from every probe to every EC2 instance and from every EC2 instance to every probe (the EC2 instances were located in the United States, Brazil, Canada, Ireland, Germany, Japan, Australia, and Singapore). After mapping the IP addresses to countries, we analyzed the paths for symmetry. First, we compared the set of countries on the forward path to the set of countries on the reverse path; we found that about 30% of the paths were symmetric at the country level. We compared the number of countries on the forward and reverse paths to determine how many reverse paths traversed a subset of the countries in the respective forward path; this situation occurred for 55% of the paths. This level of asymmetry suggests that our results are a lower bound on on how many countries a client's path traverses en route to a web site. It also suggests that while providing lower bounds on transnational detours is feasible, designing systems to *completely* prevent these detours on both forward and reverse paths is challenging. If tools that shed light on the reverse path between endpoints (*e.g.*, Reverse Traceroute [102]) see more widespread deployment, the characterizations and avoidance techniques that we develop in this work could be extended to include reverse paths.

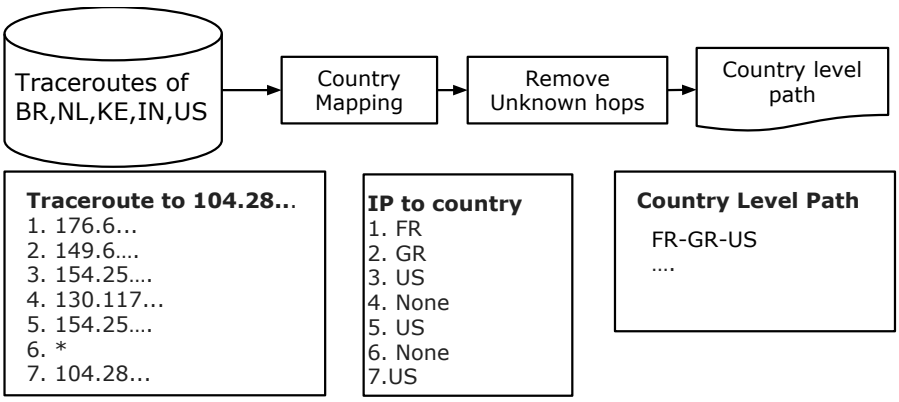



Figure 3.2: Mapping country-level paths from traceroutes.

### Traceroute Origin and Destination Selection

Each country hosts 75 to several hundred RIPE Atlas probes. Because of resource restrictions, we could not use all of the probes in each country. We selected the set of probes that had unique ASes in the country to get the widest representation of origination (starting) points.

To determine how many websites we must measure to sufficiently capture client paths to popular websites in a country, we first compare the country-level paths from a small set of vantage points to the Alexa Top 100 domains and to the Alexa Top 1000 domains. The proportion of paths that transited (and ended in) each country are similar in both cases; the paths to the top 1000 domains exhibit a longer tail of countries that transit or host content, likely because these domains are less popular and therefore hosted in more obscure locations. Otherwise, the results are similar. Figure 3.3 shows this comparison. Therefore, we used the Alexa Top 100 domains in each of the respective countries as our destinations, as well as the third-party domains that are requested as part of an original web request.

To obtain the third-party domains that are hosted on each popular website, we use `curl` to retrieve the homepage for each respective domain from within the country that is hosting the vantage point in question. RIPE Atlas probes do not support these types of Web requests; instead, we establish a VPN connection within each of these

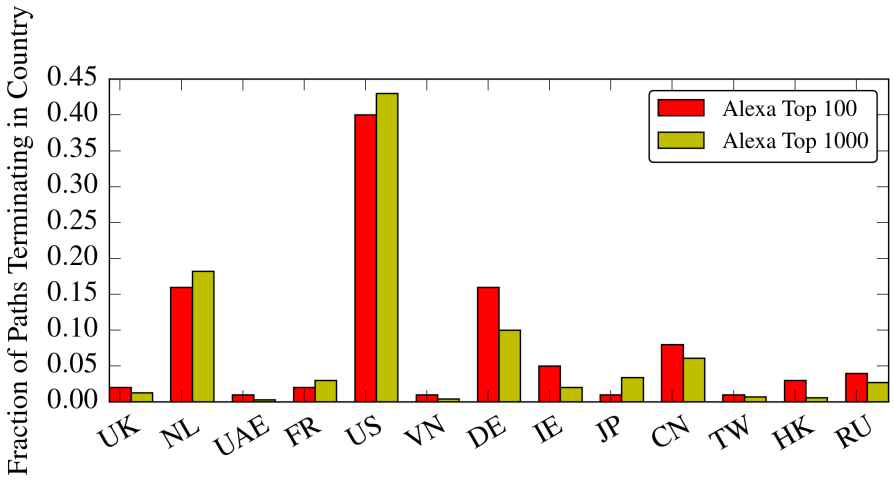



Figure 3.3: Comparison of path endpoints between the Alexa Top 100 and the Alexa Top 1000. For simplicity, we have removed the long tail of countries that are the endpoint for less than 1% of the measured paths. The countries listed on the x-axis are the countries in which paths terminate.

countries to `curl` each domain and extract the third-party domains; we `curl` from the client's location in case web sites are customizing content based on the region of the client.

### Country Mapping

Accurate IP geolocation is challenging [57, 58, 73, 77, 89, 101, 133]. We use MaxMind's geolocation service to map IP addresses to their respective countries [120], which is known to contain inaccuracies. Fortunately, our study does not require high-precision geolocation; we are more interested in providing accurate lower bounds on detours at a much coarser granularity. Fortunately, previous work has found that geolocation at a country-level granularity is more accurate than at finer granularity [91]; a recent study on geolocation accuracy found that the MaxMind database used in this work has a country-level coverage rate of 95% and a country-level accuracy rate of 79% (this is significantly better than the city-level coverage rate—53%—and city-level accuracy rate—67% [71]. In light of these concerns, we post-processed our IP to

country mapping by removing all IP addresses that resulted in a 'None' response when querying MaxMind, which causes our results to underestimate of the number of countries that paths traverse. It is important to note that removing 'None' responses will *always* tend to underestimate the set of countries in the path. Figure 3.2 shows an example of this post-processing.

#### Traceroute Accuracy and Completeness

Our study is limited by the accuracy and completeness of traceroute. Anomalies can occur in traceroute-based measurements [11], but most traceroute anomalies do not cause an overestimation in states that manipulate or monitor traffic. The incompleteness of traceroutes, where a router does not respond, causes our results to underestimate the number of states that interfere with network traffic.

#### IPv4 vs. IPv6 Connectivity

We collect and analyze only IPv4 paths. IPv6 paths likely differ from IPv4 paths as not all routers that support IPv4 also support IPv6. A comparable study of IPv6-level paths is an avenue for future work.

### 3.2.2 Results

Table 3.1 shows five of the countries that we studied along the top of the table and the countries that host their content along in each row. A "-" represents the case where no paths ended in that country. For example, the United States is the endpoint of 77.4% of the paths that originate in Brazil, and no Brazilian paths terminated in South Africa. Table 3.2 shows the fraction of paths that transit (or end in) certain countries, with a row for each country that is transited. We report on measurements conducted on January 31, 2016, and we are continuing to run these measurements and publish the data. We have published our data to a repository [42].

| *Terminating in* \ *Originating in* | Brazil | Netherlands | India | Kenya | United States |
|---|---|---|---|---|---|
| Brazil | .169 | - | - | - | - |
| Canada | .001 | .007 | .015 | .006 | - |
| United States | .774 | .454 | .629 | .443 | .969 |
| France | .001 | .022 | .009 | .023 | .001 |
| Germany | .002 | .013 | .014 | .028 | .001 |
| Great Britain | - | .019 | .021 | .032 | .002 |
| Ireland | .016 | .064 | .027 | .108 | .001 |
| Netherlands | .013 | .392 | .101 | .200 | .024 |
| Spain | .001 | - | - | - | - |
| Kenya | - | - | - | .022 | - |
| Mauritius | - | - | - | .004 | - |
| South Africa | - | - | - | .021 | - |
| United Arab Emirates | - | - | - | .011 | - |
| India | - | - | .053 | .002 | - |
| Singapore | - | .002 | .103 | .027 | - |

Table 3.1: Fraction of paths (to the Alexa Top 100 domains and associated third party domains) terminating in a country by default. The fraction in each cell represents the fraction of paths originating in the country at the top of the column and ending in the country indicated in the first cell of the same row.

**Finding 3.2.1 (Hosting Diversity)** *About half of the top domains in each of the five countries studied are hosted in a single country. The other half are located in two or more different countries.*

Hosting diversity reflects how many unique countries host a domain. The more countries host a domain, the greater the likelihood that a client can find a path to that domain that avoids a certain country. As a separate measurement experiment, we queried DNS from 26 vantage points around the world, in geographically diverse locations. We then mapped the IP addresses in the DNS responses to countries to determine how many unique countries host a domain. Figure 3.5 shows the fraction of domains in the Alexa Top 100 that are hosted in different numbers of countries; we can see two common hosting cases: (1) CDNs and (2) a single hosting country. This

shows that many domains are hosted in a single unique country, which leads us to our next analysis—where are these websites hosted, and which countries are traversed on the way to reach these locations.

**Finding 3.2.2 (Domain Hosting)** *The most common destination, regardless of originating country, is the United States.*

Table 3.1 shows the fraction of paths that are hosted in various countries. Despite the extent of country-level hosting diversity, the majority of paths from all of the countries we studied terminate in a single country; 77%, 45%, 63%, 44%, and 97% of paths originating in Brazil, Netherlands, India, Kenya, and the United States, respectively, are currently reaching content located in the United States. Our results

| *Originating in* / *Transit through* | Brazil | Netherlands | India | Kenya | United States |
|---|---|---|---|---|---|
| Brazil | 1.00 | - | - | - | - |
| Canada | .013 | .007 | .016 | .008 | .081 |
| United States | .844 | .583 | .715 | .616 | 1.00 |
| France | .059 | .102 | .104 | .221 | .104 |
| Germany | .005 | .050 | .032 | .048 | .008 |
| Great Britain | .024 | .140 | .204 | .500 | .006 |
| Ireland | .028 | .106 | .031 | .133 | .006 |
| Netherlands | .019 | 1.00 | .121 | .253 | .031 |
| Spain | .176 | .004 | - | - | - |
| Kenya | - | - | - | 1.00 | - |
| Mauritius | - | - | - | .322 | - |
| South Africa | - | - | - | .334 | - |
| United Arab Emirates | - | - | - | .152 | - |
| India | - | - | 1.00 | .058 | - |
| Singapore | - | .002 | .270 | .040 | .003 |

Table 3.2: Fraction of paths (to the Alexa Top 100 domains and associated third party domains) that a country transits by default. The fraction in each cell represents the fraction of paths originating in the country at the top of the column that transit or end in the country indicated in the first cell of the same row.

also show the Netherlands is a common hosting location for paths originating in the Netherlands, India, and Kenya.

**Finding 3.2.3 (Domestic Traffic)** *All of the countries we studied (except for the United States) host content for a small percentage of the paths that originate in their own country; they also host a small percentage of their respective country-code top-level domains.*

Only 17% of paths that originate in Brazil also end there, and only 5% and 2% of Indian and Kenyan paths, respectively, end in the originating country. For Kenya, 24 out of the Top 100 Domains are `.ke` domains, but only 5 of the 24 are hosted within Kenya. 29 out of 40 `.nl` domains are hosted in the Netherlands; four of 13 `.in` domains are hosted in India; 18 of 39 `.br` domains are hosted in Brazil. Figure 3.4 shows these results. As one might expect, all `.gov` domains were hosted in their respective country.

**Finding 3.2.4 (Transit Traffic)** *The United States and Great Britain are on more paths than any other (foreign) country.*

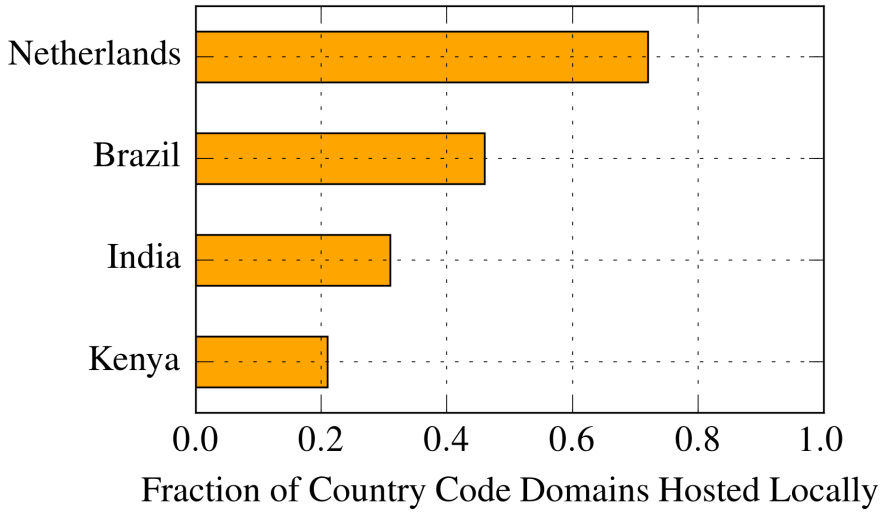



Figure 3.4: Fraction of country code top-level domains that are hosted locally. For example, 46% of .br domains are hosted in Brazil.

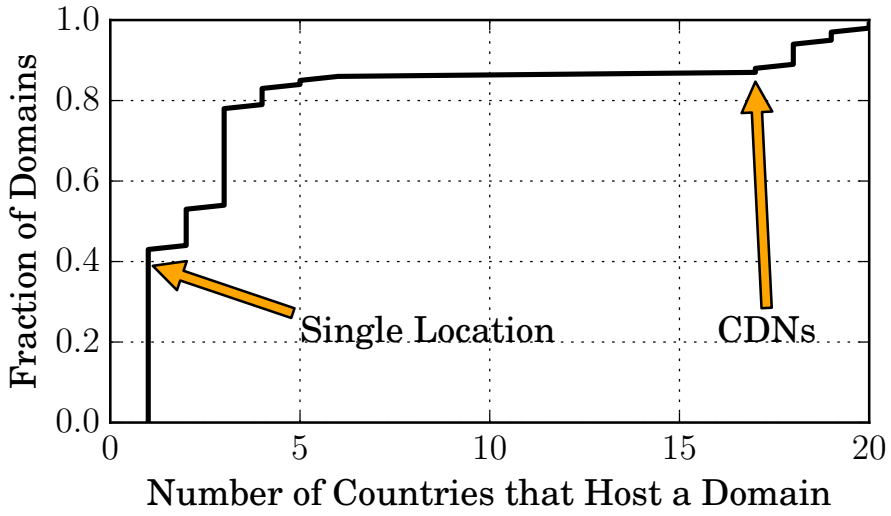
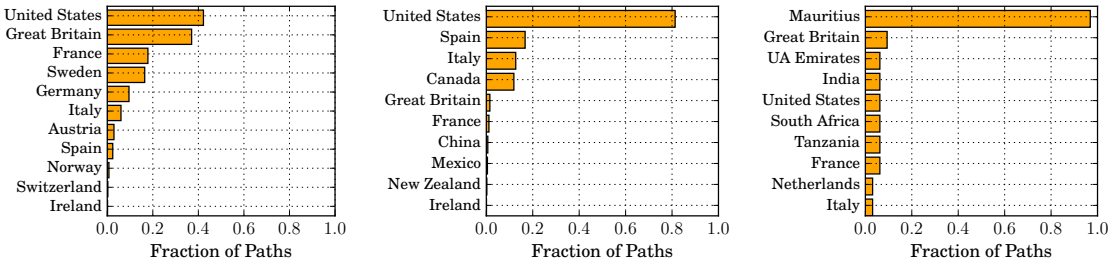



Figure 3.5: The number of Alexa Top 100 US Domains hosted in different countries.




(a) The Netherlands. (b) Brazil. (c) Kenya.

Figure 3.6: The countries that tromboning paths from the Netherlands, Brazil, and Kenya transit.

84% of Brazilian paths traverse the United States, despite Brazil's strong efforts to avoid United States surveillance [20, 21, 23–26]. Although India and Kenya are geographically distant, 72% and 62% of their paths also transit the United States.

Great Britain and the Netherlands are on many of the paths from Kenya and India: 50% and 20% of paths that originate in Kenya and India, respectively, transit Great Britain. Many paths likely traverse Great Britain and the Netherlands due to the presence of large Internet Exchange Points (*i.e.*, LINX, AMS-IX). Mauritius, South Africa, and the United Arab Emirates transit 32%, 33%, and 15% of paths

from Kenya. There are direct underwater cables from Kenya to Mauritius, and from Mauritius to South Africa [160].

**Finding 3.2.5 (Tromboning Traffic)** *Brazilian and the Netherlands paths often trombone to the United States, despite the prevalence of IXPs in both countries.*

Figure 3.6 shows the fraction of paths that trombone to different countries for the Netherlands, Brazil, and Kenya. 24% of all paths originating in the Netherlands (62% of domestic paths) trombone to a foreign country before returning to the Netherlands. Despite Brazil's strong efforts in building IXPs to keep local traffic local, many of their paths still trombone to the U.S. This is due to IXPs being seen as a threat by competing commercial providers; providers are sometimes concerned that interconnection will result in making business cheaper for competitors and stealing of customers [134].

Brazilian providers may see one another as competitors and therefore as a threat at IXPs, which causes them to peer with international providers instead of other local providers [134]. Additionally, we see Brazilian paths trombone to Spain and Italy. We see Italy often in tromboning paths because Telecom Italia Sparkle is one of the top global Internet providers [12]. MaxMind's geolocation sometimes mislabels IP addresses to be in Spain when they are actually located in Portugal. Despite our inability to disambiguate Spain and Portugal, some of the issues associated with tromboning, such as performance, are still pertinent. We are not aware of specific laws in either of these countries that would make this distinction important from a policy or legal aspect, either.

Tromboning paths that originate in Kenya most commonly traverse Mauritius, which is expected considering the submarine cables between Kenya and Mauritius. Additionally, a cable from Mombasa, Kenya to Fujairah, United Arab Emirates likely explains why many paths include these countries.

**Finding 3.2.6 (United States as an Outlier)** *The United States hosts 97% of the content that is accessed from within the United States, and only five foreign countries—France, Germany, Ireland, Great Britain, and the Netherlands—host content for the other 3% of paths.*

We find that Brazilian, Dutch, Indian, and Kenyan paths often transit the U.S. The results from studying paths that originate in the United States are drastically different from those of the other four countries. The majority of locally popular content in these countries is hosted outside of the respective country, which is shown in Table 3.1; in contrast, the United States hosts 97% of the content that is accessed from within the country. Only 13 unique countries are ever on a path from the United States to a webpage in our dataset, whereas 30, 30, 25, and 38 unique countries are seen on the paths originating in Brazil, Netherlands, India, and Kenya, respectively.

## 3.3 Feasibility of Routing Around Nation-States

We now explore the extent to which overlay networks and Internet protocols like DNS can improve path diversity and help clients route around specific countries. We explore and evaluate possible methods to (1) increase path diversity with the use of overlay nodes and (2) discover additional website replicas by diverting DNS queries through global open DNS resolvers. In this section, we develop an avoidance metric and algorithm, and evaluate the effectiveness of open resolvers and overlay nodes to avoid specific countries.

### 3.3.1 Measurement Approach

**Country Avoidance with Open Resolvers** If content is replicated on servers in different parts of the world, open DNS resolvers located around the world may be able to help clients discover a more diverse set of replicas.

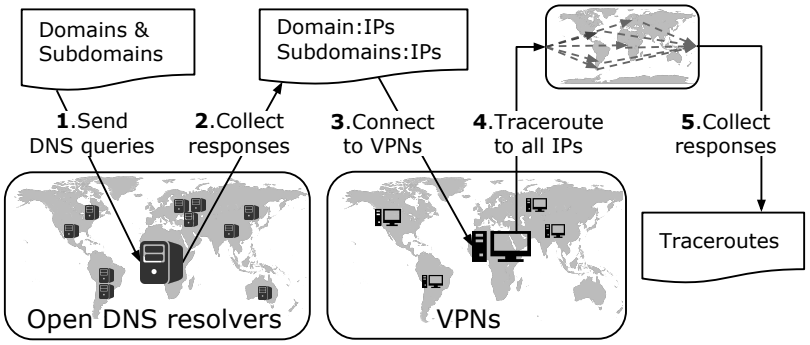



Figure 3.7: Measurement approach for country avoidance with open DNS resolvers.

We must use a different measurement approach than that described in the previous section because instead of *locally* resolving the domains, we resolve them using an open resolver. Figure 3.7 illustrates our measurement approach for this study, which utilizes open DNS resolvers located around the world [93]. These open DNS resolvers may provide different IP addresses in the DNS responses, which represent different locations of content replicas. The measurement study in Section 3.2.1 used RIPE Atlas probes to traceroute to the IP addresses in DNS response; in contrast, for this portion of the study, we initiate a VPN connection to the client's country and traceroute (through the VPN connection) to the IP addresses in the DNS responses returned by the open resolvers.

**Country Avoidance with Relays** An overlay network of relay nodes could help clients route around countries or access content that is hosted in a different country; this section performs measurements to evaluate the feasibility of such an approach. Figure 3.8 shows the steps in our measurement experiment. After selecting potential relay nodes, we perform traceroute measurements from the country of origin to each relay (1',2'), and from each relay to the set of top 100 domains in the original country (1,2,3). We then analyze these traceroutes using the approach shown in Figure 3.2 to determine the resulting country-level paths.

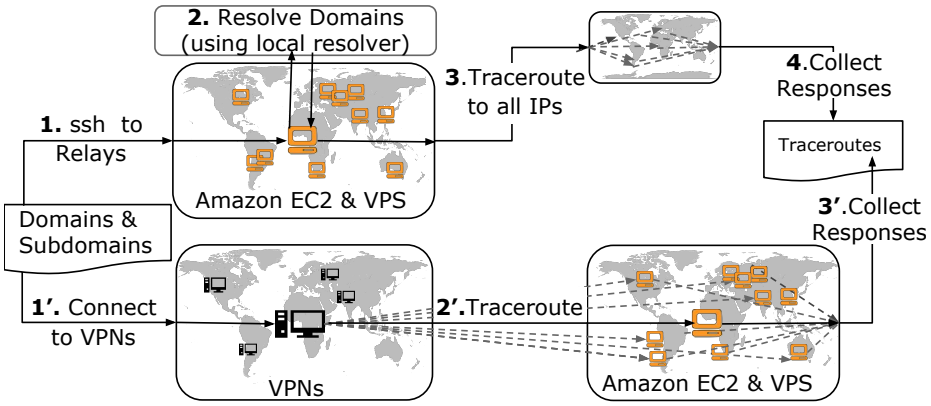



Figure 3.8: Measurement approach for country avoidance with overlay network relays.

We use eight EC2 instances, one in each geographic region (United States, Ireland, Germany, Singapore, South Korea, Japan, Australia, Brazil), as well as four Virtual Private Server (VPS) machines (France, Spain, Brazil, Singapore), which are virtual machines. Combining these two sets of machines allows us to evaluate country avoidance with a geographically diverse set of relays. By selecting an open resolver in each country that also has a relay in it we can keep the variation in measurement methods low, leading to a more accurate comparison of country avoidance methods.

### 3.3.2 Avoidability Metrics

We introduce a new metric, avoidability, to measure how often a client in one country can avoid another specific country. Using the proposed metric and algorithm, we can compare how well the different methods achieve country avoidance for any (X, Y) pair of countries.

**Avoidability metric** We introduce an avoidability metric to quantify how often traffic can avoid Country Y when it originates in Country X. Avoidability reflects the fraction of paths that originate in Country X and do not transit Country Y. We calculate this value by dividing the number of paths from Country X to domains that do not traverse Country Y by the total number of paths from Country X. The

resulting value is in the range [0,1], where 0 means the country is unavoidable for all of the domains in our study, and 1 means the client can avoid Country Y for all domains in our study. For example, there are three paths originating in Brazil: (1) $BR \rightarrow US$, (2) $BR \rightarrow CO \rightarrow None$, (3) $BR \rightarrow *** \rightarrow BR$.[1] After processing the paths as described in Section 3.2.1, the resulting paths are: (1) $BR \rightarrow US$, (2) $BR \rightarrow CO$, (3) $BR \rightarrow BR$. The avoidance value for avoiding the United States would be 2/3 because two out of the three paths do not traverse the United States. This metric represents a lower bound, because it is possible that the third path timed out ($***$) because it traversed the United States, which would make the third path: $BR \rightarrow US \rightarrow BR$, and would cause the avoidance metric to drop to 1/3.

**Avoidability algorithm with relays** Measuring the avoidability of Country Y from a client in Country X using relays entails two components: (1) Is Country Y on the path from the client in Country X to the relay? (2) Is Country Y on the path from the relay to the domain? For every domain, our algorithm checks if there exists at least one path from the client in Country X through any relay and on to the domain, and does not transit Country Y. The algorithm (Algorithm 1) produces a value in the range [0,1] that can be compared to the output of the avoidability metric.

**Avoidability algorithm with open resolvers** Recall from the measurement pipeline for avoidance with open resolvers, described in Section 3.3.1, that the resulting data are traceroutes from the client in Country X to *all* IP addresses in *all* open DNS resolver responses. To avoid a country, there must exist at least one path from the client in Country X to the domain for the client to be able to avoid Country Y when accessing the domain. The country avoidance value is the fraction of domains accessible from the client in Country X without traversing Country Y.

[1] $***$ denotes a hop in traceroute that timed out, and therefore resulted in no IP address.

**Algorithm 1** Avoidability Algorithm (with relays). This is the method to calculate the avoidability of a given country when using relays. *paths*1 is the set of country-level paths from client vantage points to relays, *paths*2 is the set of country-level paths from relays to destination domains.

```
1: function CalcAvoidance(set paths1, set paths2, string c)
2:     set suitableRelays
3:     for each (relay, path) in paths1 do
4:         if c not in path then
5:             suitableRelays ← path
6:     set accessibleDomains
7:     for each (relay, domain, path) in paths2 do
8:         if relay in suitableRelays then
9:             if c not in path then
10:                accessibleDomains ← domain
11:    D ← number of all unique domains in paths2
12:    A ← length of accessibleDomains
13:    return A/D
```

**Upper bound on avoidability** Although the avoidability metric provides a way to quantify how avoidable Country Y is for a client in Country X, some domains may be hosted only in Country Y, so the avoidance value would never reach 1.0. For this reason, we measured the *upper bound* on avoidance for a given pair of (Country X, Country Y) that represents the best case value for avoidance. This algorithm is shown in Algorithm 2; it analyzes the destinations of all domains from all relays and if there exists at least one destination for a domain that is not in Country Y, then this increases the upper bound value. An upper bound of 1.0 means that every domain that we measured is hosted (or has a replica) outside of Country Y. This value puts the avoidance values in perspective for each (Country X, Country Y) pair.

### 3.3.3 Results

We examine the effectiveness of relays for country avoidance, as well as for keeping local traffic local. Table 3.3 shows avoidance values; the top row shows the countries we studied and the left column shows the country that the client aims to avoid. Table 3.3 shows two trends: (1) the ability for a client to avoid a given Country Y increases with the use of relays; and (2) certain countries such as the United States, the United

**Algorithm 2** Avoidance Upper Bound Algorithm. This is the method used to calculate the upper bound on avoidance when using relays. For example, if a domain is solely hosted in a single country, then that country is unavoidable — this algorithm takes this case into account.

```
1: function CalcUpperbound(set relayDomainPaths, string c)
2:     zeros(domainLocations)
3:     for each (r, d, p) in relayDomainPaths do
4:         dest ← last item in p
5:         domainLocations[d] ← dest
6:     set accessibleDomains
7:     for each domain in domainLocations do
8:         if domainLocations[domain] ≠ set[c] then
9:             accessibleDomains ← domain
10:    D ← all unique domains in relayDomainPaths
11:    A ← length of accessibleDomains
12:    return A/D
```

Kingdom, and other countries that are known to perform interference on traffic are also often the most difficult countries to avoid.

| | Brazil | | | Netherlands | | | India | | | Kenya | | | United States | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Country to Avoid* | No Relay | Open Resolvers | Relays | No Relay | Open Resolvers | Relays | No Relay | Open Resolvers | Relays | No Relay | Open Resolvers | Relays | No Relay | Open Resolvers | Relays |
| Brazil | 0.00 | 0.00 | 0.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| Canada | .98 | 1.00 | 1.00 | .99 | 1.00 | 1.00 | .98 | .98 | .98 | .99 | .99 | .99 | .92 | 1.00 | 1.00 |
| United States | .15 | .19 | .62 | .41 | .57 | .63 | .28 | .45 | .65 | .38 | .55 | .40 | 0.00 | 0.00 | 0.00 |
| France | .94 | .98 | 1.00 | .89 | .96 | .99 | .89 | .98 | 1.00 | .77 | .89 | .98 | .89 | .99 | .99 |
| Germany | .99 | .99 | 1.00 | .95 | .98 | .99 | .96 | .97 | .99 | .95 | .99 | 1.00 | .99 | .99 | 1.00 |
| Great Britain | .97 | .97 | 1.00 | .86 | .87 | .99 | .79 | .79 | 1.00 | .50 | .71 | .97 | .99 | .99 | 1.00 |
| Ireland | .97 | .98 | .99 | .89 | .97 | .99 | .96 | .99 | .99 | .86 | .98 | .99 | .99 | .99 | .99 |
| Netherlands | .98 | .98 | .99 | 0.00 | 0.00 | 0.00 | .87 | .98 | .99 | .74 | .98 | .99 | .97 | .99 | .99 |
| Spain | .82 | 1.00 | 1.00 | .99 | .99 | .99 | 1.00 | .99 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| Kenya | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.00 | 0.00 | 0.00 | 1.00 | 1.00 | 1.00 |
| Mauritius | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | .67 | .97 | .99 | 1.00 | 1.00 | 1.00 |
| South Africa | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | .66 | .87 | .66 | 1.00 | 1.00 | 1.00 |
| United Arab Emirates | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | .84 | 1.00 | .99 | 1.00 | 1.00 | 1.00 |
| India | 1.00 | 1.00 | 1.00 | .99 | 1.00 | 1.00 | 0.00 | 0.00 | 0.00 | .94 | .94 | 1.00 | .99 | 1.00 | 1.00 |
| Singapore | .99 | .99 | 1.00 | .99 | .99 | 1.00 | .73 | .92 | .94 | .96 | .96 | 1.00 | .99 | .99 | 1.00 |

Table 3.3: Avoidance values for different techniques of country avoidance. The upper bound on avoidance is 1.0 in most cases, but not all. It is common for some European countries to host a domain, and therefore the upper bound is slightly lower than 1.0. The upper bound on avoidance of the United States is significantly lower than the upper bound on avoidance for any other country; .886, .790, .844, and .765 are the upper bounds on avoidance of the United States for paths originating in Brazil, Netherlands, India, and Kenya, respectively.

### Avoidability with Open Resolvers

A given country is more avoidable (higher avoidance value) when open resolvers are used as a tool for country avoidance.

**Finding 3.3.1 (Open Resolver Effectiveness)** *Using open DNS resolvers for country avoidance achieves more country avoidance than using local resolvers and no better avoidance than using relays for clients in most countries.*

For Brazilian paths, open resolvers only achieve 4% more avoidability than using local resolvers when avoiding the United States, whereas relays achieve 47% more avoidance. On the other hand, open resolvers are about as effective as relays are for avoidance for paths originating in the United States.

### Avoidability with Relays

**Finding 3.3.2 (Relay Effectiveness)** *For 84% of the (Country X, Country Y) pairs shown in Table 3.3 the avoidance with relays reaches the upper bound on avoidance.*

In almost every (Country X, Country Y) pair, where Country X is the client's country (Brazil, Netherlands, India, Kenya, or the United States) and Country Y is the country to avoid, the use of an overlay network makes Country Y more avoidable than the default routes. The one exception we encountered is when a client is located in Kenya and wants to avoid South Africa, where, as mentioned, all paths through our relays exit Kenya via South Africa.

**Finding 3.3.3 (Relays Achieve Upper Bound)** *Clients in the U.S. can achieve the upper bound of avoidance for all countries—relays help clients in the U.S. avoid all other Country Y unless the domain is hosted in Country Y.*

Relays are most effective for clients in the United States. On the other hand, it is much more rare for (Kenya, Country Y) pairs to achieve the upper bound, showing that it is more difficult for Kenyan clients to avoid a given country. This is not to say that relays are not effective for clients in Kenya; for example, the default routes to the top 100 domains for Kenyans avoid Great Britain 50% of the time, but with relays this percentage increases to about 97% of the time, and the upper bound is about 98%.

**Finding 3.3.4 (U.S. is Least Avoidable)** *The ability for any country to avoid the U.S. is significantly lower than its ability to avoid any other country in all three situations: without relays, with relays, and the upper bound.*

Despite increasing the ability to avoid the U.S., relays are less effective at avoiding the U.S. compared to all other Country Y. Clients in India can avoid the U.S. more often than clients in Brazil, Netherlands, and Kenya, by avoiding the U.S. for 65% of paths. Even using relays, Kenyan clients can only avoid the U.S. 40% of the time. Additionally, the upper bound for avoiding the U.S. is significantly lower in comparison to other countries.

**Finding 3.3.5 (Keeping Local Traffic Local)** *Using relays decreased both the number of tromboning paths, and the number of countries involved in tromboning paths.*

Where there were relays located in one of the five countries that we studied, we evaluated how well the relays kept local traffic local. This evaluation was possible for the U.S. and Brazil. Tromboning Brazilian paths decreased from 13.2% without relays to 9.7% with relays; when relays are used, all tromboning paths go only to the U.S. With the relays, we see only 1.3% tromboning paths for a U.S. client, compared to 11.2% without relays. The 1.2% of paths that trombones from the U.S. traverse Ireland.

### Comparing Avoidance Techniques

From the results shown in Table 3.3, we can see that using open DNS resolvers for country avoidance is, for the most part, less effective than using overlay network relays. Only 4% of the (origin country, country to avoid)-pairs shown in the table have a higher avoidance value when using open resolvers in comparison to overlay network relays. For this reason, we design and implement our system, RAN, solely using overlay network relays (and not open DNS resolvers). These few instances where relays were less effective could be remedied by increasing the number and/or geographic diversity of the relays, resulting in the open resolvers providing no additional avoidance after the relays. We discuss the system and the implementation of the relays in further detail in the next few sections.

## 3.4 RAN: Routing Around Nation-States

From our experience conducting measurement studies of Internet paths, we have identified a number of obstacles standing in the way of building such systems. These primarily include a lack of possible measurement methods to learn reverse paths—these are crucial because paths are asymmetric even at the country level—and a lack of knowledge about the locations in which content is replicated.

We can surmount many of these obstacles if content providers contributed to the surveillance avoidance system. To address the issue of path asymmetry, the reverse path could be measured from within the provider and used to determine if an unfavorable country is on the reverse path; this could be used in conjunction with our measurements of the forward path. In addition, content providers could strategically publish DNS records such that when a client receives a DNS response, it is for a content replica that allows her to avoid a given country. A content provider could

also replicate content in specific regions to allow clients to access replicas without traversing a specific country.

We take an approach at designing a system that is a first to route traffic around a given country *without* the help of providers. Because the design does not assume any cooperation from content providers, the system does not (and cannot) always ensure that some Internet path avoids a particular country.

### 3.4.1 Threat Model

RAN addresses an adversary who is restricted to a specific region of the world. The adversary can be passive, and conduct surveillance, or active, and interfere with traffic. We realize that a country's surveillance capabilities are not limited to the infrastructure within its borders, but a country typically can only interfere and manipulate traffic within its borders. For the purposes of this system, we assume the adversary can only view and manipulate traffic within its borders.

An adversary who taps routers around the world, splices undersea fiber cables, or participates in surveillance in foreign states is out of the scope of this work; while RAN does not address this type of attacker, RAN does protect against an attacker whose interference and monitoring capabilities are limited to a specific land mass.

### 3.4.2 Design Goals

Our measurement results motivate the design and implementation of a relay-based avoidance system, RAN, with the following design goals.

- **Country Avoidance.** The primary goal of RAN is to avoid a given country when accessing web content. RAN should provide clients a way to route around a specified country when accessing a domain. This calls for the role of

measurement in the system design and systematizing the measurement methods discussed earlier in the paper.

- **Usability.** RAN should require as little effort as possible from clients. Clients should not have to download or install software, collect any measurements, or understand how the system works. This requires a way for clients to automatically and seamlessly multiplex between relays (proxies) based on different destinations. RAN uses a Proxy Autoconfiguration (PAC) file to support this function. PAC files are supported on many types of devices, including mobile devices (smartphones, tablets, etc.). Additionally, this is a mechanism that is already being used in systems and tools. Many Internet users that use a VPN have *already* used a PAC file; when a user establishes a VPN connection, his device's proxy settings are modified to point to a PAC file.

- **Scalability.** This country avoidance system should be able to scale to large numbers of users. Therefore, RAN should be able to handle the addition of relays, as well as be cost-effective in terms of resources required. This requires clever measurement vantage points, such that each vantage point is representative of more than one client. The PAC file allows RAN to grow with the number of clients and also supports incremental deployment.

- **Non-goals.** There are some challenges that RAN does not attempt to solve; in particular, it does not provide anonymity; it routes around countries, but it does not attempt to keep users anonymous in the event that traffic can be observed. RAN also does not address domestic interference or surveillance. For example, a client in the United States cannot use RAN to avoid network interference by the United States.

### 3.4.3 Overview

RAN comprises (1) an overlay network of relays; and (2) an oracle that directs clients to the appropriate relays, as shown in Figure 3.9. RAN's relays are TCP proxy servers that allow clients to access web content without installing custom software. RAN uses the measurement methods described in Section 3.3 to learn paths between clients, relays, and domains; these results are stored at the oracle, which uses the data to decide which relay a client in some location should use for accessing a certain domain while avoiding a certain country. The oracle periodically computes paths for many combinations of client AS, destination, and country. A client can then query the oracle to determine the appropriate relay to use to avoid a certain country en route to a particular destination.

After describing our threat model and enumerating our design goals for RAN, we explain each component of the system in more detail.

### 3.4.4 Periodic Path Measurement

RAN measures all paths using `traceroute`, which is then mapped to the country level using the same methods as described in Section 3.2 and shown in Figure 3.2. The paths we measure are the: forward paths from the client to each relay; forward paths from each relay to each domain; forward paths from the client to each domain; and reverse paths from each relay to the client. The portion of the reverse path from the domains to the relays is challenging to measure due to a lack of vantage points in ASes of common destinations. As discussed in Section 3.2.1, we found that the forward and reverse paths are asymmetric at the country level, and therefore RAN cannot make any guarantees about which countries are on the path between domains and relays even though it has calculated the paths from relays to domains. Despite the lack of knowledge about this part of the reverse path, we can reason about possible scenarios. If the client's traffic is encrypted, then a country on this part of the reverse

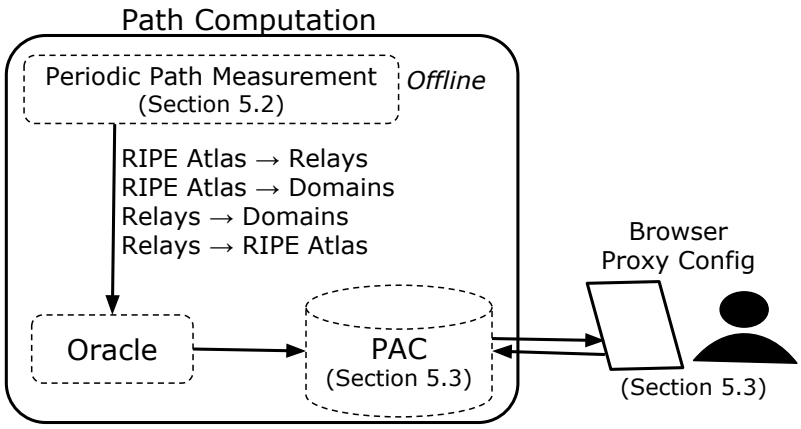



Figure 3.9: RAN architecture.

path that the client wishes to avoid cannot perform any traffic correlation attacks or website fingerprinting attacks, as the country cannot see who the client is (necessary for website fingerprinting) and does not have access to more than one part of the path (necessary for traffic correlation attacks).

**Client-to-Relay Paths.** To avoid requiring the client to install custom software, RAN measures client-to-relay paths from RIPE Atlas probes that serve as vantage points for the ASes where RAN clients might be. RAN selects probes that are geographically close the client (*e.g.*, in the same country). The oracle triggers the probe to run traceroutes to each relay. After collecting the responses, the oracle maps the IP-level paths to country-level paths and stores the results.

**Relay-to-Client Paths.** The RAN relays perform traceroutes to the IP addresses of RIPE Atlas probes, which represent client ASes. They then derive country-level paths; the oracle learns these paths from each relay.

**Relay-to-Server Paths.** Relays perform traceroutes to each domain. As with paths to clients, relays derive country-level paths and send them to the oracle.

**Client-to-Server Paths.** In case a path from a client to a domain does not pass through the country specified to avoid *by default*, then none of the proxies should be used. These paths are measured using the RIPE Atlas probes in similar locations as the clients, and the oracle triggers traceroutes from each of them to each of the domains. Corresponding country-level paths are stored at the oracle.

RAN must recompute these paths as they change. To our knowledge, there has not been any previous work on how often country-level paths change; prior work has explored how often AS-level paths change. We measured the country-level paths from a RIPE Atlas probe to the Alexa Top 100 domains once per day for a month to see how stable country-level paths are. Across the measured domains, we found the average time between path changes to be about five days. Therefore, RAN re-computes the paths every five days to incorporate the most recent country-level paths; note that the time between path re-computations is partially due to resource constraints on the RIPE Atlas network (as described in Section 3.2.1), but with more resources paths can be re-computed more often.

### 3.4.5 PAC File Generation

The oracle follows four steps to decide which relay a client should use to access a specific domain: (1) If the default path from the client to the domain does not pass through the specified country, then do not use any of the relays. (2) Otherwise, for all the paths from the client to the relays, select suitable relays, which are relays where the country to avoid is not on the forward or reverse path between the client and relay. (3) From this set, if there is a path from a suitable relay to the domain that does not include the specified country, then use that relay for that domain. (4) If there is no path from the client through any of the relays to the domain that does not pass through the specified country, then select the relay that provides the most avoidance (measured by how many other domains that avoid the specified country). The oracle

Configuration 3.1: Example PAC file.

```
function FindProxyForURL(url, host){
    if ((shExpMatch(host, "*.google.com")))
        return "PROXY 1.2.3.4:3128";
    if ((shExpMatch(host, "*.twitter.com")))
        return "PROXY 5.6.7.8:3128";
    return "DIRECT";
}
```

applies this decision process to each domain, which results in a mapping of domains to relays that can be used to avoid the given country. To facilitate automatic multiplexing between relays, RAN utilizes Proxy Autoconfiguration (PAC) files, which define how browsers should choose a proxy when fetching a URL. In the example PAC file in Configuration 3.1, proxy 1.2.3.4:3128 should be used when accessing `www.google.com`, but proxy 5.6.7.8:3128 should be used when accessing `www.twitter.com`. The oracle uses the mapping of domains to relays to generate a PAC file, which specifies which domains should be accessed through which proxy. The PAC file is published online to a URL of the format <client_country>_<country_to_avoid>_pac.pac. The client uses this URL to specify their proxy configuration. Paths are re-computed every five days, so the contents of the PAC file are also updated every five days.

### 3.4.6 Extending RAN with Content Provider Support

While the current version of RAN does not include support by content providers, it can be easily, incrementally, extended to do so without any fundamental changes to the system. This would simply require providers (or a CDN) to collect and share traceroute data from their server locations to different client and proxy locations; RAN would then convert the traceroute data to country level paths and incorporate them into the calculation of the PAC files. For content hosted in public clouds, we could set up our own VM in those same data centers and have RAN collect the reverse path traceroute data to use when creating the PAC files.

Content provider participation beyond gathering traceroute data would be even more beneficial and would lead to more extensive changes to RAN. Some of the help that content providers and CDNs could provide include publishing domain names that embed information about which country to avoid, strategically publish DNS records such that clients can take advantage of open DNS resolvers, and replicating content in diverse geographic locations.

## 3.5 Implementation and Deployment

Our implementation of RAN includes relays, an oracle, and a client. RAN is open source and written in Python; the oracle is written in just 175 lines of code and the relay is written in just under 200 lines of code. RAN is currently deployed globally, and any user may use it today. We have released an anonymized source code repository, complete with usage instructions [136].

We assume that users and machines that operate relays are trustworthy, and therefore the system runs securely. This implementation of RAN allows a client to avoid a single country at a time; attacks on RAN, such as Denial of Service attacks and targetted surveillance of the relays, are outside the scope of this work.

**Relays.** The current deployment has ten relays, one in each of the following countries: Brazil, Germany, Singapore, Japan, Australia, France, United States, United Kingdom, Netherlands, and Canada; Figure 3.10 shows these relay locations, along with their corresponding ASes. These relays operate as Ubuntu Virtual Private Servers (VPSes) with Squid as the proxy server and the RAN Relay software.

**Oracle.** The oracle software runs on a Fujitsu RX200 S8 server with dual, eight-core 2.8 GHz Intel Xeon E5 2680 v2 processors with 256GB RAM running RedHat Linux.

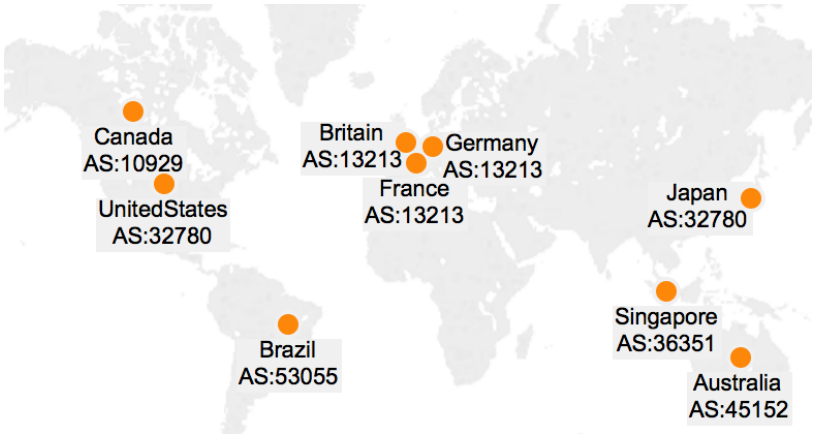



Figure 3.10: The locations and ASNs for RAN relays.

**Client.** To evaluate the RAN deployment, we set up a client machine in the Netherlands, which simply accesses web content and uses the PAC file generated by the oracle.

### 3.5.1 Other Considerations

**Adding relays and oracles.** To add a relay, the system operator must set up a machine as a proxy server, install the relay software, and update the oracle's list of relays. From that point onward, paths will be computed to and from the new relay, and clients will begin using the new proxy. Adding an oracle requires installing the oracle software on a different machine, and specifying the client locations handled by that oracle (*e.g.*, one oracle handles clients in North America and Europe, and another handles clients elsewhere). Both oracles will publish the PAC files to the same server, which causes no changes for the client.

**Failed relays and oracles.** Unresponsive relays are handled by the PAC file. The PAC file allows the oracle to specify multiple proxies in a sequential order, such that if the first proxy fails, then the client uses the second proxy (and so on). This feature can be used to specify all of the relays that have a path to the domain. Among other mechanisms, we can detect a failed oracle by determining that its PAC file is older

than one hour. Detecting a failed oracle could trigger a backup oracle to re-compute the PAC files periodically. Because oracles are stateless, failover is straightforward. Without backup oracles, clients can still use the system when the oracle fails. The clients will simply be using stale paths, which are likely (but not guaranteed) to be functional, since country-level paths change infrequently.

## 3.6 Evaluation

We evaluate RAN's ability to avoid a given country, its performance, and its storage and measurement costs.

### 3.6.1 Country Avoidability

We measured RAN's effectiveness in achieving country avoidance. We did so by first calculating the number of *default* paths that avoid a given country. Then we added a single relay, and calculated how many domains the client could access without traversing through the given country. We repeated this approach for the remaining relays. We conducted the evaluation under the condition that the client wished to avoid different countries when accessing the Netherlands top 100 domains; Figure 3.11 shows these results. Each line represents the fraction of domains accessible while avoiding the country that the line represents. For example, 46% of domains are accessible without traversing the U.S. when RAN is not being used (zero relays), and if RAN is used, then 63% of domains are accessible without traversing the U.S.

RAN helps a client avoid a foreign country, as the fraction of domains accessible without traversing the specified country without RAN is lower than with RAN. Additionally, adding the first relay provides the greatest benefit, while subsequent relays offer diminishing returns. Figure 3.11 clearly shows that avoiding the U.S. is much more difficult (or impossible) than any other country. Only 63% of domains can be

accessed while avoiding the U.S., whereas almost all domains can be accessed while avoiding any other given country.

It is important to note that RAN cannot guarantee that a country is avoided because for some domains, the path must go through the unfavorable country, as evidenced by our results for avoiding the United States. Despite this lack of guarantees, the system reduces the number of requests that transit the unfavorable country; additionally, the client can learn which domains are not accessible without passing through the unfavorable country, and can then decide whether or not to fetch that page.

### 3.6.2 Performance

To measure the performance of RAN, we measure both the throughput and latency.

To measure throughput, we ran `wget` for each of the top 100 domains from the client machine in the Netherlands using an oracle-generated PAC file. Because different relays could have been used to avoid a single domain, the oracle selected a

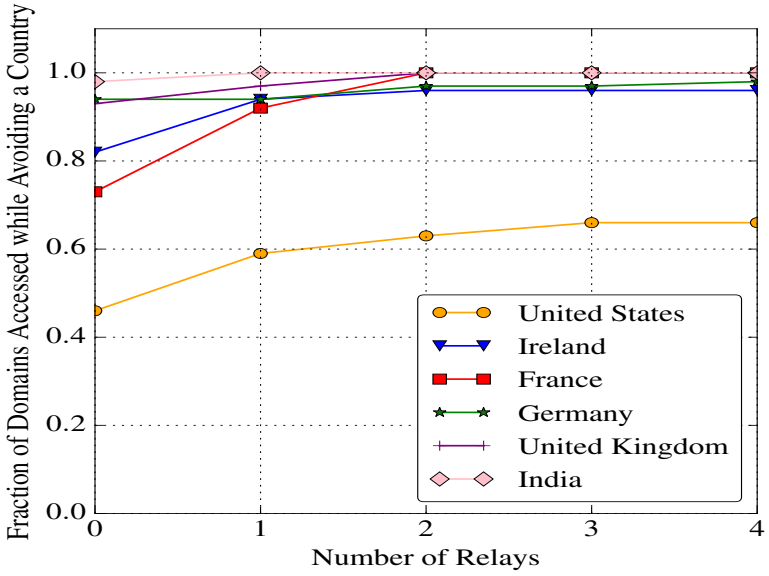



Figure 3.11: The effect of the number of relays on avoidance, for a client in the Netherlands. We tested RAN with up to nine relays.

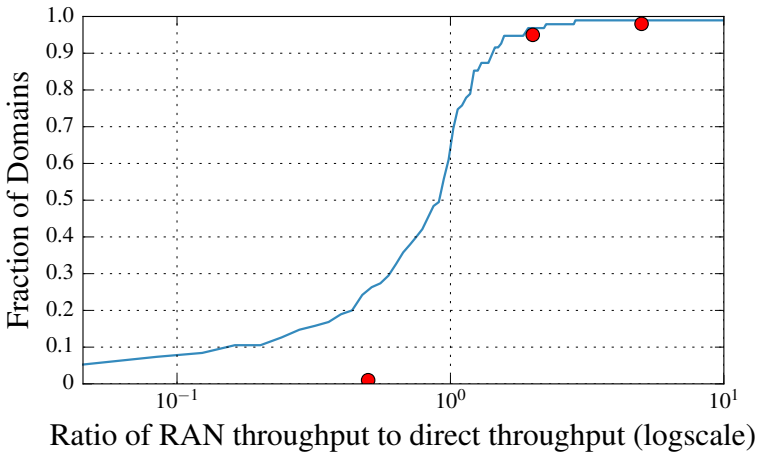



Figure 3.12: The ratio of RAN throughput to direct throughput. The points on the graph show measurements from the Resilient Overlay Networks (RON [5]) system and thus represent the performance of overlay network that is solely designed to improve reliability.

random relay from those that would allow the client to avoid the country. The oracle generated ten PAC files for a client in the Netherlands who wishes to avoid the United States, randomly selecting a relay for domains that could have used different relays, and `wget` was used for the top 100 domains for each PAC file generated. Based on the `wget` output, we calculate the number of seconds to access content using our system and take the average across the ten experiments.

Figure 3.12 shows a CDF of the ratio of RAN throughput to direct throughput. The throughput of RAN is not significantly worse than that of default paths. In some cases the performance of RAN is *better* than that of default paths. Such improvements could be a result of the relays keeping local traffic local, or due to a closer content replica being selected. These results show that RAN's performance is comparable to the performance of accessing domains without RAN. Figure 3.12 also compares RAN's throughput to RON's throughput, illustrated with the red dots; these data points are taken directly from the RON paper [5]. RAN performs worse than RON ($x < 1$), which is expected, as the detours that RAN introduces inherently inflate

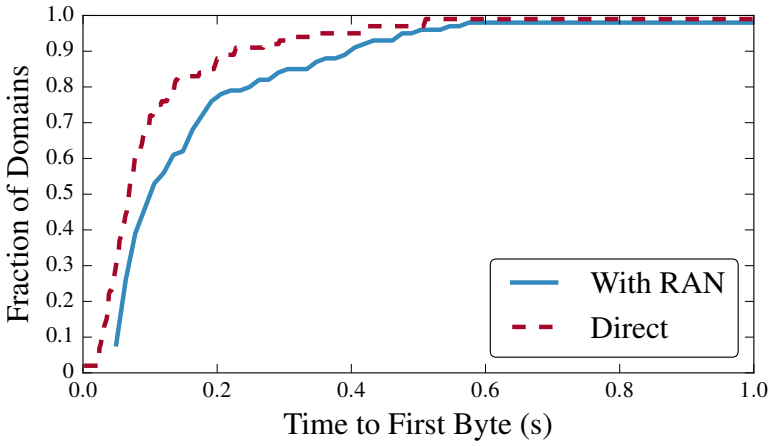



Figure 3.13: Time to First Byte for RAN and direct paths.

paths. Interestingly, both RON and RAN improve throughput for a similar fraction of samples ($x > 1$).

To measure the latency of RAN, we ran `curl` to each of the top 100 domains from the client in the Netherlands, using the ten oracle-generated PAC files to allow the client to select the appropriate relays. This experiment allowed us to measure the time to first byte (TTFB) for web downloads; we found the average TTFB when accessing content using RAN and found the TTFB when using direct paths; Figure 4.10 shows these results. The median TTFB for direct paths is 68.5 ms; for RAN paths the median is 100.8 ms; 90th percentile TTFB is 22.5 ms and 40.4 ms, respectively.

### 3.6.3 Storage and Measurement Costs

As the number of clients increases, and hence the number of paths being computed increases, the amount of storage must remain reasonable. The storage used by paths can be calculated as $DR + 2CR + CD$ where $D$ is the number of domains; $R$ is the number of relays; and $C$ is the number of ASes from which RAN measures. The storage required for a single client, 100 domains, and nine relays is 480 KB. Because there is a single PAC file for all clients in a country, $C$ will grow much slower than

if there was a different PAC file for each individual client. There are 196 countries; if RAN computed paths and a PAC file for each country, with 100 domains, and three relays required storage would be only 94 MB, making it feasible to increase the number of relays and domains.

RIPE Atlas credits are also a limited resource. Cost is proportional to $C \cdot (R + D)$. Each traceroute costs 60 RIPE Atlas credits, so one set of measurements for one client, 100 domains, and nine relays costs 6,180 credits; because these paths are updated each hour, then the daily credit cost is 148,320 credits. In return for hosting a RIPE Atlas probe, we earn 216,000 credits per day, which will support our existing prototype. To provide for more clients, more domains, or more resources, we can tune the system to re-compute paths less frequently, as we discuss in Section 4.8.

## 3.7 Discussion

**Avoiding multiple countries** We have studied only the extent to which Internet paths can be engineered to avoid a single country. Yet, avoiding a single country may force an Internet path into *other* unfavorable jurisdictions. Future work should explore the feasibility of avoiding multiple countries or perhaps even entire regions.

**Evolution over time** Our study is based on a snapshot of paths. Over time, paths change, hosting locations change, IXPs are built, submarine cables are laid, and the countries conducting network interference change. We are continuing to collect the measurements that we have presented in this paper to facilitate future exploration of how these characteristics evolve over time.

**Isolating DNS diversity vs. path diversity** In our experiments, the overlay network relays perform DNS lookups from geographically diverse locations, which provides some level of DNS diversity in addition to the path diversity that the relays

inherently provide. This approach somewhat conflates the benefits of DNS diversity with the benefits of path diversity and in practice may increase clients' vulnerability to surveillance, since each relay is performing DNS lookups on each client's behalf. We plan to conduct additional experiments where the client relies on its local DNS resolver to map domains to IP addresses, as opposed to relying on the relays for both DNS resolution and routing diversity.

**ISPs controlling country avoidance** Future work includes modifying RAN to be implemented within an ISP. Adding country avoidance functionality within ISPs (government-controlled or otherwise) allows ISPs to provide this as a transparent service to customers. A government that wishes to control which countries its citizens' traffic is traversing might deploy RAN in the country's ISPs.

**Additional RAN features** The oracle could add additional steps in the decision chain introduced in Section 3.4.5 that take into account relay and path loads. For example, if multiple relays provide a path to a domain that does not traverse the specified country, then the decision between the suitable proxies could be determined based on current relay load or performance. Our current implementation of RAN re-computes all paths once per five days; we could only re-compute paths when necessary. For example, a BGP monitoring system detect routing changes and trigger path measurements.

## 3.8 Related Work

**Nation-state routing analysis** Shah and Papadopoulos recently measured international routing detours, specifically, tromboning paths—paths that originate in one country, cross international borders, and then return to the original country—using public Border Gateway Protocol (BGP) routing tables [151]. The study dis-

covered 2 million detours each month out of 7 billion paths. Our work differs by *actively* measuring Internet paths using traceroute, yielding a more precise (and accurate) measurement of the paths, as opposed to analyzing BGP routes. Obar and Clement analyzed traceroutes that started and ended in Canada, but tromboned through the United States, and argued that this is a violation of Canadian network sovereignty [126]. Karlin *et al.* developed a framework for country-level routing analysis to study how much influence each country has over interdomain routing [100]. Their work measures country centrality using BGP routes and AS-path inference; in contrast, our work uses active measurements and measures avoidability of a given country.

**Mapping national Internet topologies** Roberts *et al.* developed a method for mapping national networks and identifying ASes that act as points of control [142]. Several studies have also characterized network paths *within* a country, including Germany [170,171] and China [185], or a country's interconnectivity [18,63,78]; these studies focus on paths within a country, as opposed to paths that traverse multiple countries.

**Routing overlays and Internet architectures** Alibi Routing uses round-trip times to prove that that a client's packets did not traverse a forbidden country or region [114, 184]; RAN differs by measuring which countries a client's packets traverse. Our work then uses active measurements to determine the best path for a client wishing to connect to a server. RON, Resilient Overlay Network, is an overlay network that routes around failures [5], whereas our overlay network routes around countries. ARROW introduces a model that allows users to route around ISPs [130], but requires ISP participation, making it considerably more difficult to deploy than RAN. ARROW also aims to improve fault-tolerance, robustness, and security, rather than explicitly attempting to avoid certain countries; ARROW provides mechanisms

to avoid individual ISPs, but such a mechanism is at a different level of granularity, because an ISP may span multiple countries. Zhang *et al.* presented SCION, a "clean-slate" Internet architecture that provides route control, failure isolation, and explicit trust information for communication [183]; SCION, however, requires fundamental changes to the Internet architecture, whereas RAN is deployable today.

**Circumvention systems** Certain tools, such as anonymous communications systems or virtual private networks [40, 46, 70, 108, 109, 124, 131, 166, 168, 178], use a combination of encryption and overlay routing to allow clients to avoid censorship and surveillance. Tor is an anonymity system that uses three relays and layered encryption to allow users to communicate anonymously [46]. In contrast, RAN does not aim to achieve anonymity; instead, its aim is to ensure that traffic does not traverse a specific country, a goal that Tor cannot achieve. Even tools like Tor do not inherently thwart surveillance: Tor is vulnerable to traffic correlation attacks and some attacks are possible even on encrypted user traffic. VPNGate is a public VPN relay system aimed at circumventing national firewalls [124]. Unfortunately, VPNGate does not allow a client to choose any available VPN endpoint, which makes it more difficult for a user to ensure that traffic avoids a particular country. Neither of these systems explicitly avoid or route around countries. Additionally, existing circumvention systems generally rely on encryption, which is different from RAN; prior research has shown that websites can be fingerprinted based on size, content, and location of third party resources, which reveals information about the content a user is accessing [174]. Finally, ISPs often execute man-in-the-middle attacks on TLS connections to perform network-management functions [74].

# Chapter 4

# Hosting: CDN Design to Prevent Surveillance

As Content Distrubtion Networks (CDNs) host an increasing amount of content from a diversity of publishers, they are fast becoming targets of requests for data about their content and who is requesting it, as well as requests for takedown of material ranging from alleged copyright violations to offensive content. The shifting legal and political landscape suggests that CDNs may soon face liability for the content that they host. For example, the European Union has been considering laws that would remove safe harbor protection on copyright infringement for online service providers if they do not deploy tools that can automatically inspect and remove infringing content [59]. In the United States, various laws under consideration threaten aspects of Section 230 of the Communications Decency Act, which protects CDNs from federal criminal liability for the content that they host. Tussles surrounding speech, from copyright violations to hate speech, are currently being addressed in the courts, yet the legal outcomes remain ambiguous and uncertain, sometimes with courts issuing opposing rulings in different cases. And while this legal landscape is still changing, CDN operators have the capability to observe an enormous amount of information about both clients and

content. Not only is this a privacy concern if the CDN operator wishes to monitor traffic, but also if the CDN operator is served with a data request. Regardless of the legal and policy outcomes, CDNs are increasingly in need of *technical* protections against the monitoring and collection of clients' data and content that they serve.

Towards this end, we design and implement a system that allows clients to retrieve web objects from one or more CDNs, while preventing the CDNs from learning either (1) the content that is stored on the cache nodes; or (2) the content that clients request. We call this system an *oblivious CDN* (OCDN), because the CDN is oblivious to both the content it is storing and the content that clients request.

OCDN allows clients to request individual objects with identifiers that are encrypted with a key that is shared by an open proxy and the origin server that is pushing content to cache nodes, but is not known to any of the CDN cache nodes. To do so, the origin server publishes content encrypted with a shared key, which is subsequently shared with a proxy that is responsible for routing requests for objects corresponding to that URL. A client forwards a request for content through a set of peers (*i.e.*, other OCDN clients) in a way that prevents both other clients and the CDN from learning the client identity or requested content. After traversing one or more client proxies, an exit proxy transforms the URL that it receives from a client to an obfuscated identifier using the key that is shared with the origin server corresponding to the identifier. Upon receiving that request from the exit proxy, the CDN returns the object corresponding to the object identifier; that object is encrypted with a key that is shared between the origin and the proxy. This approach allows a user to retrieve content from a CDN without any node in the CDN ever seeing the URL or the corresponding content, or even knowing the identity of the client that made the original request. Using OCDN requires only minimal modification to existing clients; clients can also configure aspects of the system to trade off performance for privacy.

Ensuring that the CDN operator never learns information about either (1) what content is being stored on its cache nodes or (2) which objects individual clients are requesting is challenging, due to the many possible inference attacks that a CDN might be able to mount. For example, previous work has shown that even when web content is encrypted, the retrieval of a collection of objects of various sizes can yield information about the web page that was being retrieved [29, 128]. Similarly, URLs can often be inferred from relative popularity in a distribution of web requests, even when the requests themselves are encrypted. Additionally, the OCDN design assumes a strong attack model (Section 4.2), whereby an adversary can request logs from the CDN, interact with OCDN as a client, a proxy, or a publisher, and mount coordinated attacks that depend on multiple such capabilities. Our threat model does not include active attempts to disrupt the system (*e.g.*, blocking access to parts of the system, mounting denial of service attacks), but it includes essentially any type of attack that involves observing traffic and even directly interacting with the system as a client or a publisher.

The design of OCDN (Section 5.2) under such a strong attack model entails many unique aspects and features. Because the system allows any client to join as a proxy, even setting up the infrastructure is challenging. For example, an attacker could try to join the system as a proxy with the intent of proxying for specific web content, in an attempt to either disrupt or surveil those requests. To counter this threat, OCDN uses consistent hashing to map object identifiers (*i.e.*, URLs) to the proxy responsible for ultimately routing traffic to the CDN that hosts the object; to ensure that publishers only communicate keys to the proxies responsible for their content, each proxy must prove its identity to the respective publisher using a proof that relies on a self-certifying identifier.

Requesting and retrieving content, a process that we describe in detail in Section 4.4, is challenging since neither the CDN nor the proxy must know which client

originated a request for a specific piece of content. The key exchange between an origin server and its respective proxy protects the confidentiality of both the content and the identifier (*i.e.*, the URL) from the CDN. To obfuscate the source of the original request, clients construct a source route to an *exit proxy*, but the route can be prepended with proxies that precede the client who originated the request. To defend against various inference attacks, as well as to balance load, the OCDN design allows publishers to use multiple CDNs to distribute the same content, ensuring that no single CDN has access to information such as the relative popularity distribution of all objects. To ensure that no single proxy learns the request pattern for a single object, as well as to balance load, the design also can also use consistent hashing to assign a set of proxes to a single object.

## 4.1 Background

We now outline how a CDN typically operates, including what information it has access to by virtue of running a CDN. We also discuss some of the ongoing legal questions that CDNs currently face.

### 4.1.1 Content Distribution Networks

CDNs provide content caching as a service to content publishers. A content publisher may wish to use a CDN provider for several reasons:

- CDNs cache content in geographically distributed locations, which allows for localized data centers, faster download speeds, and reduces the load on the content publisher's server.
- CDNs typically provide usage analytics, which can help a content publisher get a better understanding of usage as compared to the publisher's understanding without a CDN.

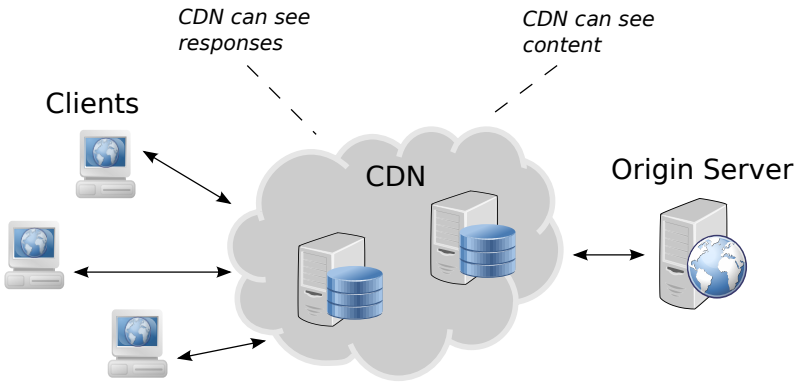



Figure 4.1: The relationships between clients, the CDN, and content publishers in CDNs today.

- CDNs provide a high capacity infrastructure, and therefore provide higher availability, lower network latency, and lower packet loss.
- CDNs' data centers have high bandwidth, which allows them to handle and mitigate DDoS attacks better than the content publisher's server.

CDN providers usually have a large number of edge servers on which content is cached; for example, Akamai has more than 216,000 servers in over 120 countries around the world [4]. Having more edge servers in more locations increases the probability that a cache is geographically close to a client, and could reduce the end-to-end latency, as well as the likelihood of some kinds of attacks, such as BGP (Border Gateway Protocol) hijacking. This is evident when a client requests a web page; the closest edge server to the client that contains the content is identified and the content is served from that edge server. Most often, this edge server is geographically closer to the client than the content publisher's server, thus increasing the speed in which the client receives the content. If the requested page's content is not in one of the CDN's caches, then the request is forwarded to the content publisher's server, the CDN caches the response, and returns the content to the client.

### 4.1.2 What CDNs Can See

Because the CDN interacts with both content publishers and clients, as shown in Figure 4.1, it is in a unique position to learn an enormous amount of information. CDN providers know information about all clients who access data stored at the CDN, information about all content publishers that cache content at CDN edge servers, and information about the content itself.

**Content.** CDNs, by nature, have access to all content that they distribute, as well as the corresponding URL. First, the CDN must use the URL, which is not encrypted or hidden, to locate and serve the content. Therefore, it is evident that the CDN already knows what content is stored in its caches. Because CDNs provide analytics to content publishers, they keep track of cache hit rates, and how often content is accessed. The CDN not only knows about the content identifier; it also also has access to the plaintext content. A CDN performs optimizations on the content to increase performance; for example, CDNs minimize CSS, HTML, and JavaScript files, which reduces file sizes by about 20%. They can also inspect content to conduct HTTPS re-writes; we discuss how OCDN handles these types of optimizations in Section 4.8. In addition, requesting content via HTTPS does not hide any information from the CDN; if a client requests a web page over HTTPS, the CDN terminates the TLS connection on behalf of the content publisher. This means that not only does the CDN know the content, the content identifier, but also it knows public and private keys, as well as certificates associated with the content it caches.

**Client information.** Clients retrieve content directly from the CDN's edge servers, which reveals information about the client's location and what the client is accessing. CDNs can also see each client's cross-site browsing patterns: CDNs host content for many different publishers, which allows them to see content requests for content

published by different publishers. This gives an enormous amount of knowledge to CDNs; for example, Akamai caches enough content around the world to see up to 30% of global Internet traffic [3]. The implications of a CDN having access to this much information was evident when Cloudflare went public with the National Security Letters they had received [41]; these National Security Letters demanded information collected by the CDN and also included a gag order, which prohibits the CDN from publicly announcing the information request.

**Content publisher information.** A CDN must know information about their customers, the content publishers; the CDN keeps track of who the content publisher is and what the publisher's content is. The combination of the CDN seeing all content in plaintext and the content's linkability with the publisher, gives the CDN even more power. Additionally, as mentioned previously, the CDN often holds some of the publisher's keys (including private keys), and the publisher's certificates. This has led to doubts about the integrity of content because a CDN can impersonate the publisher from the client's point of view [115].

### 4.1.3 Open Legal Questions

Various parties are battling in the courts over cases that pertain to user data requests and intermediary liability. Large companies often have large numbers of users, which makes them a target of data requests, for example by a government entity. Intermediary liability would impose criminal liability on an Internet platform (or a CDN) for the content it provides on behalf of its customers or users. In the following section, we highlight some of these cases, which all point to a key problem that CDNs face: by knowing all the content that they distribute, CDNs may be burdened with the legal responsibility because of the actions of their customers and clients.

**User Data Requests.** There are numerous open questions in the legal realm regarding which government can request data stored in different countries, which has led to much uncertainty. A series of recent events have illustrated this uncertainty. In the struggle over government access to user data, cases such as *Microsoft vs. United States* (often known as the "Microsoft Ireland Case") concerns whether the United States Government should have access to data about U.S. citizens stored abroad, given that Microsoft is a U.S. corporation.

Additionally, there have been user data requests asked of CDNs. The Cloudflare CDN has been required to share data with FBI [41]; similarly, leaked NSA documents showed that the government agency "collected information 'by exploiting inherent weaknesses in Facebook's security model' through its use of the popular Akamai content delivery network" [87].

**Intermediary Liability.** More recently, questions on intermediary liability have been in the spotlight. For example, many groups, including the Recording Industry Association of America (RIAA) and the Motion Picture Association of America (MPAA), have started targeting CDNs with takedown notices for content that allegedly infringes on copyright, trademarks, and patent rights; CDNs are a more convenient target of these takedown notices than the content provider because oftentimes the content provider is either located in a jurisdiction where it is difficult to enforce the takedown, or it is difficult to determine the owner of the content [38,39]. In 2017, a district court ruled that Cloudflare is not protected from anti-piracy injuctions by the Digital Millennium Copyright Act (DMCA); the RIAA obtained a permanent injunction against a site known as MP3Skull, which contained pirated content, and was distributed by Cloudflare. The ruling did not specify that Cloudflare was enjoined with MP3Skull under the DMCA, but rather that Cloudflare was helping MP3Skull in evading the injunction (under Rule 65 of the Federal Rules of Civil Procedure) [47].

The role of a CDN as an intermediary has also come into question in new and currently pending legislation, including a new German hate speech law and a bill proposed by the U.S. Senate called Stop Enabling Sex Traffickers Act (SESTA). In October 2017, Germany passed a new law that imposes large fines, upwards of five million euros, on social media companies that do not take down illegal, racist, or slanderous comments and posts within 24 hours [43]. The law targets companies such as Facebook, Google, and Twitter, but could also apply to smaller companies, which could be serviced by CDNs. In the latter case, it is an open question whether this new law also applies to CDNs. In the United States, the SESTA bill makes Internet platforms liable for their user's illegal comments and posts [13]. SESTA can hold CDNs liable for the content that they distribute (despite the CDN not being a party in the content publishing); these types of laws can naturally lead to overblocking, where an intermediary errs on the side of caution and censors more content than it needs to.

## 4.2 Threat Model and Security Goals

In this section, we describe our threat model, outline the capabilities of the attacker, and introduce the design goals and protections that OCDN provides.

### 4.2.1 Threat Model

Our threat model is a powerful adversary who has a variety of capabilities, including both surveilling activities and joining the system in various capacities. We assume that an adversary can gain access to the CDNs logs, which typically contains client IP addresses and URLs for each request. This adversary could be the CDN itself or a party who can compromise the CDN. Additionally, the adversary could join OCDN as either a client or any number of clients, or as an arbitrary number of exit

proxies. The adversary could also act as an origin server (a content publisher). We also assume that the adversary can coordiante several of these actions to learn more information. For example, the adversary could join as a client and an exit proxy, and access the CDN's logs to observe how its own requests are obfuscated. Additionally, the adversary can perform actions, such as generating requests as a client, or creating content as a content publisher. The goal of this type of adversary is to learn about the content being stored at the CDN and/or learn about which clients are accessing which content.

The strong adversary that we consider has seen some precedent in practice: for example, governments have demanded access to CDNs' data [41]. Although one possible adversary is a government requesting logs from the CDN, the government could also be colluding with a CDN; the CDN operator might even be an adversary.

Our design does not defend against an attacker who attempts to actively disrupt or block access to the system, such as by actively modifying content, disrupting communications (*e.g.*, through denial of service), or blocking access, content, or requests. Prior work on securing CDNs has introduced methods to handle an actively malicious adversary by preserving the integrity of content stored on CDN cache nodes [115]. We do not address an adversary that tampers, modifies, or deletes any data, content, or requests.

### 4.2.2 Security and Privacy Goals for OCDN

To defend against the adversary described in Section 4.2.1, we highlight the design goals for OCDN. Each stakeholder—in this case the content publisher, the CDN, and the client—has different risks, and therefore should have different protections. All three stakeholders can be protected by preventing CDNs from learning information, decoupling content distribution from trust, and maintaining the performance benefits of a CDN while reducing the probability of attacks. One strength of OCDN is that

it protects the origin server, the CDN itself, and the client, whereas existing systems, such as Tor, only protect the client.

**Prevent the CDN from knowing the content it is caching.** First and foremost, the CDN should not have access to the information outlined in Section 5.1. By limiting the information that the CDN knows, OCDN limits the amount of information that an adversary can learn or request. OCDN should hide the content as well as the URL associated with the content. If the CDN does not know what content it is caching, then the CDN will not be able to supply an adversary with the requested data and it will have a strong argument as to why it cannot be held liable for its customers' content.

**Prevent the CDN from knowing the identity of users accessing content.** CDNs can currently see clients' browsing patterns. OCDN should provide privacy protections by hiding which client is accessing which content at the CDN. In addition, it should hide cross-site browsing patterns, which a CDN is unique in having access to. Some CDNs block legitimate Tor users because they are trying to protect cached content from attacks, such as comment spam, vulnerability scanning, ad click fraud, content scraping, and login scanning [37]; for example, Akamai blocks Tor users [105]. As a positive side effect, OCDN prevents privacy-conscious Tor users from being blocked by CDNs. Finally, some CDNs, due to their ability to view cross site browsing patterns, could de-anonymize Tor users [164], but OCDN would prevent a CDN from compromising the anonymity of clients.

### 4.2.3 Performance Considerations

As one of the primary functions of a CDN is to make accessing content faster and more reliable, OCDN should consider performance in design decisions. The performance of OCDN will be worse than that of traditional CDNs because it is performing more

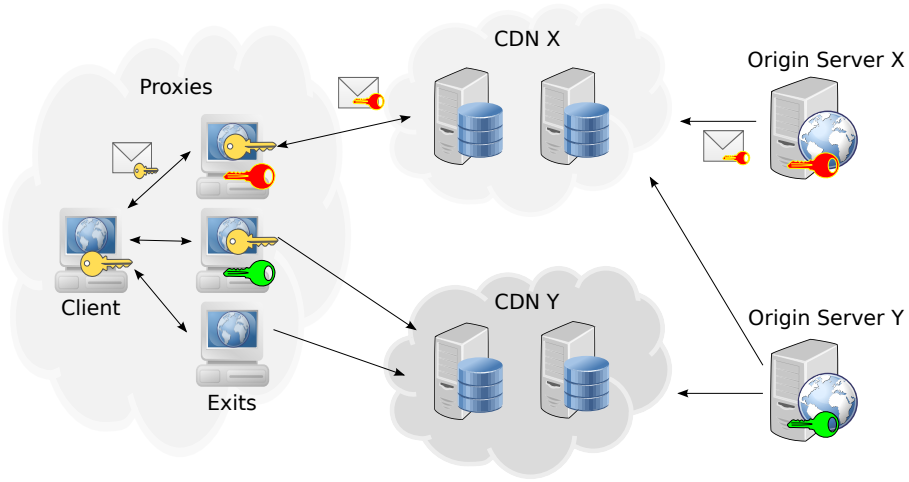



Figure 4.2: The relationships between clients, exit proxies, CDNs, and origin servers in OCDN.

operations on content, but OCDN is offering confidentiality, whereas traditional CDNs are not. OCDN should scale linearly with the number of clients in terms of load and storage requirements on exit proxies; additionally, it should be able to scale with the number of clients using the system, as well as with the growing number of web pages on the internet.

## 4.3 OCDN Design

OCDN provides oblivious content distribution and comprises the following components: clients, exit proxies, CDNs, and origin servers. *Clients* are the Internet users who use the system to access content stored on CDN cache nodes; *exit proxies* are proxies that obfuscate the requests and responses retrieved from the CDNs; and the *origin servers* are the content publishers who are customers of the CDNs. Figure 4.2 shows how these components interact in the system. This section describes the decisions made in the design of OCDN, and what functionality each decision provides. We separate design decisions into two parts: 1) hiding content and 2) hiding clients. We also highlight some additional options that the design of OCDN allows.

| Design Decision | Function |
|---|---|
| Shared Keys | Hides content on cache nodes from CDN. |
| Consistent Hashing | Load balance requests across proxies; ensure no proxy can control a given URL. |
| Self-Certifying Identifiers | Authenticates exit proxies to origin servers. |
| DNS for Key Sharing | Allows origin server to share shared keys with exit proxies. |

Table 4.1: Design decisions associated with hiding content from a CDN.

### 4.3.1 Hiding Content

We start by discussing how the system components communicate and authenticate one another; Table 4.1 summarizes these decisions. We introduce shared keys between origin servers and exit proxies, how these keys are stored, how the exit proxies authenticate themselves to origin servers, and how these keys are distributed.

**Shared Keys.** To prevent an adversary from learning information, the CDN must not know anything about the content that it is caching. Therefore, the content *and* the associated URL must be obfuscated before the CDN sees them. The content can be obfuscated by encrypting it with a key that is not known to the CDN. Because this must be done prior to any caching, the content publisher must generate a shared key $k$ to encrypt the content with. Encrypting the content alone does not hide much from the CDN; the content identifier, or URL, must also be obfuscated, otherwise the CDN can still reveal information about which clients accessed which URLs (which is indicative of the content). In obfuscating the URL, the result should be fixed and relatively small; these requirements reduce storage requirements and prevent the adversary from guessing the URL based on the length of the obfuscated URL. Unfortunately, using a simple hash allows an attacker to guess the content identifier by hashing guesses and comparing with the hashes stored in the CDNs caches. Therefore, the content publisher incorporates the use of the shared key $k$ into the hash of the

URL by using a hash-based message authentication code (HMAC). Additionally, if the domain supports HTTPS requests, then the content publisher must also encrypt the associated certificate with the same key $k$.

The encrypted content and corresponding HMAC are sent to the CDN[1] and stored in its caches. The content publisher then shares the key $k$ with an exit proxy. This key allows the exit proxy to request encrypted content on behalf of clients by computing the HMAC on the URL.

**Consistent Hashing.** Each exit proxy stores a mapping of URLs to their associated shared key $k$; for example, if an origin server has shared key $k$ and publishes a web page `www.foo.com`, then an exit proxy will store the mapping of `www.foo.com` to $k$. The set of exit proxies jointly compute a distributed hash table where the key is the URL (`www.foo.com`) and the value is the shared key ($k$). To assign (key,value) pairs to exit proxies, OCDN uses consistent hashing [99, 116]. Consistent hashing uses a hash function $H(.)$ to generate identifiers for both exit proxies and for URLs; the identifiers are $H(exit\_ID)$ and $H(URL)$. We discuss what $exit\_ID$ is in the next section on Self-Certifying Identifiers. After the hashes are computed, then they are mapped to a point on an identifier circle (modulo $2^m$, where $m$ is the length of identifier in bits); each URL ($H(URL)$) on the circle is assigned to the first exit proxy ($H(exit\_ID)$) that is equal to or follows $H(URL)$ on the circle. This hashing method is used in OCDN because it provides: 1) an evenly distributed mapping of URLs to shared keys among the exit proxies, 2) a way to prevent an exit proxy from choosing which URL it wishes to be responsible for, and 3) a relatively small amount of (key,values) to be moved when a new exit proxy is established (or removed).

[1]Most CDNs allow the publisher to decide on a push or pull model, but OCDN is compatible with either approach.

**Self-Certifying Identifiers.** Consistent hashing uses identifiers for both the URLs and the exit proxies. While the identifiers for URLs are straightforward ($H(URL)$), the identifiers for exit proxies must provide more information; an exit proxy identifier must be able to prove to an origin server that it is the exit proxy that is responsible for the associated URL. If this validation was not part of OCDN, then any (potentially malicious) exit proxy could request the shared key $k$ from any or all origin servers. To prevent a malicious exit proxy from learning any shared key $k$, the proxy must be identified by a self-certifying identifer. This technique was first introduced in a self-certifying file system [121]; it allows for other entities (such as origin servers) to certify the exit proxy solely based on its identifier. The format of this identifier ($exit\_ID$) is `IP:hostID`, where `IP` is the exit proxy's IP address and `hostID` is a hash of the exit proxy's public key. A malicious exit proxy cannot *choose* where on the consistent hashing ring it sits because it cannot frequently change and re-hash its own IP address (whereas it could re-generate a new public key).[2] When an exit proxy is requesting the shared key $k$ from an origin server, it sends its identifier and its public key to the origin server. The origin server can then hash the exit proxy's public key and verify it against the `hostID`; this action serves as a proof of the exit proxy's position in the consistent hashing circle, and thus prevents a proxy from lying about where it lies on the ring (and subsequently lying about which URL's shared key it is responsible for). Note that this $exit\_ID$ is used on the consistent hashing circle as $H(IP) : H(hostID)$; the $exit\_ID$ must be the same length as $H(URL)$, so the $exit\_ID$ consists of the first half of the bits of $H(IP)$ concatenated with the first half of the bits of $H(hostID)$.

[2]While a malicious exit proxy cannot specifically choose its location on the hashing ring, it could recompute a public key until it finds a certain location on the hashing ring. This is limited by the fact that the exit proxy's IP address is part of its identifier, and we assume that the adversary running the exit proxy cannot change IP addresses to a value of his choice or in a frequent manner. A potential attack that an adversary can execute on a DHT using a consistent hashing scheme is a Sybil attack, where the adversary runs *many* exit proxies to hopefully place himself in his desired location on the hashing ring. We describe countermeasures to a Sybil attack in Section 4.6.

| Design Decision | Function |
|---|---|
| Spoofed Source Routes | Hides origin of client request from other clients, exit proxies, and CDN. |
| Session Keys | Hides URL and response from other clients. |
| Multicast Response | Allows CDN to return content directly to client without knowing the client that requested the content. |

Table 4.2: The design decisions associated with content requests and responses, and what these decisions provide.

**DNS for Key Sharing.** We have discussed how shared keys are generated, used, and stored, and here we describe how they are shared. As previously stated, the origin servers generate shared keys and must share them with the (correct) exit proxies. OCDN uses DNS to do so. To retrieve a shared key $k$, an exit proxy sends a DNS query to the origin server's authoritative DNS, and it includes its identifier, $exit\_ID$, and its public key in the `Additional Info` section of the query. The authoritative DNS for the origin server validates the exit proxy by hashing the public key and comparing it to the second part of $exit\_ID$, and verifying that the exit proxy is responsible for its URL based on the consistent hashing circle. If the verification is successful, then the authoritative DNS sends the shared key $k$ encrypted under the exit proxy's public key, $\{k\}_{PK_{exit}}$ in the SRV record of the DNS response. The exit proxy extracts $k$ by decrypting with its private key, and stores it in its hash table.

### 4.3.2 Hiding Clients' Identities

We make additional design choices that concern the requests that clients initiate and the responses they receive. Table 4.2 highlights these decisions; we introduce session keys, how requests are routed from clients to exit proxies, and how responses are routed from exit proxies back to the original client.

**Potentially Spoofed Source Routes.** As previosuly described, exit proxies query the CDN on behalf of clients, but the exit proxy should not be able to learn which client sent which request. This obfuscation is accomplished by routing requests through a series of other clients. In OCDN, each client is running a proxy and is also a peer in this system; this peer-to-peer system of clients borrows the protocols used for clients joining, leaving, and learning about other clients from the vast literature on peer-to-peer systems. A client routes a request through her peers by using source routing; when the client generates a request, it also generates a source route, which includes the addresses of a set of her peers. The last hop in the source route is the exit proxy that is responsible for the shared key $k$ associated with the URL in her request. The client determines the correct exit proxy by looking this up in a local mapping (which is retrieved from a central system that keeps the mapping of URLs to exit proxies). It appends this source route to its request and forwards it to the next peer in the route. When a peer receives a request, she simply forwards it on to the next peer; this continues until the last hop in the source route, which is an exit proxy.

Although it might initially appear as if it is easy to identify the client that initiated a request as the first hop in the source route, OCDN allows each client to spoof source routes; specifically, a client can prepend other peers in the route before it initiates a request. For example, a client with identity $C$ could generate a route to exit proxy $E$ that looks like $C \rightarrow G \rightarrow F \rightarrow E$ and can further obfuscate the source of the route by prepending additional clients to the beginning of the route as follows:

$D \rightarrow A \rightarrow C \rightarrow G \rightarrow F \rightarrow E$.

Neither $G$, $F$, nor $E$ know who the original requestor was; from $E$'s point of view, the original requestor could have been $D$, $A$, $C$, $G$, or $F$. Using a sequence of peers, or even just knowing that a client *can* use a series of peers, hides the identity of the client from other clients, exit proxies, and the CDN.

**Session Keys for Request and Response Confidentiality.** In addition to shared keys between origin servers and exit proxies, OCDN uses session keys shared between clients and exit proxies. Session keys provide confidentiality of the requested URL and the response. When the client generates a request, it generates a session key *skey*. and encrypts the URL in her request with this key, which provides $\{\text{URL}\}_{skey}$. The client must also share this session key with the exit proxy, so that the exit proxy can learn the plaintext URL and subsequently compute the HMAC to query the CDN. The client encrypts the session key with the exit proxy's public key, result in $\{\text{skey}\}_{PK_{exit}}$, and appends this value as an additional header on the request. Because her request could be forwarded through a set of client peers, this hides the URL of the request from other clients.

When an exit proxy receives a request from a client, it first extracts the session key *skey* by decrypting it with the proxy's private key, and then the proxy decrypts the URL with the session key. This operation yields the original plaintext URL. Using the shared key $k$ from the origin server, the proxy can then compute $\text{HMAC}_k(\text{URL})$ and forward the request to the CDN. Upon receiving a response from the CDN, the exit proxy then decrypts the content with the shared key $k$, and encrypts the content with the session key *skey* before sending it to the client. When it receives the encrypted response, the client can then decrypt it using *skey*.

**Multicast Responses.** Using session keys allows for a performance optimization in sending responses back to clients. Instead of sending the encrypted response from the exit proxy back to the client via the set of peers used in the source route, the exit proxy can send it in a multicast manner to all clients that were on the source route. The only client that knows *skey* is the true client that originated the request, therefore none of the other clients can interpret the response, and it reduces the latency for sending the response to the client.

### 4.3.3 Incentives for Running OCDN

As described in Sections 4.3.1 and 4.3.2, OCDN relies on the use of a system of proxies. These include: 1) client proxies, and 2) exit proxies. For a client to join the system, they must also run client proxy on his/her machine. On the other hand, there is no requirement for a client, organization, or company to run an exit proxy; despite this lack of requirement, we believe that both clients and companies have enough incentives to run an exit proxy (or multiple exit proxies). Clients benefit from running an exit proxy because it allows OCDN to perform better in terms of both performance and privacy; when there are more exit proxies, then there is less load put on each exit proxy, and there is a smaller chance that an attacker has access to most exit proxies. Companies benefit from running an exit proxy for similar reasons — performance and privacy —, but it could help Internet users access their content quicker (assuming they have content cached by the CDN).

### 4.3.4 Design Alternatives

While our explanation of the design of OCDN describes a series of proxies that are run by us, but are not trusted; here we describe alternative designs regarding the system of proxies. The system will work with both a closed, trusted system of proxies, as well as with an open (untrusted) system of proxies.

**Closed System of Proxies.** While the proxies in OCDN are not trusted, OCDN could use a system of closed and trusted proxies. There are potential groups or organizations that would support this type of system, and would be willing (and trusted) to run the exit proxies. If the exit proxies were trusted, then parts of the design of OCDN could be simplified; for example, if proxies were trusted, then OCDN would not need to hide the identity of clients from the exit proxies, and could remove spoofed source routes from the design. The primary drawback of this approach is finding an organization that everyone could trust to run the exit proxies.

**Open System of Proxies.** In OCDN, exit proxies are not trusted with client identities and information, which removes the need to find a universally trustworthy organization. In an alternative approach, OCDN could use a completely open system of proxies that are untrusted, which would allow anyone (clients, companies, etc.) to run an exit proxy. The addition of an exit proxy follows the protocol in consistent hashing for when a new node joins; some keys would be transferred to the new exit proxy, and clients' mapping of exit proxies will be updated. This allows for the load to be split among more proxies and increases the geographic diversity of the exit proxies.

### 4.3.5 Design Enhancements

Up to this point, we have discussed how OCDN is designed in the general case. Here we describe some additional options that OCDN's design can include.

**Multiple CDNs.** While describing the design decisions that went into OCDN, we referred to a single CDN for simplicity. In reality, OCDN allows for many CDNs to participate; distributing content across multiple CDNs could provide additional privacy. Origin servers can also take advantage of multiple CDNs.

**Encoding URLs.** As described earlier, each URL is obfuscated by using a HMAC and then stored on the CDN. An adversary could potentially correlate a URL's popularity with its access patterns. To prevent this, OCDN allows origin servers to generate multiple different encodings of its URLs, such that $\text{HMAC}_k(\text{enc}_1(\text{URL})) \neq \text{HMAC}_k(\text{enc}_2(\text{URL}))$. Each origin server could produce $n$ different encodings of popular URLs, such that the popularity distribution seen by an adversary is a uniform distribution of URL requests across all URLs. Another way to prevent correlation based on access patterns is to keep the URL constant, but change the encryption

scheme; for example, the URL could be hashed and then encrypted with a randomized encryption scheme.

**DHT Replicas.** Each exit proxy's hash table can be replicated by another (or many other) exit proxies. This would provide less load per exit proxy, as well as redundancy in case of failures. Additionally, the CDN can cache the content associated with a given URL at more than one cache node; if only one exit proxy is responsible for a given URL's content, then it would likely only be cached at cache node closest to the exit proxy. Having multiple exit proxies responsible for a URL's content helps decrease the load on the proxies while maintaining some of the performance benefits of a CDN.

**Partial Content.** Different origin servers have different needs, and each origin server might have different needs for different content. The design of OCDN allows origin servers to publish some of their content on OCDN and some on other CDNs. This is useful in a case where some content is more sensitive, while other content needs better performance.

**Pre-Fetch DNS Responses.** One way to increase the performance of OCDN is to pre-fetch DNS responses at the exit proxies. This would allow the exit proxy to serve each client request faster because it would not have to send as many DNS requests. Pre-fetching DNS responses would not take up a large amount of space, but it also would not be a complete set of all DNS responses. Additionally, if the content is moved between cache nodes at the CDN, then DNS response must also change; therefore, the pre-fetched DNS responses should have a lifetime that is shorter than the lifetime of the content on a cache node.

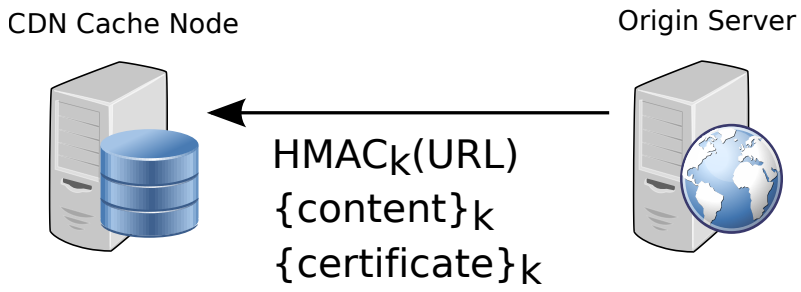



Figure 4.3: How content is published in OCDN. $k$ is shared between the origin server and the corresponding exit proxy; the CDN has no knowledge of $k$.

**Privacy vs. Performance Tradeoffs.** There are two different modes that OCDN can operate in, where one provides better performance, and the other provides better privacy. In the first mode, the client can choose to send a request directly to the exit proxy.

In this case, the exit proxy might be able to discover the identity of the client, but the CDN would still not be able to map a request to the client that made the request. Alternatively, the client can forward a request through a set of peers before it reaches the exit proxy. In this case, the client can prepend other clients' identifiers (as previously described) to make it appear as though the request came from a different client. This action further obscures the relationship between the client and the request. As another option, the client could *only* prepend other clients'identifiers but simply forward the request *directly* to the exit proxy; this action provides the same performance benefit as the first mode, but still offers some additional privacy benefits. Although the last option would appear to strike the optimal balance between privacy and performance, it cannot be the only option because the exit proxy would always know that the true client is the previous hop in the source route. These modes of operation provide clients with different ways to use the system both based on their privacy preferences and the type of content they are requesting.

## 4.4 OCDN Protocol

Based on the design decisions discussed in the previous section, we specify the steps taken to publish and retrieve content in OCDN.

### 4.4.1 Publishing Content

In order to publish content such that the CDN never sees the content, the publisher must first obfuscate her content, as described in Section 5.2. Figure 4.3 shows the steps taken to publish content.

The most important step in content publishing is obfuscating the data. We assume that the origin server already has a public and private key pair, as well as a certificate. To obfuscate the data the origin server will need to generate a shared key $k$.

Once the key is established, the origin server must first pad the content to the same size for some range of original content sizes (i.e., if content is between length x and y, then pad it to length z). The range of content sizes should be small, such that this causes negligible padding overhead, but reduces the probability of identifying the content based on the content length. This content padding is done to hide the original content's length, as it may be identifiable simply by its length. After content is padded, then the content is divided into fixed size blocks and padded to some standard length. Then each block is encrypted using the shared key $k$, resulting in a set of encrypted blocks. Because the CDN does not have access to the shared key, it cannot see what content it is caching.

Now that the content is obfuscated, the origin server must also obfuscate the content's identifier. To do so, she computes the HMAC of the URL using the shared key $k$.

Once the identifier and the content are obfuscated with $k$, they can be pushed to the CDN, or optionally to multiple CDNs. Recently, services have cropped up to

allow and help facilitate the use of multiple CDNs for the same content; an origin server could use multiple CDNs' services. This mechanism could be used in OCDN to increase reliability, performance, and availability; an origin server can use a service, such as Cedexis [33], to load balance between CDNs. We discuss the use of multiple CDNs more in Section 4.4.4 on OCDN in partial deployment.

As the exit proxies use consistent hashing to divide keys among proxies while balancing load, the origin server determines which exit proxy is correct (based on the consistent hashing circle). The origin server then encrypts the shared key $k$ with the correct exit proxy's public key $PK_{exit}$. Figure 4.6 shows the steps for retrieveing a shared key. First, the exit proxy sends a DNS request to the origin server's authoritative DNS server, including its self-certifying identifier and its public key (these are both included in the `Additional Info` section of the DNS message). The origin server hashes the exit proxy's public key and verifies it against its self-certifying identifier; this acts as a proof of the exit proxy's position in the consistent hashing circle. If the origin is able to certify the exit proxy, then it will send the DNS response with $\{k\}_{PK_{proxy}}$ in the SRV record. The exit proxy will receive the encrypted shared key, which it can decrypt with its private key.

**Updating Content.** For an origin server to update content, she must follow similar steps as described in publishing content. Once she has updated the content on her origin server, she must obfuscate it using the same steps: 1) pad the original content length, 2) divide the content into fixed size blocks, and 3) encrypt the content blocks with the shared key $k$. Because she is updating the content (as opposed to creating new content), the obfuscated identifier will remain the same. The origin server signs the updated obfuscated content with her private key, such that the CDN can verify it was true origin server that sent the update.

**Updating Keys.** An origin server must be able to update keys in case of compromise. To minimize the amount of time a key is compromised for, the origin server specifies an expiration date and time for the key when it is originally generated. The origin server periodically checks if the key is valid or not based on the expiration timestamp. If the key is still valid, the origin server continues to use it. Otherwise, the origin server generates a new key $k_{new}$, computes $\text{HMAC}_{k_{new}}(\text{URL})$, and encrypts the content (and possibly certificate) with $k_{new}$. The content publisher then follows the same steps as in Updating Content to push the content to the CDN, and it publishes $k_{new}$ encrypted with the exit proxy's public key in it's DNS SRV record.

The corresponding exit proxy must also be able to fetch this new key $k_{new}$ and replace the expired key with it. When the exit proxy sees an incoming request for a URL that uses key $k$, it first checks $k$'s timestamp. If valid, then it continues as normal. Otherwise, it sends a DNS request to the publisher's authoritative DNS, and extracts $\{k_{new}\}_{PK_{proxy}}$ from the DNS response. The exit proxy then decrypts it to obtain $k_{new}$, updates its version of the key, and proceeds as normal.

### 4.4.2 Retrieving Content

The steps for a client to retrieve a web page that has been cached by OCDN are shown in Figure 4.4, where the client forwards a request directly to an exit proxy; Figure 4.5 shows the steps to retrieve content when the client forwards its request through two peers. We assume the client has already joined the system, which is described in more detail in Section 4.4.3; at this stage, the client has knowledge of a subset of its peers (other OCDN clients) and a mapping of exit proxies and which URLs they hold keys for. The client generates a request for a specific URL, and looks up which exit proxy holds the key for that URL in its local mapping. Next, the client selects a source route; this source route allows the client to specify which mode of OCDN they would like to use: 1) no additional source route, which has better

performance, or 2) a source route, which has better privacy. If the client decides to use the privacy-preserving mode, then she generates a source route, which includes some of its peers, and could potentially include a false originator (as described in Section 5.2). Before sending the request, the client generates a session key $k_{session}$ and encrypts it with the exit proxy's public key. The client appends both the source route and $\{k_{session}\}_{PK_{proxy}}$ to the request and encrypts the URL with $k_{session}$ such that no other clients on the path can learn what the requested URL is. The client then sends it onto the next proxy in the source route, which could be either another client proxy or the exit proxy. The request is forwarded through all subsequent hops in the source route until it reaches the exit proxy. The exit proxy decrypts $\{k_{session}\}_{PK_{proxy}}$ with its private key and stores the source route locally; it then decrypts the URL with $k_{session}$.

The exit proxy then resolves the domain using its local resolver, which will redirect it to the CDN's DNS resolver. In order for the exit proxy to generate the obfuscated identifier to query the cache node for the correct content, it must have the shared key $k$ that the origin server generated and obfuscated the content and identifier with. The steps an exit proxy takes to retrieve the shared key were outlined in Section 4.4.1 and are shown in Figure 4.6.

Now that the proxy has obtained the shared key $k$ from the origin server, it can generate the obfuscated content identifier based on the request the client sent. It computes the HMAC of the URL with the shared key. The proxy then sends the (obfuscated) request to the edge server, where the CDN locates the content associated with the identifier. The CDN returns the associated obfuscated content, which we recall is the fixed-size blocks encrypted with the same shared key that the identifier was obfuscated with. The proxy can decrypt the content blocks with the shared key from the origin server, assemble the blocks, and strip any added padding, to reconstruct the original content.

Lastly, the exit proxy must send the response back to the correct client without knowing who the client is. First, the exit proxy fetches the session key $k_{session}$ that it stored for the corresponding incoming request, and it uses this key to encrypt the response. Then, it looks up the source route it stored for the corresponding request and uses a multicast technique to send to the encrypted response to all clients on the source route. At this point, the exit proxy can delete the source route and session key entries for this request/response. Only the original (true) client has $k_{session}$, so only the original (true) client can decrypt the response. All other clients will discard the encrypted response because they cannot decrypt it.

### 4.4.3 Clients Joining & Leaving

When a client joins OCDN, she will download OCDN client software. This includes information about exit proxy mappings to URLs for which they hold a key, software for modifying requests with session keys and source routes, and software for running a proxy. Clients will learn about other clients in the system via a gossip protocol. We do not detail this as gossip protocols have been studied extensively in the past [60, 104, 169]. Similarly, when a client leave the system, this information is propogated to its peers using a gossip protocol.

### 4.4.4 Partial Deployment

OCDN should be partially deployable, in the sense that if only some origin servers participate or only some CDNs participate, then the system should still offer some protections. We outline two different partial deployment possibilities below.

**Deployment with Origin Servers' Full Participation.** One option for deploying OCDN is to ensure there is some set $S$ of origin servers that participate fully in the system. These publishers obfuscate their content, identifiers, and certificates,

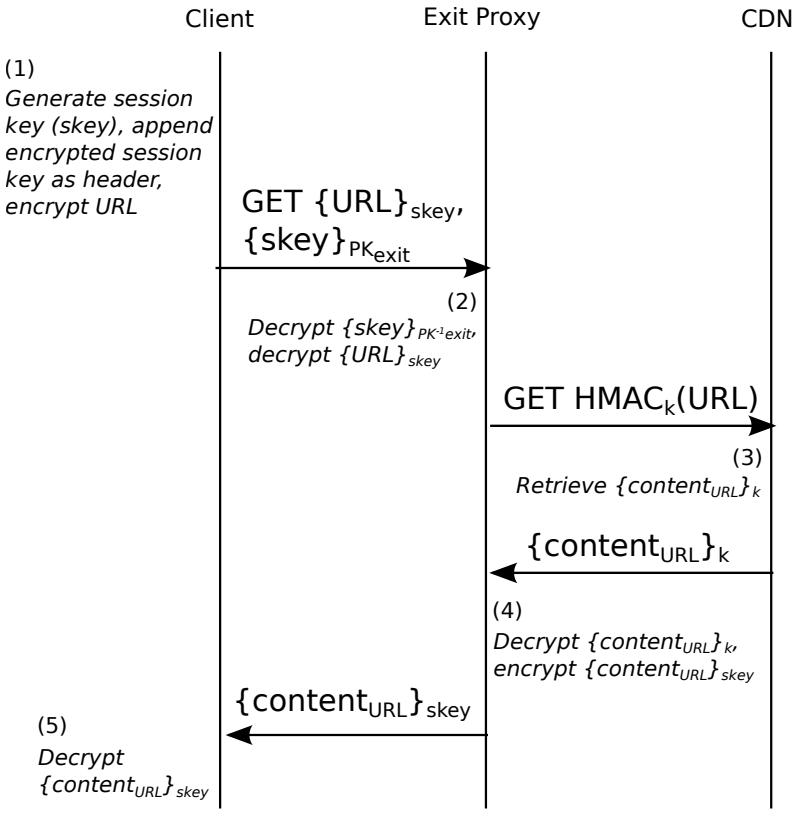



Figure 4.4: Steps for retrieving content in OCDN when a client is prioritizing performance and goes directly to an exit proxy.

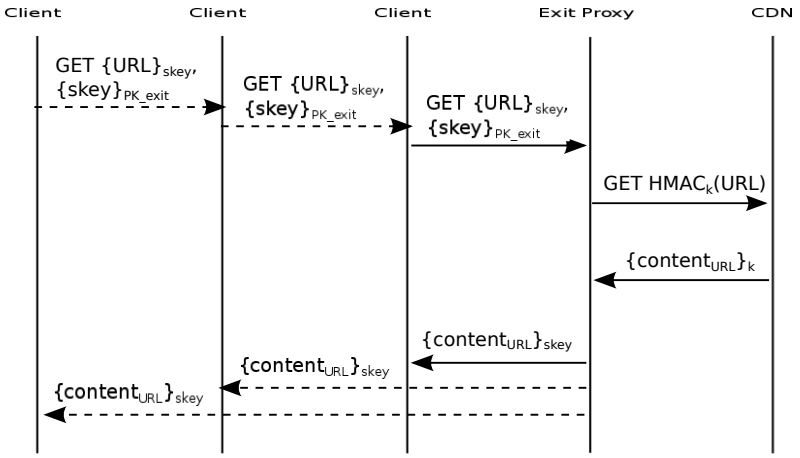



Figure 4.5: Steps for retrieving content in OCDN when a client is prioritizing privacy and proxies a request through two other clients before reaching the exit proxy. This figure shows that the request is sent sequentially through peers, and the response is sent in a multicast manner back to the clients.

and most importantly, only have obfuscated data stored on the CDNs cache nodes. $S$ must be greater than one, otherwise the CDN can infer that a client accessing this obfuscated content is actually accessing content that can be identified. This partial deployment plan protects the privacy of the clients accessing the content created by the set of origin servers $S$. It does not protect the clients' privacy as completely as full participation of all origin servers in OCDN because the CDN can still view cross site browsing patterns among the origin servers that are not participating. It is important to note though, that because the clients are behind proxies, the CDN cannot individually identify users. The CDN can attribute requests to exit proxies, but not to clients.

**Deployment with Origin Servers' Partial Participation.** Some origin servers may prioritize performance and availability. Therefore, they should have the option to gradually move towards full participation by pushing both encrypted and plaintext content to the CDN. In this partial deployment plan, we foresee some set of origin servers fully participating with only encrypted content, some other set of origin servers partially participating with both encrypted and plaintext content, and some last set of publishers that are not participating. Unfortunately, if a publisher has the same content that is both encrypted and plaintext content at a cache node, then an adversary can correlate the access patterns on encrypted and plaintext content for the origin server. In order to prevent this identification of the content, OCDN can use encoded URLs (described in Section 5.2), which obfuscates the access patterns for a given piece of content; this holds true if an origin server chooses to distribute its content in an encrypted manner using OCDN and in plaintext form on a different CDN. In this case, the origin server can still encode its URLs in multiple ways to prevent correlating access patterns between the encrypted and plaintext content. Therefore,

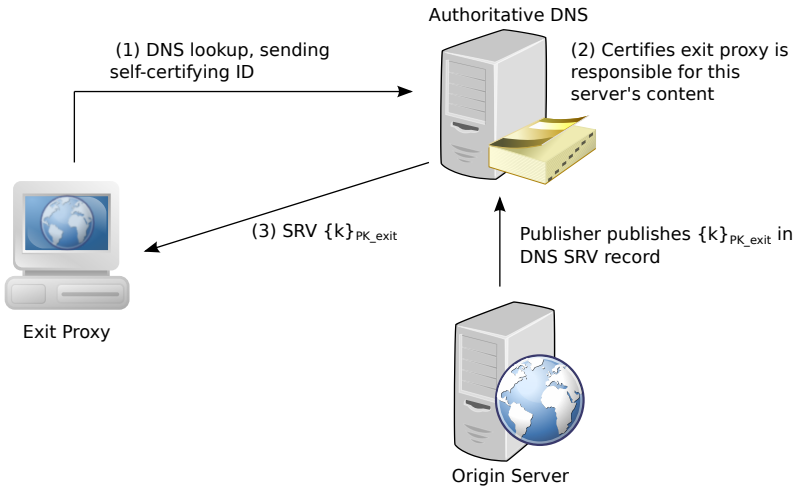



Figure 4.6: How an origin server certifies an exit proxy and distributes its shared key to an exit proxy. In step (1), the exit proxy sends his self-certifying ID in the *Additional* section of the DNS message.

this deployment option allows for differing levels of participation in the system, while still preserving the protections provided by OCDN.

## 4.5 Implementation

We have implemented a prototype of OCDN to demonstrate its feasibility and evaluate its performance. Our implementation allows a client to send a request for content through an exit proxy, which will fetch the corresponding encrypted content. Figure 4.7 shows our prototype; the solid line represents how OCDN communicates between the components, and the dotted line represents how a traditional CDN would communicate in our prototype. Here we will discuss each component—client proxy, exit proxy, and CDN—separately, and how they fit together.

**CDN.** As the design for OCDN requires encrypted content and identifiers to be stored in the CDN, we cannot request content from real-world CDNs. Additionally, we must evaluate the performance of OCDN in comparison to the same content, cache locations, etc., so we set up a data storage server. This server is run on a Virtual

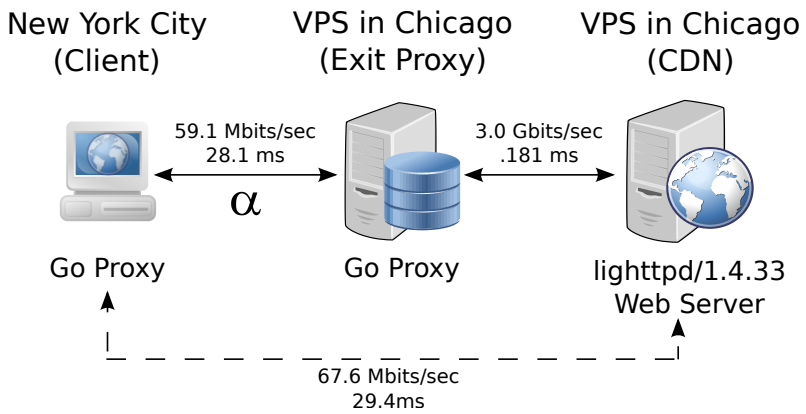



Figure 4.7: The implementation of our OCDN prototype. The solid line represents how OCDN communicates between the components; the dotted line represents how a traditional CDN would communicate. $\alpha$ represents the latency between the client and the exit proxy; we simulate additional clients on this path by increasing $\alpha$.

Private Server (VPS) located in Chicago, USA. To access content, we set up a web server on this VPS machine. To generate plaintext web content, we used Surge [15], which allows us to generate a set of files that are representative of real-world web server file distributions. In OCDN, the files are encrypted with a shared key $k$ and the obfuscated file name is the $\text{HMAC}_k$(file name). The shared keys are AES 256-bit keys and OCDN uses SHA-256 for the hash function. Both the plaintext files and encrypted files are stored on this web server, and for the purposes of evaluating our prototype, act as a CDN in OCDN.

**Exit Proxy.** The exit proxy is the component that queries the CDN for encrypted content on behalf of a client. We have implemented a web proxy in Go; this proxy runs on a different VPS machine in Chicago, USA. In addition to proxying web requests, the exit proxy also provides cryptographic functionality. When receiving a request, it rewrites the URL in the request to be the $\text{HMAC}_k$(URL), and it parses the headers to retrieve a specific header, `X-OCDN`, which contains the client's session key encrypted under the exit proxy's public key. Our implementation uses 2048-bit RSA for asymmetric encryption. After decrypting the session key, it stores it in memory

| | Preserves Integrity at CDN | Preserves Confidentiality at CDN | Protects Client Identity |
|---|---|---|---|
| Stickler [115] | ✓ | | |
| R & C [122] | ✓ | | |
| Tor [46] | | | ✓ |
| **OCDN** | | ✓ | ✓ |

Table 4.3: The security and privacy features offered by related systems. To our knowledge, OCDN is the first to address confidentiality at the CDN.

for use on the response. When it receives a response from the CDN, it decrypts the content with the shared key $k$, and subsequently encrypts it with the session key (both using AES 256-bit encryption). The exit then forwards the response onto the client proxy.

**Client Proxy.** The client proxy acts on behalf of the client that is requesting content. This proxy uses the same implementation as the exit proxy, but provides different cryptographic functions on the requests and responses. When a client makes a request, the client proxy generates a session key (AES 256-bit) and looks up the correct exit proxy's public key. The client proxy then adds a header to the request, where X-OCDN is the key, the encrypted session key is the value. The client then forwards this on to the exit proxy. When the client receives a response from the exit proxy, it must decrypt the content with the session key it originally generated.

## 4.6 Security Analysis

We analyze and discuss how OCDN addresses different attacks. Table 4.3 shows what security and privacy features OCDN provides in comparison to other related systems.

**Popularity Attacks.** An attacker that has requested or otherwise gained access to CDN cache logs can learn information about how often content was requested. Because not all content is requested uniformly, the attacker could potentially correlate the most commonly requested content with very popular webpages. While this does

not allow the CDN to learn which clients are accessing the content, it can reveal information about what content is stored on the CDN cache nodes. OCDN handles this type of attacker by making the distribution of content requests appear fairly uniform (not necessarily completely uniform). The content publisher (of popular content) generates multiple encodings of their content and URLs, and encrypts each one with the shared key $k$, such that they have multiple, different-looking copies of their content. All of the content copies are pushed to the CDN and the key is shared with the exit proxy.[3] Now, the popular content does not appear as popular, and it makes it difficult for an attacker to infer the popularity of the content.

**Chosen Plaintext Attacks.** An attacker could attempt to determine whether a particular URL was being accessed by sending requests through specific OCDN proxies and requesting access to the CDN cache logs, which contain the corresponding obfuscated requests and responses. Blinding the clients' requests with a random nonce that is added by the proxy should prevent this attack.[4] We also believe that such an attack reflects a stronger attack: from a law enforcement perspective, receiving a subpoena for *existing* logs and data may present a lower legal barrier than compelling a CDN to attack a system.

**Traffic Analysis Attacks.** If a CDN itself is malicious and is attempting to learn information about the content and/or clients, the CDN may act as a client in the system. In this attack the (malicious) client sends a request for content and the CDN can correlate the request with a content access at the CDN because they have knowledge of both the CDN logs and the requests they are making as a client. OCDN defends against this by using the exit proxies as mixes; each exit proxy is receiving different requests from different clients and then forwarding the requests on to the CDN. These exit proxies should mix the requests enough that the CDN

[3]This also provides load balancing for exit proxies that hold the shared key $k$ for the popular webpage because it distributes the load across multiple exit proxies (where each of these exit proxies are responsible for one of the encodings).

[4]The blinding feature was not implemented as part of the prototype discussed in Section 5.4.

cannot conduct traffic analysis and determine which request corresponds to which content on the CDN's cache nodes. There has also been numerous studies that have proposed and evaluated defenses against traffic analysis attacks [135, 180]; OCDN could implement one of these solutions at the exit proxy.

**Spoofed Content Updates.** Because the CDN cache nodes do not know either the content that they are hosting or the URLs corresponding to the content, an attacker could masquerade as an origin server and could potentially push bogus content for a URL to a cache node. There are a number of defenses against this possible attack. This simplest solution is for CDN cache nodes to authenticate origin servers and only accept updates from trusted origins; this approach is plausible, since many origin servers already have a corresponding public key certificate through the web PKI hierarchy. An additional defense is to make it difficult for to discover which obfuscated URLs correspond to which content that an attacker wishes to spoof; this is achievable by design. A third defense would be to only accept updates for content from the same origin server that populated the cache with the original content.

**Timing Attacks.** An attacker who is passively observing traffic could potentially correlate requests and responses based on timing information. To address this type of attack, OCDN could employ techniques used in previous research, such as implementing timing delays, at different proxies (either client or exit proxies).

**Sybil Attacks.** An adversary who runs *many* exit proxies can learn information about many clients and many content requests as they are responsible for encrypting/decrypting *many* requests and responses. Previous work has analyzed the security of DHTs in this context [69, 177]. OCDN can employ a few different defenses to limit the probability and size of a Sybil attack. To limit the number of exit proxies running on a single machine, OCDN can limit the number of exit proxies with the same IP address (which is a part of the exit proxy's self-certifying identifier) to one; therefore, the attack becomes more expensive as the number of machines the adversary must

control increases.[5] This defense can be expanded to entire network prefixes; for example, if a large (malicious) organization owns an entire prefix, they could launch a Sybil attack using various IP addresses within their network. OCDN could limit the number of IP addresses in a given prefix to either one (which may result in a smaller set of exit proxies) or some small number (in which case the size of the Sybil is extremely small and cannot achieve its goal of being in a certain location on the hashing ring).

**Flashcrowds.** A flashcrowd is a large spike in traffic to a specific web page. An attacker could see that some content on the CDN has just seen a surge in traffic and correlate that with other information (for example, major world events). This leaks information about what content the CDN is caching. Fortunately, the design of OCDN can defend against this type of inference attack. The exit proxy can cache content in the time of a flashcrowd, such that the CDN (and therefore the attacker) does not see the surge in traffic.[6]

## 4.7 Performance Analysis

To evaluate how much overhead is caused by OCDN we measure the performance of OCDN. In addition to understanding the latency and overhead produced by the system, we also discuss the scalability of the design and show how OCDN scales well with an increasing number of clients.

[5]Note also that this countermeasure prevents two or more exit proxies from being behind the same NAT.

[6]This raises billing issues because the CDN can't charge as much if edge servers don't see as many requests for the origin; fortunately, RFC 2227 describes a solution for this [154].

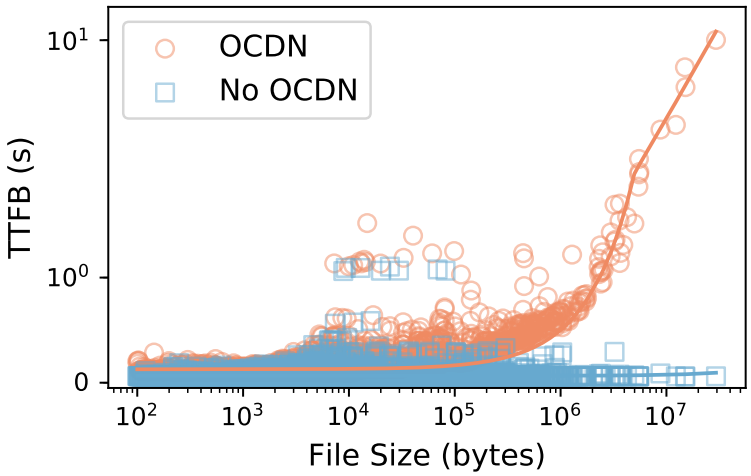



Figure 4.8: Time to First Byte measurements with and without OCDN.

### 4.7.1 OCDN Overhead

For measuring performance characteristics of OCDN, we use the implementation described in Section 5.4. Figure 4.7 shows how our measurements reflect OCDN (solid line) and a traditional CDN (dotted line).

Figure 4.8 shows the Time to First Byte (TTFB) for both OCDN and without OCDN. We can see the the TTFB using OCDN grows linearly with file size, whereas without OCDN TTFB remains fairly constant. Interestingly, we can see that there are some fixed time operations that OCDN performs, which is visible by looking at the smaller file sizes.

In addition to measuring TTFB, we measured the time it took to complete a request (with and without OCDN); the results are shown in Figure 4.9. Again, completion time grows linearly with file size, but for both OCDN and without OCDN; while both follow the same pattern, the time to complete requests is, as expected, longer using OCDN as it performs many cryptographic operations and proxies traffic between the client and the CDN.

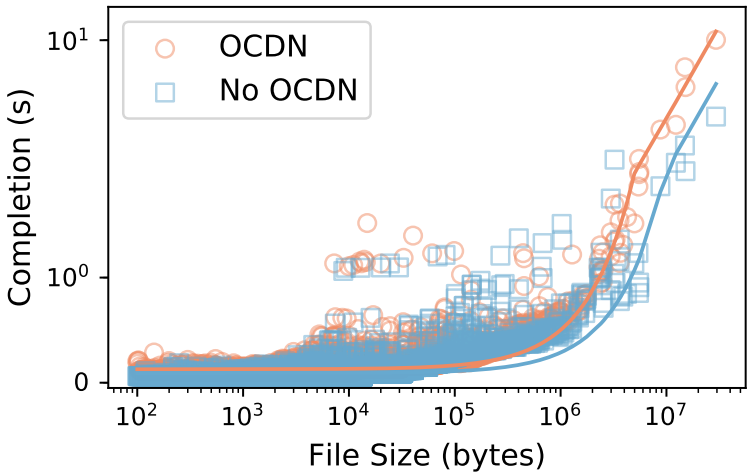



Figure 4.9: Time to complete a request with and without OCDN.

As described in Section 5.4, our prototype included only a single client, but our design allows for a client to proxy her request through additional clients. To simulate this, we add latency between the client and the exit proxy, and measure both the TTFB and time to complete a request when there are different values of latency, which represent different numbers of clients on the path between the original client and the exit proxy. Figure 4.10 shows the results for three different file sizes. The bottom portion of each bar in the graph shows the TTFB, and the top portion shows the additional time needed to complete the request. As expected, the TTFB grows much slower as file size and latency increase; completion time grows more quickly than TTFB as the file size and latency increase.

Finally, we measure the performance overhead of the individual operations used in OCDN; figure 4.11 shows the overhead of different components of the system for three different file sizes. We can see that some of the fixed cost/time operations include the client locally looking up the correct exit proxy to use for a given URL and the exit proxy generating the $\text{HMAC}_k(\text{URL})$. The operations that have the most overhead and continue to grow with the size of the file are the exit proxy decrypting the response

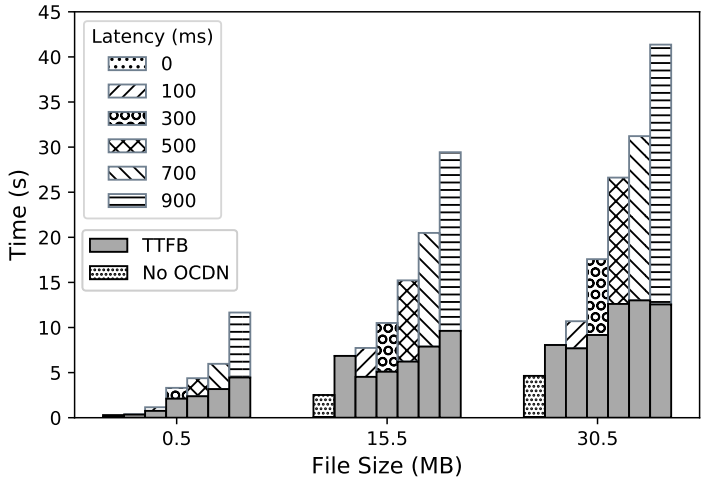



Figure 4.10: Time to First Byte and time to complete a request with varying the file size and latency; this latency correspondes to $\alpha$ in Figure 4.7.

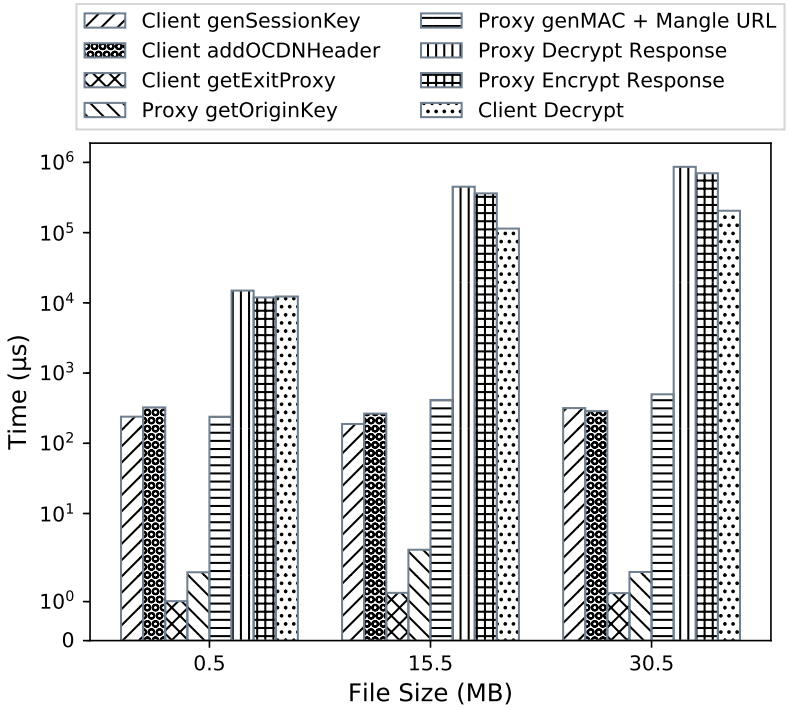



Figure 4.11: Overhead of different operations performed by OCDN.

with the shared key $k$, the exit proxy encrypting the response with the session key $k_{session}$, and the client decrypting the response with the session key $k_{session}$.

### 4.7.2 Scalability

For evaluating performance, we are also concerned with how well OCDN will scale with more users and more URLs. In particular, we need to reason about how much load is put on the exit proxies as the system grows; clients do not bare much load in the system as they simply proxy requests and the CDN is designed to handle high numbers of requests, therefore, we limit our scalability analysis to the exit proxies.

As previously mentioned, we balance load among the proxies by using consistent hashing to assign URLs to exit proxies. OCDN can additionally distribute load by replicating exit proxies, meaning that two exit proxies can have the same distributed hash table of shared keys. This way, both exit proxies can accept requests from clients for the URLs they are responsible for. Also worth noting is that the exit proxy is only recieving requests for the content corresponding to the shared keys it contains. Therefore, as the number of clients grow, the exit proxy is still only responsible for its set of shared keys and subsequent URLs. And as the number of URLs increase, the additional load per proxy is still low because of the load balancing properties of consistent hashing. We also discuss in Section 4.8 how clients can set up exit proxies; this will further decrease the load per exit proxy because each exit proxy will be responsible for a smaller number of shared keys/URLs.

## 4.8 Discussion

In this section, we discuss the various technical, political, and legal limitations of OCDN, as well as possible avenues for future work.

**OCDN limitations.** CDNs become slightly limited in terms of the possible performance optimizations when following OCDN's design. For example, many CDNs perform HTTPS re-writes on cached content, but this can only be done if the CDN has access to the decrypted content. Similarly, the CDN needs the decrypted content

to perform minimizations on HTML, CSS, and Javascript files. While this likely increases performance in traditional CDNs, it does not provide the greatest increase in performance; content caching around the world is the greatest benefit to performance, which OCDN preserves.

**CDNs operated by content hosts.** The design of OCDN assumes that the entities operating the proxies and delivering content are distinct from original content provider. In many cases, however—particularly for large content providers such as Netflix, Facebook, and Google—the content provider operates their own CDN cache nodes; in these cases, OCDN will not be able to obfuscate the content from the CDN operator, since the content host and the CDN are the same party. Similarly, because the CDN operator is the same entity as the original server, it also knows the keys that are shared with the clients. As a result, the CDN cache nodes could also discover the keys and identify both the content, as well as which clients are requesting which content.

**Legal questions and political pushback.** Recent cases in the United States raise some questions over whether a system like OCDN might face legal challenges from law enforcement agencies. For example, the rise of end-to-end encryption in messaging services (*e.g.* Signal, Threema, Telegram, Wire) has led to much controversy and debate between tech companies and law enforcement (particularly on the topic of "backdoors" to gain access to information) [145]. It remains to be seen whether OCDN would face similar hurdles, but similar systems in the past have faced scrutiny and pushback from law enforcement.

## 4.9 Related Work

To our knowledge, there has been no prior work on preventing surveillance at CDNs, but there has been relevant research on securing CDNs, finding security vulnerabilities in CDNs, and conducting different types of measurements on CDNs.

**Securing CDNs.** Most prior work on securing CDNs has focused on providing content integrity at the CDN as opposed to content confidentiality (and unlinkability). In 2005, Lesniewski-Laas and Kaashoek use SSL-splitting — a technique where the proxy simulates an SSL connection with the client by using authentication records from the server with data records from the cache (in the proxy) — to maintain the integrity of content being served by a proxy [113]. While their work does not explicitly apply SSL-splitting to CDNs, it is a technique that could be used for content distribution. Michalakis et al., present a system for ensuring content integrity for untrusted peer-to-peer content delivery networks [122]. Their system, Repeat and Compare, use attestation records and a number of peers act as verifiers. More recently, Levy et al., introduced Stickler, which is a system that allows content publishers to guarantee the end-to-end authenticity of their content to users [115]. Stickler includes content publishers signing their content, and users verifying the signature without having to modify the browser. Unfortunately, systems like Stickler do not protect against an adversary that wishes to learn information about content, clients, or publishers; OCDN is complementary to Stickler.

There has been prior work in securing CDNs against DDoS attacks; Gilad et al. introduce a DDoS defense called CDN-on-Demand [72]. They provide a complement to CDNs, as some smaller organizations cannot afford the use of CDNs and therefore do not receive the DDoS protections provided by them. CDN-on-Demand is a software defense that relies on managing flexible cloud resources as opposed to using

a CDN provider's service.

**Security Issues in CDNs.** More prevalent in the literature than defense are attacks on CDNs. Recent work has studied how HTTPS and CDNs work together (as both have been studied extensively separately). Liang et al., studied 20 CDN providers and found that there are many problems with HTTPS practice in CDNs [117]. Some of these problems include: invalid certificates, private key sharing, neglected revocation of stale certificates, and insecure back-end communications; the authors point out that some of these problems are fundamental issues due to the man-in-the-middle characteristic of CDNs. Similarly, Zolfaghari and Houmansadr found problems with HTTPS usage by CDNBrowsing, a system that relies on CDNs for censorship circumvention [187]. They found that HTTPS leaks the identity of the content being accessed, which defeats the purpose of a censorship circumvention tool.

Research has also covered other attacks on CDNs, such as flash crowds and denial of service attacks; Jung et al., show that some CDNs might not actually provide much defense against flash events (and they differentiate flash events from denial of service events) [98]. Su and Kuzmanovic show that some CDNs are more susceptible to intentional service degradation, despite being known for being resilient to network outages and denial of service attacks [158]. Additionally, researchers implemented an attack that can affect popular CDNs, such as Akamai and Limelight; this attack defeats CDNs' denial of service protection and actually utilizes the CDN to amplify the attack [163]. In the past year, researchers have found forwarding loop attacks that are possible in CDNs, which cause requests to be served repeatedly, which subsequently consumes resources, decreases availability, and could potentially lead to a denial of service attack [35].

Recently, researchers have studied the privacy implications of peer-assisted CDNs; peer-assisted CDNs allow clients to cache and distribute content on behalf of a website. It is starting to be supported by CDNs, such as Akamai, but the design of the paradigm makes clients susceptible to privacy attacks; one client can infur the cross site browsing patterns of another client [96].

**Measuring and Mapping CDNs.** As CDNs have increased in popularity, and are predicted to grow even more [182], much research has studied the deployment of CDNs. Huang et al., have mapped the locations of servers, and evaluated the server availability for two CDNs: Akamai and Limelight [90]. More recently, Calder et al., mapped Google's infrastructure; this included developing techniques for mapping, enumerating the IP addresses of servers, and identifying associations between clients and clusters of servers [31]. Scott et al., develop a clustering technique to identify the IP footprints of CDN deployments; this analysis also analyzes network-level interference to aid in the identification of CDN deployments [149]. In 2017, researchers conducted an empirical study of CDN deployment in China; they found that it is significantly different than in other parts of the world due to their unique economic, technical, and regulatory factors [181].

Other measurement studies on CDNs have focused on characterizing and quantifying flash crowds on CDNs [173], inferring and using network measurements performed by a large CDN to identify quality Internet paths [157], and measuring object size distributions and request characteristics to optimize caching policies [16].

# Chapter 5

# Naming: Privacy-Preserving DNS

Almost all communication on the Internet today starts with a Domain Name System (DNS) lookup. Before communicating with any Internet destination, a user application typically first issues a Domain Name System (DNS) lookup, which takes a domain name (*e.g.*, `google.com`) and returns an IP address that the application should contact. Today, the DNS requires the user to place tremendous trust in DNS operators, who can see all of the DNS queries that a user issues. Whether the operator is an Internet service provider (ISP) or some other third party is less concerning than the fact that some single operator can observe and retain this sensitive information. This chapter presents a system called ODNS which attempts to solve this problem.

Figure 5.1 illustrates the basic operation of DNS, as well as the privacy vulnerabilities that it introduces for every Internet user. When a client generates a request for `foo.com`, the client's stub resolver first contacts its recursive DNS resolver. Assuming that no DNS records are cached, the recursive sends the query to the root nameserver, which refers the recursive to the authoritative server for `.com`, which in turn returns a referral to the authoritative server for `foo.com`, which ultimately returns the IP address for the name.

DNS queries and responses are not encrypted. As a result, they can reveal significant information about the Internet destinations that a user or device is communicating with. For example, the domain names themselves may reveal the websites that a user is visiting. In the case of smart-home Internet of Things (IoT) devices, the DNS queries may reveal the types of devices in user homes. Previous work has also demonstrated that DNS lookups can identify the websites that a user is visiting, even when they are using an anonymizing service such as Tor [76]. The operator of a DNS resolver may also retain information about DNS queries and responses—including the IP addresses that query the domains and the DNS names that are queries.

A recursive DNS resolver can easily to link domain names with the IP addresses who are querying them. A third party who can observe communication between a client and a recursive resolver, a recursive resolver, or an authoritative server may be privy to various pieces of this information, depending on which part of the DNS query resolution they may see. Operators of recursive DNS resolvers may see individual IP addresses coupled with the fully qualified domain name that accompanies the query.

A user's Internet Service Provider (ISP) often operates the user's default recursive DNS resolver, giving the ISP potentially extraordinary access to DNS query information. To mitigate this risk, several entities, including Google, Cloudflare [175], and Quad9 [1] operate "open" recursive resolvers that anyone can use as an alternative to the ISP's DNS recursive resolver. Yet, when a user switches to an alternate DNS recursive resolver, the privacy problem isn't solved; rather, the user must trust the operator of the DNS recursive instead of the ISP. Essentially, the user must decide whether they trust their ISP or some other organization—some of which are even in the business of collecting data about users.

Other approaches have layered encryption on top of DNS. For example, DNS-over-TLS [81], DNS-over-DTLS [137], and DNS-over- HTTPS [86] send DNS queries over an encrypted channel, which prevents an eavesdropper from learning the contents of

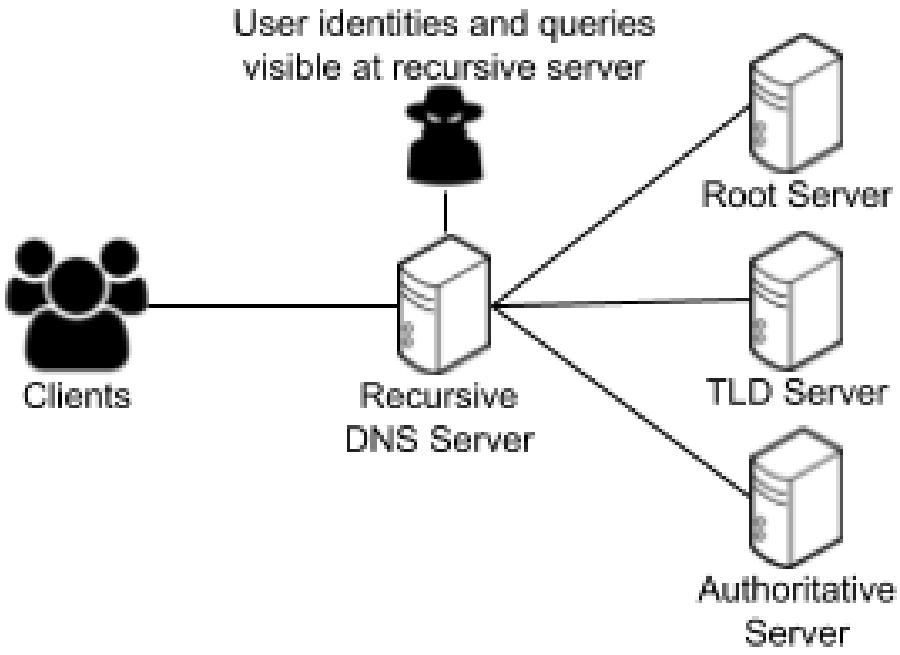



Figure 5.1: In a typical DNS lookup, a recursive resolver sees DNS queries and responses, as well as the IP addresses that issue the queries.

a DNS lookup but does not prevent the operator of the recursive resolver from linking queries and IP addresses. DNSCurve uses elliptic curve cryptography to encrypt DNS requests and responses; it also authenticates all DNS responses and eliminates any forged responses [17]. DNSCrypt encrypts and authenticates DNS traffic between a client and a recursive resolver [44]. Neither of these approaches prevent the recursive resolver from observing DNS queries and responses.

This work takes a different tack: Instead of merely shifting the trust anchor from an ISP to some other third party, we seek to prevent a recursive DNS resolver from ever associating IP addresses with the queries they make. To do so, we design, implement, and deploy Oblivious DNS (ODNS), which (1) obfuscates the queries that a recursive resolver sees from the clients that issue DNS queries; and (2) obfuscates the client's IP address from upper levels of the DNS hierarchy that ultimately resolve the query (*i.e.*, the authoritative servers). ODNS operates in the context of the existing DNS protocol, allowing the existing deployed infrastructure to remain unchanged. A

client sends an encrypted query to a recursive resolver, which then forwards the query to an authoritative DNS server that can resolve ODNS queries. The recursive resolver never sees the DNS domain that the client queries, and the ODNS server never sees the IP address of the client.

ODNS requires small changes to a client's stub resolver, as well as to the authoritative DNS server. The stub resolver must take an existing DNS name, encrypt it, and append the ODNS domain to ensure that the query is forwarded to the ODNS authoritative DNS server. The ODNS authoritative DNS server must also act as a recursive DNS resolver, ultimately retrieving the DNS record that corresponds to the client's initial query. We implemented this functionality in a prototype stub resolver and DNS authoritative server in Go. Our trace-driven evaluation shows that any individual uncached DNS lookup is slower by about 10% (about 14 ms on average), as a result of the cryptographic operations on the domain name. ODNS can take advantage of caching, however, which reduces performance overhead significantly in practice. Ultimately, the performance overhead on web page load times is negligible.

## 5.1 Background

We first take a look at how the existing DNS infrastructure is susceptible to traffic monitoring and data requests. Then later in this section we highlight some existing approaches and proposals and explain why they are not a complete solution to the problem we are addressing.

### 5.1.1 DNS

Figure 5.1 shows how conventional DNS operates and how the different entities communicate. When a client generates a request for `foo.com`, it gets sent to his recursive resolver. The recursive resolver may have a cached response for `foo.com`, and in that

case sends the response back to the client. If there is no cached response for `foo.com`, the recursive sends the query onto the root nameserver, which responds with the location of the TLD nameserver. The recursive then forwards the request onto the TLD nameserver, which responds with the location of the authoritative nameserver. The recursive sends the request to the authoritative nameserver, which responds with the location of `foo.com`. This response is then forwarded on to the client.

DNS queries may reveal information that an Internet user may want to keep private, including the websites that user is visiting and the IP address or IP subnet of the device that issued the initial query. A third party who can observe communication between a client and a recursive resolver, a recursive resolver, or an authoritative server may be privy to various pieces of this information, depending on which part of the DNS query resolution they may see. Operators of recursive DNS resolvers may see individual IP addresses (which may correspond to an ISP subscriber, or perhaps an individual end-device) coupled with the fully qualified domain name that accompanies the query. Even in the case of authoritative resolvers, extensions to DNS such as EDNS0 Client Subnet may reveal information about the user's IP address or subnet to authoritative DNS servers higher in the DNS hierarchy.

### 5.1.2 Existing Approaches

Many recent efforts protect certain, orthogonal aspects of privacy for users. However, none of these approaches prevent the operator of a DNS server from learning which IP addresses are issuing queries for particular domain names.

**DNS-over-TLS.** Conventional DNS queries are typically sent in clear text, which allows eavesdroppers to learn all the DNS lookups performed. To prevent such an eavesdropper, DNS-over-TLS was proposed, which sends DNS queries over an encrypted channel [81]. While this method prevents an eavesdropper from learning the

contents of a DNS lookup, it does not prevent a potentially malicious DNS operator from learning and logging all DNS lookups.

**DNS-over-DTLS.** This method is similar to the previously mentioned DNS-over-TLS, with the primary difference being that DNS-over-DTLS relies on UDP, whereas DNS-over-TLS relies on TCP [137].

**DNS-over-HTTPS.** Like DNS-over-TLS and DNS-over-DTLS, this approach provides confidentiality to DNS requests and responses, but uses a different transport medium (HTTPS) [86].

**Quad9.** This approach provides both security and privacy features for DNS. Quad9 uses IBM X-Force threat intelligence data at the recursive resolver to prevent a client from accessing a malicious site [1]. Although this recursive resolver does not store or distribute the DNS data passing through, it still allows a DNS operator to observe this data. Once such information is retained, of course, it may become vulnerable to other threats to user privacy, including data requests from law enforcement.

**1.1.1.1** Cloudflare recently released `1.1.1.1`, which is a privacy-first consumer DNS recursive resolver; it supports both DNS-over-TLS and DNS-over-HTTPS, and also offers the feature of query name minimization [175]. While `1.1.1.1` only logs data for 24 hours at the recursive resolver (for debug purposes), it is still susceptible to a malicious DNS operator (or other adversary) saving that information before it is purged. Furthermore, even though a DNS operator who aims to protect user privacy may purge this information periodically, a user has no guarantee that information that an operator learns might be retained, for operational or other purposes.

**DNSCurve.** This project improves DNS security by addressing message integrity and confidentiality, as well as DNS availability. DNSCurve uses elliptic curve cryptography to encrypt DNS requests and responses; it also authenticates all DNS responses (and eliminates any forged responses) [17]. While DNSCurve drastically improves the

security features of conventional DNS, it does not prevent surveillance or monitoring conducted at the recursive resolver.

**DNSCrypt.** DNSCrypt is another approach that encrypts and authenticates DNS traffic between a client and a recursive resolver [44]. It also provides additional security features, such as preventing DNS spoofing. Like the other existing approaches, DNSCrypt does not prevent monitoring of DNS traffic at the recursive resolver.

## 5.2 Design

This section describes the design of ODNS, which operates seamlessly in the context of the existing DNS. ODNS defends against an adversary that can (1) eavesdrop on communications between clients and recursive resolvers, and between recursive resolvers and authoritative name servers; (2) request data (via subpoena/warrant) from any number of DNS operators; or (3) access data and logs (e.g., query logs) at any DNS server.

### 5.2.1 Overview

Figure 5.2 summarizes the design. ODNS operates similarly to conventional DNS, but adds two components: (1) each client runs a modified stub resolver; and (2) ODNS runs an authoritative name server that also acts as a recursive DNS resolver for the original DNS query:

- The client's stub resolver obfuscates domain that the client is requesting (via symmetric encryption), resulting in the recursive resolver being unaware of the requested domain.

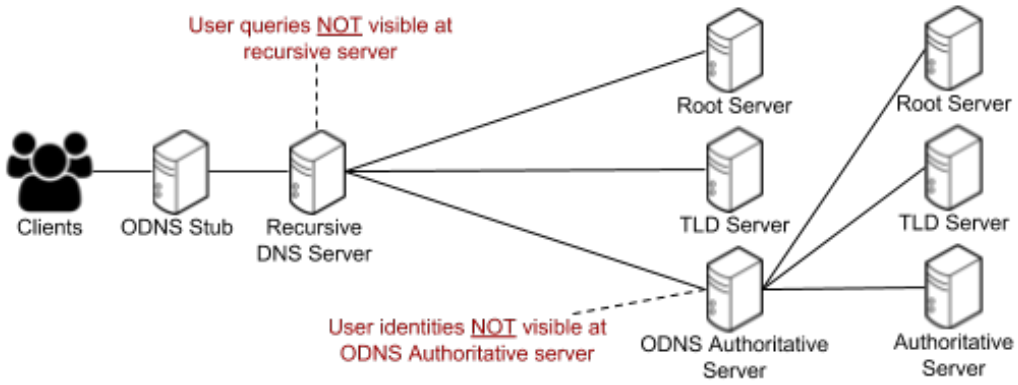



Figure 5.2: Overview of interacting components in ODNS.

- The authoritative name server for ODNS separates the clients' identities from their corresponding DNS requests, such that the name servers cannot learn who is requesting specific domains.

Operators of recursive DNS resolvers see individual IP addresses and with the fully qualified domain name that accompanies the query. Operators of authoritative resolvers may also be able to learn information about the client by using one of the extensions to DNS, such as EDNS0 Client Subnet. EDNS0 can reveal information about the user's IP address or subnet to authoritative DNS servers higher in the DNS hierarchy (not only recursive DNS resolvers). ODNS hides a client's IP address from the authoritative name servers at different levels of the DNS hierarchy.

The recursive DNS resolver knows the client IP address but never sees the domain that it queries. ODNS requires the client to use a custom local stub resolver, which hides the requested domain from the recursive resolver. The ODNS stub resolver, which runs at the client, encrypts the original DNS query for the ODNS authoritative DNS server before it appends the `.odns` domain to the query, which causes the recursive resolver to forward the encrypted domain name on to the ODNS authoritative server.

When an ODNS authoritative DNS server receives a DNS query, it removes any client information from the request (*e.g.*, the client IP address, ENDS0 client subnet

information) before performing additional DNS lookups. The `.odns` name server then acts as a recursive resolver.[1] The authoritative server forwards any response to the original recursive DNS resolver, which in turn sends the response to the client.

The recursive DNS resolver receives the request from the client, but cannot identify the genuine domain. It parses the TLD (`.odns`) and forwards the request onto the `.odns` authoritative server. Because the session key was originally encrypted with the authoritative server's public key, the authoritative server can decrypt the session key with its private key, and subsequently decrypt the domain with the session key. The authoritative server then acts as a recursive resolver and contacts the necessary name servers to resolve the domain. Once an answer is obtained, the authoritative server encrypts the domain with the session key, appends the `.odns` TLD and forwards the response to the recursive DNS resolver. As explained by the use of session keys, the recursive resolver cannot learn the domains a client requests, despite being able to learn who the client is.

### 5.2.2 ODNS Protocol

Figure 5.3 shows the various steps involved in answering a client's DNS request, which proceeds as follows:

1. When a client generates a DNS request, the local stub resolver generates a symmetric session key, encrypts the domain name with the session key, encrypts the session key with the authoritative server's public key, and appends the `.odns` TLD to the encrypted domain. ($\{\texttt{www.foo.com}\}_k\texttt{.odns}$.) The stub also appends the session key encrypted under the `.odns` authoritative server's public key ($\{k\}_{PK}$)

[1] For simplicity, we say that this authoritative server is for `.odns` domains, but any authoritative DNS domain can run an ODNS server.

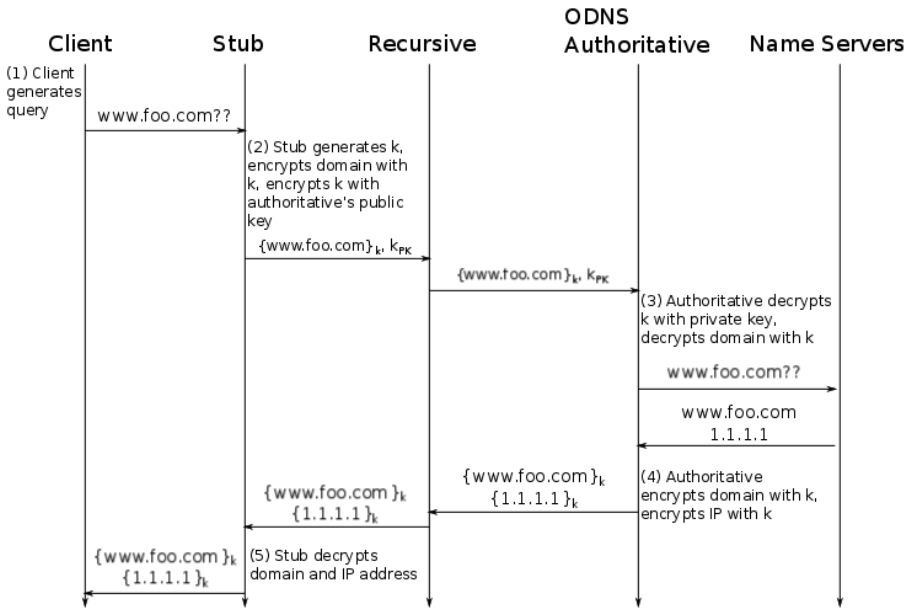



Figure 5.3: ODNS protocol.

2. The client sends the query in the Additional Information portion of the DNS query to the recursive resolver, which then sends it to the authoritative name server for `.odns`.

3. The authoritative server for ODNS queries decrypts the session key, which it uses to decrypt the domain in the query.

4. The authoritative server forwards a recursive DNS request to the appropriate name server for the original domain, which then returns the answer to the ODNS server.

5. The ODNS server returns the answer to the client's recursive resolver.

Other authoritative DNS servers see incoming DNS requests, but these only see the IP address of the ODNS authoritative resolver, which effectively proxies the DNS request for the original client. The client's original recursive resolver can learn the client's IP address, but cannot learn the domain names in the client's DNS queries.

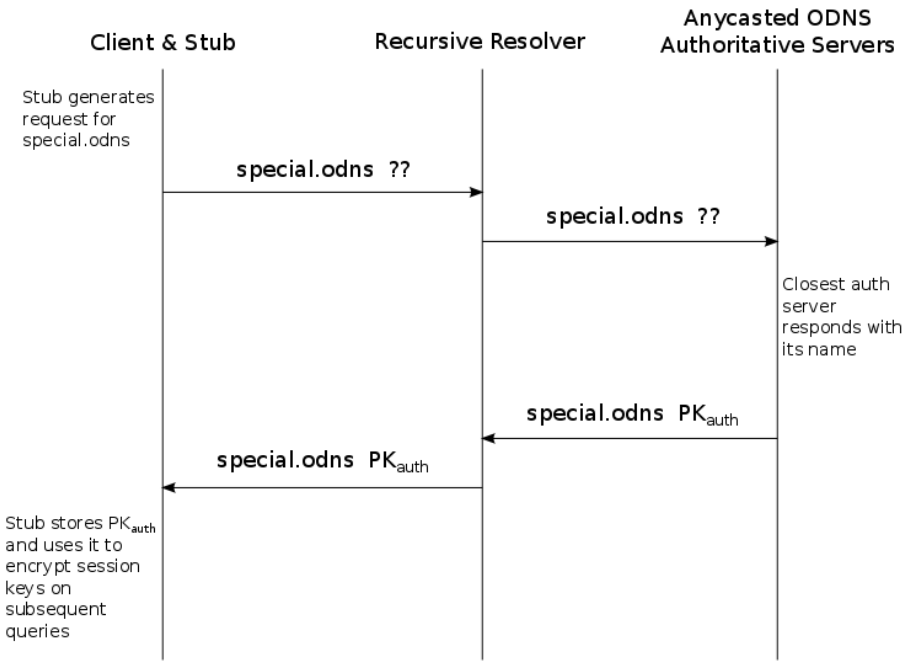



Figure 5.4: ODNS protocol for key distribution and selecting the optimal authoritative server.

**Georeplication and key distribution.** ODNS supports georeplication. To direct clients to the closest ODNS server instance ODNS's authoritative servers have an anycast IP address and a unique IP address. The client's stub resolver sends a DNS query to the anycast IP address; the response contains the authoritative server's public key IP address. Figure 5.4 summarizes this process in more detail. Specifically, the client's stub resolver sends a DNS query for `special.odns`. The recursive resolver forwards the the query to the authoritative name server, which is anycasted. The name server responds to the DNS query with a self-certifying name, such that the name of the server is derived from the public key itself; this response is returned to the client's stub resolver via the recursive.

## 5.3 Practical Challenges

This section describes specific modifications to the basic ODNS protocol to cope with various practical limitations and considerations.

### Protocol Constraints

**OPT records and query length.** In principle, a query could include the encrypted session key in a special Resource Record (RR) in the "Additional Information" section of the DNS message (known as an OPT), but we discovered that, in practice, most open resolvers strip all OPT records before forwarding the query on to the nameserver; in this case, ODNS cannot simply use an OPT to communicate the session key. ODNS overcomes this challenge by placing the encrypted key in the QNAME field of the DNS message; the QNAME field consists of 4 sets of 63 bytes, which limits both the key size and encryption scheme used. For this reason, ODNS uses 16-byte AES session keys and encrypts the session keys using the Elliptic Curve Integrated Encryption Scheme (ECIES). Once the session key is encrypted, the resulting value takes up 44 bytes of the QNAME field. In the future, if OPT records are used by open resolvers, the encrypted session key can be sent in that record, which would allow different key sizes and encryption schemes to be used.

**EDNS0 Client Subnet.** our initial experiments, we found that Google's open resolvers use EDNS0 client subnet, thereby revealing the client's identity to the authoritative server. We did not observe other open resolvers (such as Cloudflare's `1.1.1.1` or Quad9's `9.9.9.9`) support the addition of the EDNS client subnet. Therefore, clients wishing to get the privacy benefits of ODNS should not use Google's open resolvers in conjunction with ODNS.

### 5.3.1 Performance

**Caching.** ODNS cache queries at the ODNS authoritative server. As the ODNS authoritative server is essentially an extra recursive resolver in the system, ODNS is simply acts as the shared cache, in lieu of the shared cache that would ordinarily exist at the ISP recursive resolver. further into the system; this decreases the time to conduct a look up for the domain from the perspective of the authoritative server. In addition to caching at the authoritative server, ODNS could also cache DNS responses at the stub resolver; while this may provide some caching benefits on a per- client basis, it does not provide cross-client caching benefits. It is important to note that caching at the recursive resolver or the client's stub local resolver would not provide any performance enhancement for the client in this system. Each DNS query is encrypted with a new session key $k$, thus two DNS queries for the same domain do not appear the same to these resolvers ($\{\texttt{www.foo.com}\}_{k1} \neq \{\texttt{www.foo.com}\}_{k2}$); therefore, if the resolver cached the response for $\{\texttt{www.foo.com}\}_{k1}$, it would never see a cache hit for that entry because subsequent lookups for `www.foo.com` appear as a different URL.

**Anycast.** As mentioned in Section 5.2.2, the ODNS authoritative server is replicated in a variety of geographical locations and are anycasted. Therefore, when a recursive sends a DNS query to the ODNS authoritative server, it will be sending it to the closest ODNS authoritative server. And because the recursive resolver (an open resolver) is also anycast, both the recursive and the ODNS authoritative server will be selected based on the client's location *without revealing the client's location.* While this has a privacy benefit, it also exhibits a performance benefit; using anycast authoritative servers and recursive resolvers allows a client to use the closest (and likely fastest) servers.

### 5.3.2 Privacy & Security

**Striping queries across multiple recursives.** The client's local stub resolver typically forwards DNS queries on to the client's recursive resolver, but ODNS supports forwarding the DNS queries on to open resolvers. ODNS can stripe DNS queries across the many available open resolvers, which helps increase the privacy of the client because the recursive resolver does not see all (*obfuscated*) DNS queries from the client. If striping is enabled, then each open resolver only sees some portion of each client's obfuscated queries.

**Denial of service attacks.** ODNS's authoritative DNS servers cannot check the incoming IP address of queries, which could facilitate Denial of Service attacks. To defend against DoS attacks, the client's stub resolver can append bytes that indicate the DNS query is sent from an ODNS-participating client stub resolver. The authoritative can then check for these bytes and verify that it was sent via ODNS.

## 5.4 Implementation

We have designed ODNS to decouple any DNS query from the IP address that initiated the query, while allowing the existing DNS infrastructure to remain unchanged. We implement a prototype to evaluate the feasibility of ODNS by implementing a stub resolver and an authoritative name server. ODNS simply runs on top of conventional DNS, and no changes are made to the underlying DNS infrastructure. Figure 5.5 shows the details of our prototype implementation, and we detail the ODNS components as follows.

**Client.** In our prototype, the client is a machine running in the New York City area of the United States. It is used to perform our evaluation in Section 5.5 and conducts DNS lookups for different domains.

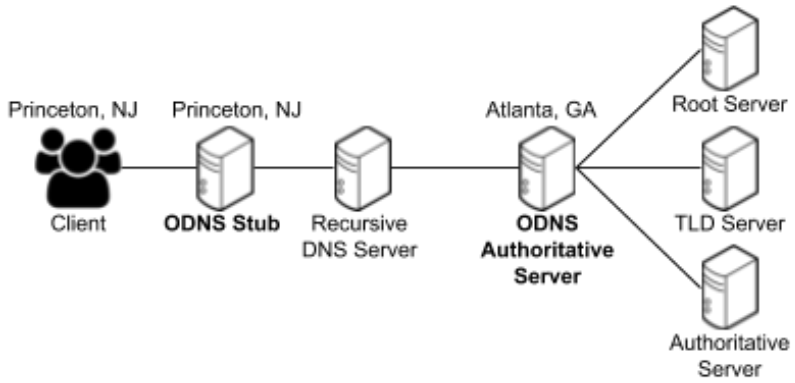



Figure 5.5: Prototype setup.

**Stub Resolver.** We implemented a stub resolver that runs on each participating client. The stub resolver is written in golang and performs the following actions. The stub resolver is running on the same local machine as the client in our prototype implementation in the New York City area. When the stub generates a session key, the key that results is a 16 byte AES symmetric key. The stub encrypts the session key with the authoritative's public key, which is generated with ECIES using curve25519.

**Recursive Resolver.** ODNS requires no changes to the recursive resolver, and therefore, our prototype uses the default recursive resolver for the client located at a university.

**ODNS Authoritative Server.** We implemented two authoritative servers for `obliviousdns.com`, which is written in golang (similar to the stub resolver). This server essentially proxies a client's DNS query by acting as a recursive resolver once the domain is decrypted. One of the authoritative servers is running on a Virtual Private Server (VPS) located in Georgia and the other is located in New York City. Each authoritative server has a public/private key pair generated with ECIES using curve25519.

**Name Servers.** ODNS requires no changes to the name servers, and therefore uses the existing DNS infrastructure for sending and receiving queries/responses from name servers.

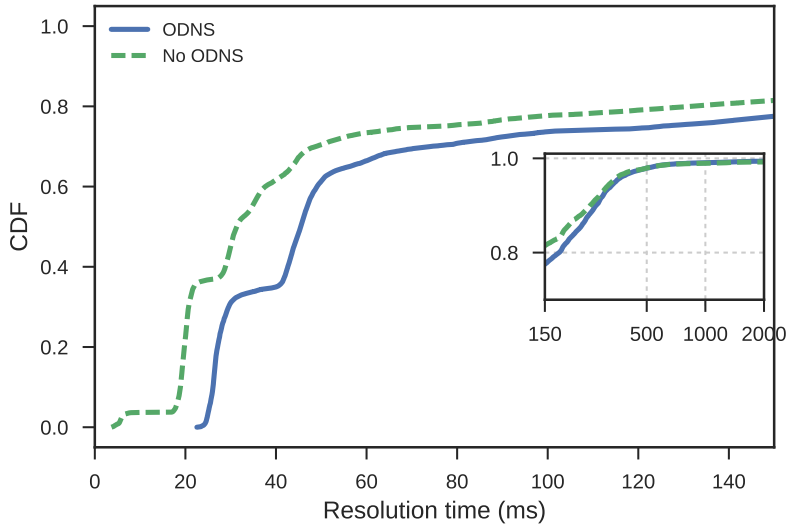



Figure 5.6: ODNS overhead. The median resolution time for ODNS is 14.1 milliseconds.

## 5.5 Performance Evaluation

We used the prototype setup described in Section 5.4 to evaluate the performance overhead of ODNS in comparison to conventional DNS. It is important to note that ODNS will incur some overhead due to the cryptographic operations it performs in addition to the latency added by using additional resolvers, but ODNS also provides protections that conventional DNS does not; these protections come at some cost, which we explore more in this section.

### 5.5.1 Microbenchmarks: DNS Query Overhead

We first evaluated ODNS's query overhead to that of conventional DNS by using the `dig` command to issue DNS queries. We ran DNS queries on the Alexa Top 10,000 domains using both ODNS and conventional DNS. The results of this are shown in Figure 5.6; we can see that ODNS takes longer to resolve domains, but follows the same trend as conventional DNS. Based on these results, ODNS does not appear to introduce significant overhead when resolving domains.

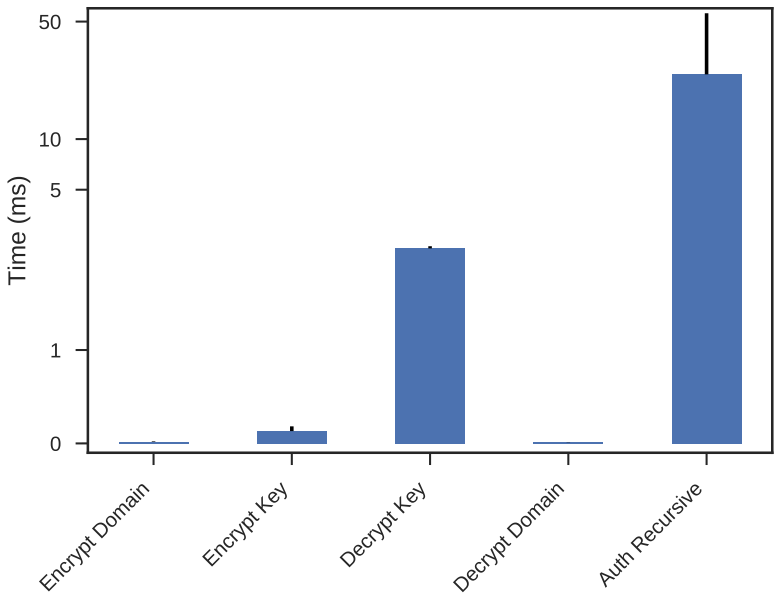



Figure 5.7: Overhead of different operations performed in the ODNS protocol.

After looking at the total overhead of ODNS, we benchmark the individual operations performed in the ODNS protocol to attribute overhead to different parts of the protocol. The results of these microbenchmarks are shown in Figure 5.7. We can see that the cryptographic operations, and particularly the asymmetric cryptographic operations, take the most time and therefore contribute the most to the total overhead of ODNS.

Our implementation includes two authoritative servers — functionally equivalent — in two different locations: Georgia and New York City. Recall that the client and stub resolver are also located in the New York City area. To evaluate the effect of the authoritative server's geographic proximity to the client, we measured the overhead using each of the authoritative servers. In this experiment, we used third party resolvers upstream from the ODNS authoritative.[2] Our results can be seen in Figure 5.8. Clearly, the overhead is significantly less when the authoritative server is closer to the client; the average overhead when using the authoritative server in New York City is about 5ms and the overhead when using the authoritative server in Georgia is

[2]This allows ODNS to gain the caching benefits of upstream resolvers.

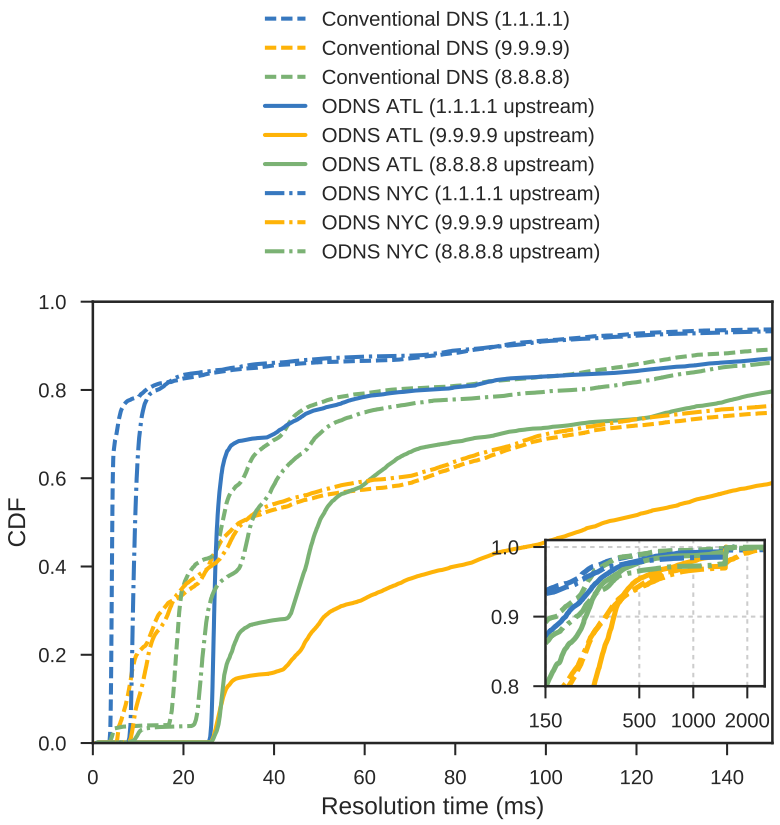



Figure 5.8: Overhead of ODNS on DNS queries using an authoritative server in Georgia and an authoritative server in New York City.

about 14ms. The additional overhead is consistent with the round-trip time between the client in New York City and the respective authoritative server.

### 5.5.2 Macrobenchmarks: Page Load Time

Next, we took a step back and looked at how ODNS would affect a typical Internet user's browsing experience by evaluating the overhead of a full page load; a full page load consists of not only conducting a DNS lookup for the page, but also fetching the page, and conducting any subsequent DNS lookups for embedded objects and resources in the page. We fetched popular web pages that have a lot of content using ODNS and then also using conventional DNS, and the results are shown in Figure 5.9. The left bar in the figure is using conventional DNS and the right bar represents

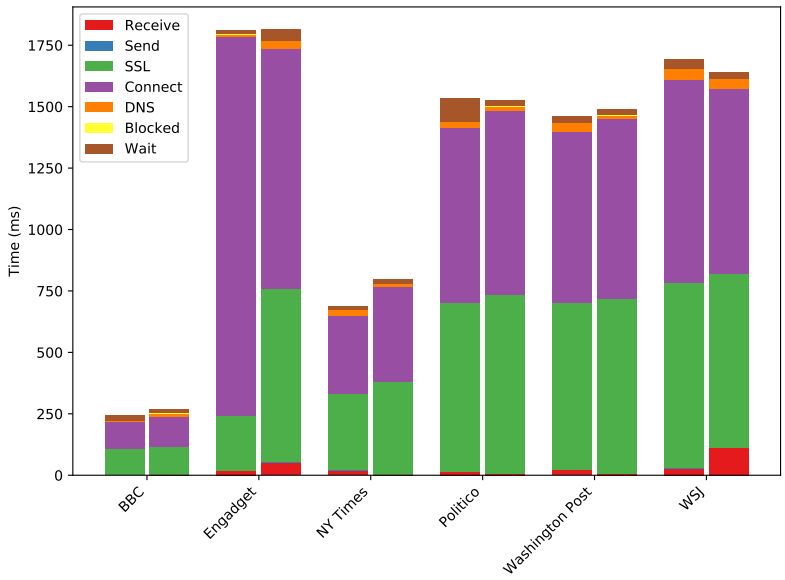



Figure 5.9: Page load time for various web pages using ODNS and conventional DNS. The left bar in the figure is using conventional DNS and the right bar represents the time it takes using ODNS.

the time it takes using ODNS. We see that there is no significant difference in page load time between ODNS and conventional DNS because DNS lookups contribute very little time to the entire page load process.

### 5.5.3 Effect of Caching

An important performance aspect of conventional DNS is caching — specifically caching DNS responses at the recursive resolver. And as mentioned previously, ODNS prevents any caching benefits at the recursive resolver, but allows caching at both the stub resolver and at the authoritative server. To evaluate the effect of caching on ODNS's performance, we measured the overhead of ODNS where the authoritative does not cache and acts as a second recursive resolver, and compared it to the overhead of ODNS using upstream recursive resolvers and their associated caches. Figure 5.10 shows the results.

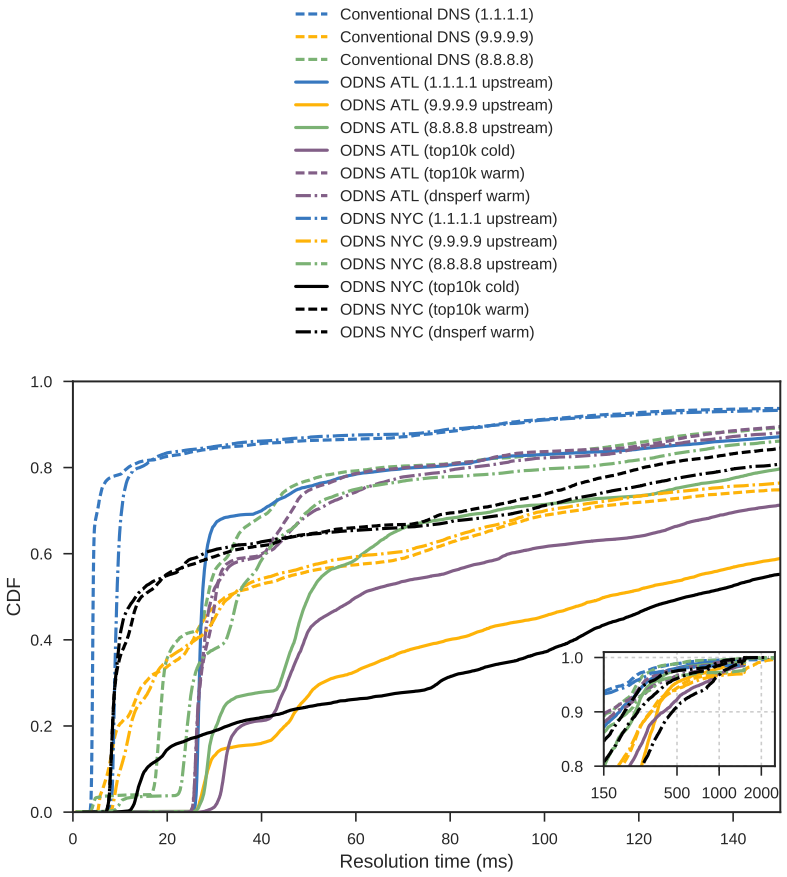



Figure 5.10: Overhead of ODNS with varying upstream caches.

## 5.6 Related Work

There's been a multitude of work on privacy and security as it pertains to DNS; here we highlight some of that work, as well as some other existing and related tools.

**DNS Privacy.** Quad9 provides both security and privacy features for DNS. Quad9 uses IBM X-Force threat intelligence data at the recursive resolver to prevent a client from accessing a malicious site [1]. Although this recursive resolver does not store or distribute the DNS data passing through, it still allows a DNS operator to observe this data. Once such information is retained, of course, it may become vulnerable to other threats to user privacy, including data requests from law enforcement. Cloudflare recently released `1.1.1.1`, which is a privacy-first consumer DNS recursive resolver; it supports both DNS-over-TLS and DNS-over-HTTPS, and also

offers the feature of query name minimization [175]. While `1.1.1.1` only logs data for 24 hours at the recursive resolver (for debug purposes), it is still susceptible to a malicious DNS operator (or other adversary) saving that information before it is purged. Furthermore, even though a DNS operator who aims to protect user privacy may purge this information periodically, a user has no guarantee that information that an operator learns might be retained, for operational or other purposes.

EncDNS protects the confidentiality of DNS messages in transit and from a DNS operator [82]; however, EncDNS suffers from several shortcomings, including the inability to share keys without using an out of band method directly, resulting in a breach of the privacy characteristics that the protocol promises.[3] In contrast to ODNS and EncDNS, most proposed DNS privacy mechanisms are protecting against an adversary, but not a DNS operator. For example, Zhu et al., propsoe T-DNS, which applies TCP and TLS to DNS [186], and Castillo-Perez and Garcia-Alfaro evaluate privacy-preserving DNS mechanisms, but show that they need additional measures to enhance their security [32]. Similarly, Query Name Minimization is a proposal that limits what name servers see in DNS queries, but a recursive resolver's operator still learns the domain requested and the corresponding client who requested the domain [19]. Researchers have also pointed out how aspects of current (operational) DNS, such as prefetching, have privacy implications [107,153]. Federrath et al. introduced a DNS anonymity service that employs broadcasting popular hostnames and low-latency mixes for requesting less popular domains; unlike ODNS, this proposed DNS anonymity service is a clean-slate architecture and requires fundamental changes to the DNS infrastructure [66].

**DNS Security.** There have been a variety of proposals for securing DNS, including DNS-over- TLS [81] and DNSSEC [111]. As mentioned in Section 5.1.2,

[3]Additionally, EncDNS code has been made publicly available for anyone to use; however, in our analysis, we found that the implementation does not work, and that a client's public key is sent directly within a DNS query, identifying the client to both the recursive and authoritative servers.

DNS-over-TLS protects the privacy of the domain being requested in transit, but does not prevent a recursive resolver's operator from learning both the client who issued the request and the content of the request. Some work has analyzed DNSSEC in more detail; Osterweil et. al., develop SecSpider to monitor and detect errors in the DNSSEC deployment [127]. While there have been many attacks on DNS, the adoption of DNS security protocols is very limited; Herzberg and Shulman highlight some of the issues with retrofitting security into DNS [83]. Researchers have also combined DNSSEC features with BIND DNS software to implement a system that prioritizes the integrity and availability of DNS [95]. Recent research has also seen the introduction of new frameworks for monitoring DNS in the hopes of detecting attacks [84, 119].

**Virtual Private Networks and Tor.** There have been many proposed anonymity systems and tools, including Virtual Private Networks (VPNs) and Tor. Unfortunately, many VPN providers send unencrypted DNS queries to the client's ISP, and all VPN providers would be able to associte DNS queries and responses with an individual client IP address. VPNGate uses different VPNs to contol where a client *appears* to be located [124], but even this approach does not prevent operators of DNS recursive resolvers from learning the domains being requested. The VPN operators themselves also still have complete information about the domains and the IP addresses that are querying them.

Tor uses layered encryption and a three-hop circuit to provide client anonymity [46]. Using Tor would not provide the same protections as ODNS because DNS is conducted at the last hop of the three-hop circuit and is conducted however that machine is configured. Additionally, the owner and operator of that machine (as well as the local resolver's operator) can learn what content is being requested. More recently, researchers have analyzed how DNS works in Tor and found that fingerprinting attacks can be performed based on DNS data [75]. The Tor Project has also designed

and implemented onion services/domains (previously hidden services), which provide server anonymity; onion domain lookup does not usew DNS, but can only be accessed via Tor and suffers from usability issues [8]. Recent work has highlighted how onion domain name leakages are a source of privacy leakage as well [162].

# Chapter 6

# Conclusion

In this dissertation, we explored how different aspects of the Internet infrastructure are susceptible to trafffic monitoring and data requests. In light of this vulnerability, we propose new techniques and systems that enhance the privacy of content and communications by protecting data in transit and at rest; these systems are designed such that an eavesdropper cannot learn information and such that an overreaching governing entity cannot issue data requests. Specifically, we analyze the routing, hosting, and naming architectures and introduce systems that increase privacy while using the underlying infrastructure, resulting in systems that are currently deployable.

**Routing.** We measured the current state of Internet routing, and in particular, which countries (and corresponding governing entities) are on the path to popular websites. We developed two techniques to route Internet traffic around an unfavorable country, and incorporated one of the techniques into RAN.

**Hosting.** Focusing on CDNs, we used encryption techniques, self-certifying identifiers, and consistent hashing to design OCDN. OCDN operates in the current ecosystem of users, CDNs, and content publishers, while protecting both users and CDNs from traffic monitoring and data requests.

**Naming.** We first analyzed how conventional DNS allows DNS operators to learn important user information, leading us to take a new approach to naming. We introduced ODNS as a way to decouple clients' identities from their DNS queries; it runs on top of conventional DNS and can be deployed today.

## 6.1 Future Work

As this work has focused on introducing privacy infrastructure for three specific components of the Internet architecture, there are many avenues of future work for other infrastructure components. For example, there's a need for future work in data privacy in the context of content providers; much of this work has had the goal of protecting users and operators, but how can we also protect providers?

Another line of future work can improve the accuracy and completeness of the systems proposed in this dissertation. With the cooperation of different entities, these systems can be further improved, both from a privacy perspective, but also a performance perspective. As an example, RAN would be greatly improved with the participation of content providers because they have a unique vantage point in the routing infrastructure. Content providers could provide more accurate information on reverse paths, as well as optimize for the quickest path that does not traverse an unfavorable region. Another example involves IXPs' participation, which could provide more route diversity in RAN.

The systems proposed in Chapters 3, 4, and 5, have been prototyped and evaluated, but should be implemented and deployed at a larger scale in the future. Evaluating these systems with large numbers of users can provide insights that will be helpful in the design of other related systems; some of these insights will be purely technical, such as unforeseen scalability issues, but they could also shed light on which entities might target these types of systems for an attack (and how they might attack

these systems). A large-scale deployment of this type of system will also potentially provide some of the first political responses (and push-back) to technical systems.

As briefly discussed in Section 2.4, there is a variety of work that needs to be done at the intersection of data privacy and physical data location. Ongoing court debates are attempting to decide who can request access to data, when it is located in a different jurisdiction, and when it is data about a client located in a different jurisdiction. OCDN and ODNS are a first attempt at addressing this issue by preventing any entity from requesting access to the data at a CDN and a recursive resolver, respectively. More specifically, these systems ensure that the entity who holds the data (*i.e.*a CDN or DNS operator) cannot provide a meaningful response to data requests.

## 6.2 Final Remarks

As emerging technologies continue to grow, evolve, and become ubiquitous in our everyday lives, we must complement them with other technologies and infrastructure that preserves and protects users' privacy. As the courts are currently debating the policies that govern users' private data, technologists can — and should — design and develop technology that can help shape future policy debates.

# Bibliography


[1] Quad9. https://quad9.net/, 2018.

[2] Josh Aas. Let's Encrypt: Delivering SSL/TLS Everywhere. 2014.

[3] Akamai Empowers Operators to Deploy Their Own Content Distribution Network. https://www.akamai.com/us/en/resources/content-distribution-network.jsp.

[4] Akamai: Facts & Figures. https://www.akamai.com/us/en/about/facts-figures.jsp.

[5] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *ACM Symposium on Operating Systems Principles (SOSP)*, volume 35. ACM, 2001.

[6] Anne Edmundson and Paul Schmitt and Nick Feamster and Jennifer Rexford and Allison Mankin. ODNS: Oblivious DNS. https://indico.dns-oarc.net/event/28/contributions/522/attachments/468/773/ODNS_slides.pdf, 2018.

[7] Anonymous. The collateral damage of internet censorship by dns injection. *ACM SIGCOMM CCR*, 42(3), 2012.

[8] Jacob Appelbaum and Alec Muffett. The '.onion' special-use domain name. 2015.

[9] Noah Apthorpe, Dillon Reisman, and Nick Feamster. A Smart Home is No Castle: Privacy Vulnerabilities of Encrypted IoT Traffic. *arXiv preprint arXiv:1705.06805*, 2017.

[10] Assessment of the Impact of Internet Exchange Points – Empirical Study of Kenya and Nigeria. http://www.internetsociety.org/sites/default/files/Assessment%20of%20the%20impact%20of%20Internet%20Exchange%20Points%20%E2%80%93%20empirical%20study%20of%20Kenya%20and%20Nigeria.pdf.

[11] Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman, Matthieu Latapy, Clémence Magnien, and Renata Teixeira. Avoiding

Traceroute Anomalies with Paris Traceroute. In *The 6th ACM SIGCOMM Internet Measurement Conference*, pages 153–158. ACM, 2006.

[12] A Baker's Dozen, 2015 Edition. http://research.dyn.com/2016/04/a-bakers-dozen-2015-edition/.

[13] Balancing Child Protection and Digital Rights. https://medium.com/@jmalcolm/balancing-child-protection-and-digital-rights-1b9c4ab0b93f.

[14] Suman Banerjee, Timothy G Griffin, and Marcelo Pias. The Interdomain Connectivity of PlanetLab Nodes. In *Passive and Active Network Measurement*, pages 73–82. Springer, 2004.

[15] Paul Barford and Mark Crovella. Generating representative web workloads for network and server performance evaluation. In *ACM SIGMETRICS Performance Evaluation Review*, volume 26, pages 151–160. ACM, 1998.

[16] Daniel S Berger, Ramesh K Sitaraman, and Mor Harchol-Balter. Adaptsize: Orchestrating the hot object memory cache in a content delivery network. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 483–498. USENIX Association, 2017.

[17] Daniel J Bernstein. Dnscurve: Usable security for dns. *dnscurve. org*, 2009.

[18] Zachary S Bischof, John P Rula, and Fabián E Bustamante. In and Out of Cuba: Characterizing Cuba's Connectivity. In *ACM Conference on Internet Measurement Conference*, pages 487–493. ACM, 2015.

[19] S. Bortzmeyer. DNS Query Name Minimisation to Improve Privacy. Work in progress (Internet draft draft-bortzmeyer-dns-qname-minimisation-02), May 2014.

[20] Brazil Builds Internet Cable To Portugal To Avoid NSA Surveillance. http://www.ibtimes.com/brazil-builds-internet-cable-portugal-avoid-nsa-surveillance-1717417.

[21] Brazil Conference will Plot Internet's Future Post NSA Spying. http://www.reuters.com/article/us-internet-conference-idUSBREA3L1OJ20140422.

[22] Brazil Looks to Break from US Centric Internet. http://www.nbcnews.com/technology/amp/brazil-looks-break-us-centric-intenet-4B11180299.

[23] Brazil Looks to Break from US Centric Internet. http://news.yahoo.com/brazil-looks-break-us-centric-internet-040702309.html.

[24] Brazil to Host Global Internet Summit in Ongoing Fight Against NSA Surveillance. https://www.rt.com/news/brazil-internet-summit-fight-nsa-006/.

[25] Brazil's President Tells U.N. That NSA Spying Violates Human Rights. http://www.usnews.com/news/articles/2013/09/24/brazils-president-tells-un-that-nsa-spying-violates-human-rights.

[26] Brazil to Press for Local Internet Data Storage After NSA Spying. https://www.rt.com/news/brazil-brics-internet-nsa-895/, 2013.

[27] Brasil Internet Exchange Participants Diversity. http://ix.br/doc/nic.br.ix.br.euro-ix-27th-berlin.20151027-02.pdf, 2015.

[28] Samuel Henrique Bucke Brito, Mateus AS Santos, Ramon dos Reis Fontes, Danny A Lachos Perez, and Christian Esteve Rothenberg. Dissecting the largest national ecosystem of public internet exchange points in Brazil. In *International Conference on Passive and Active Network Measurement*, pages 333–345. Springer, 2016.

[29] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. Touching from a distance: Website fingerprinting attacks and defenses. In *ACM Conference on Computer and Communications Security*, pages 605–616. ACM, 2012.

[30] CAIDA: Center for Applied Internet Data Analysis. http://www.caida.org/home/.

[31] Matt Calder, Xun Fan, Zi Hu, Ethan Katz-Bassett, John Heidemann, and Ramesh Govindan. Mapping the expansion of Google's serving infrastructure. In *Conference on Internet Measurement Conference*, pages 313–326. ACM, 2013.

[32] Sergio Castillo-Perez and Joaquin Garcia-Alfaro. Evaluation of two privacy-preserving protocols for the DNS. In *Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference on*, pages 411–416. IEEE, 2009.

[33] Cedexis. https://www.cedexis.com/.

[34] Josiah Chavula, Nick Feamster, Antoine Bagula, and Hussein Suleman. Quantifying the Effects of Circuitous Routes on the Latency of Intra-Africa Internet Traffic: A Study of Research and Education Networks. In *International Conference on e-Infrastructure and e-Services for Developing Countries*, pages 64–73. Springer, 2014.

[35] Jianjun Chen, Jian Jiang, Xiaofeng Zheng, Haixin Duan, Jinjin Liang, Kang Li, Tao Wan, and Vern Paxson. Forwarding-loop attacks in content delivery networks. In *Network and Distributed System Security Symposium (NDSS'16)*, 2016.

[36] Chinese Routing Errors Redirect Russian Traffic. http://research.dyn.com/2014/11/chinese-routing-errors-redirect-russian-traffic/.

[37] Cloudflare: 94 Percent of the Tor Traffic We See is "Per Se Malicious". https://arstechnica.com/tech-policy/2016/03/new-data-suggests-94-percent-of-tor-traffic-is-malicious/.

[38] Content Delivery Networks Aren't Notorious Markets. https://medium.com/niskanen-center/content-delivery-networks-arent-notorious-markets-2a448a549cf7.

[39] Content Delivery Networks (CDNs). https://www.eff.org/free-speech-weak-link/cdn.

[40] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 321–338. IEEE, 2015.

[41] CREDO and Cloudflare Argue Against National Security Letter Gag Orders. https://techcrunch.com/2017/03/23/credo-and-cloudflare-argue-against-national-security-letter-gag-orders/.

[42] Data. https://bitbucket.org/ransom_research/data/.

[43] Delete Hate Speech or Pay Up, Germany Tells Social Media Companies. https://www.nytimes.com/2017/06/30/business/germany-facebook-google-twitter.html.

[44] Frank Denis and Yecheng Fu. Dnscrypt, 2015.

[45] Deutsche Telekom to Push for National Routing to Curtail Spying. http://www.businessweek.com/news/2013-10-14/deutsche-telekom-to-push-for-national.

[46] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. Technical report, DTIC Document, 2004.

[47] District court rules against Cloudflare in anti-piracy suit. https://thestack.com/security/2017/03/29/district-court-rules-against-cloudflare-in-anti-piracy-suit/.

[48] Dutch Senate votes in favor of dragnet surveillance powers. https://www.bof.nl/2017/07/12/dutch-senate-votes-in-favor-of-dragnet-surveillance-powers/.

[49] Anne Edmundson, Roya Ensafi, Nick Feamster, and Jennifer Rexford. A first look into transnational routing detours. In *ACM SIGCOMM Conference*, pages 567–568. ACM, 2016.

[50] Anne Edmundson, Roya Ensafi, Nick Feamster, and Jennifer Rexford. Avoiding Nation-State Surveillance. https://youtu.be/0_dQQ0BP0-s, 2016.

[51] Anne Edmundson, Roya Ensafi, Nick Feamster, and Jennifer Rexford. Characterizing and avoiding routing detours through surveillance states. *arXiv preprint arXiv:1605.07685*, 2016.

[52] Anne Edmundson, Roya Ensafi, Nick Feamster, and Jennifer Rexford. Studying Transnational Routing Detours through Surveillance State. `https://ripe73.ripe.net/archives/video/1431/`, 2016.

[53] Anne Edmundson, Roya Ensafi, Nick Feamster, and Jennifer Rexford. Nation-State Hegemony in Internet Routing. In *Conference on Computing and Sustainable Societies (COMPASS)*. ACM, 2018.

[54] Anne Edmundson, Paul Schmitt, and Nick Feamster. Oblivious DNS: Plugging the Internet's Biggest Privacy Hole. `https://freedom-to-tinker.com/2018/04/02/a-privacy-preserving-approach-to-dns/`, 2018.

[55] Anne Edmundson, Paul Schmitt, Nick Feamster, and Jennifer Rexford. Ocdn: Oblivious content distribution networks. *arXiv preprint arXiv:1711.01478*, 2017.

[56] Encryption App Signal Wins Fight Against FBI Subpoena and Gag Order. `https://www.dailydot.com/layer8/signal-subpoena-privacy-gag-order/`.

[57] Brian Eriksson, Paul Barford, Bruce Maggs, and Robert Nowak. Posit: a Lightweight Approach for IP Geolocation. *ACM SIGMETRICS Performance Evaluation Review*, 40(2):2–11, 2012.

[58] Brian Eriksson, Paul Barford, Joel Sommers, and Robert Nowak. A Learning-Based Approach for IP Geolocation. In *Passive and Active Measurement*, pages 171–180. Springer, 2010.

[59] EU Proposal for a Directive of The European Parliament and of the Council on Copyright in the Digital Single Market. 2016. `https://ec.europa.eu/transparency/regdoc/rep/1/2016/EN/1-2016-593-EN-F1-1.PDF`.

[60] Patrick T Eugster, Rachid Guerraoui, A-M Kermarrec, and Laurent Massoulié. Epidemic information dissemination in distributed systems. *Computer*, 37(5):60–67, 2004.

[61] Eyes Wide Open. `https://www.privacyinternational.org/sites/default/files/Eyes%20Wide%20Open%20v1.pdf`.

[62] Facebook Loses Appeal to Block Bulk Search Warrants. `https://www.nytimes.com/2017/04/04/nyregion/facebook-bulk-search-warrants-new-york-state.html?_r=0`.

[63] Rodérick Fanou, Pierre Francois, and Emile Aben. On the Diversity of Interdomain Routing in Africa. In *International Conference on Passive and Active Network Measurement*, pages 41–54. Springer, 2015.

[64] Rodérick Fanou, Gareth Tyson, Pierre Francois, and Arjuna Sathiaseelan. Pushing the Frontier: Exploring the African Web Ecosystem. In *International Conference on World Wide Web*, pages 435–445, 2016.

[65] Nick Feamster and Roger Dingledine. Location diversity in anonymity networks. In *ACM Workshop on Privacy in the Electronic Society*, pages 66–76. ACM, 2004.

[66] Hannes Federrath, Karl-Peter Fuchs, Dominik Herrmann, and Christopher Piosecny. Privacy-preserving DNS: analysis of broadcast, range queries and mix-based protection methods. In *European Symposium on Research in Computer Security*, pages 665–683. Springer, 2011.

[67] France Must Reject Law that Gives Carte Blanche to Mass Surveillance Globally. `https://www.amnesty.org/en/press-releases/2015/09/france-must-reject-law-that-gives-carte-blanche-to-mass-surveillance-globall`

[68] Lixin Gao. On inferring autonomous system relationships in the internet. *IEEE/ACM Transactions on Networking (ToN)*, 9(6):733–745, 2001.

[69] Roxana Geambasu, Jarret Falkner, Paul Gardner, Tadayoshi Kohno, Arvind Krishnamurthy, and Henry M Levy. Experiences building security applications on DHTs. *Technical Report, UW-CSE-09–09–01*, 2009.

[70] Nethanel Gelernter, Amir Herzberg, and Hemi Leibowitz. Two cents for strong anonymity: The anonymous post-office protocol. *Privacy Enhancing Technologies*, 2:1–20, 2016.

[71] Manaf Gharaibeh, Anant Shah, Bradley Huffaker, Han Zhang, Roya Ensafi, and Christos Papadopoulos. A look at router geolocation in public and commercial databases. In *Internet Measurement Conference*, pages 463–469. ACM, 2017.

[72] Yossi Gilad, Amir Herzberg, Michael Sudkovitch, and Michael Goberman. CDN-on-demand: an affordable DDoS defense via untrusted clouds. In *Network and Distributed Security Symposium (NDSS)*, 2016.

[73] Phillipa Gill, Yashar Ganjali, Bernard Wong, and David Lie. Dude, Where's That IP?: Circumventing Measurement-Based IP Geolocation. In *USENIX Conference on Security*, pages 16–16. USENIX Association, 2010.

[74] Gogo Inflight Internet Serves up 'Man in the Middle' with Fake SSL. `http://www.csoonline.com/article/2865806/cloud-security/gogo-inflight-internet-serves-up-man-in-the-middle-with-fake-ssl.html`.

[75] Benjamin Greschbach, Tobias Pulls, Laura M. Roberts, Philipp Winter, and Nick Feamster. The Effect of DNS on Tor's Anonymity. *CoRR*, abs/1609.08187, 2016.

[76] Benjamin Greschbach, Tobias Pulls, Laura M Roberts, Philipp Winter, and Nick Feamster. The Effect of DNS on Tor's Anonymity. 2017.

[77] Chuanxiong Guo, Yunxin Liu, Wenchao Shen, Helen J Wang, Qing Yu, and Yongguang Zhang. Mining the Web and the Internet for Accurate IP Address Geolocations. In *INFOCOM 2009, IEEE*, pages 2841–2845. IEEE, 2009.

[78] Arpit Gupta, Matt Calder, Nick Feamster, Marshini Chetty, Enrico Calandro, and Ethan Katz-Bassett. Peering at the Internet's Frontier: A First Look at ISP Interconnectivity in Africa. In *Passive and Active Measurement*, pages 204–213. Springer, 2014.

[79] Yihua He, Michalis Faloutsos, and Srikanth Krishnamurthy. Quantifying Routing Asymmetry in the Internet at the AS Level. In *Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE*, volume 3, pages 1474–1479. IEEE, 2004.

[80] Yihua He, Michalis Faloutsos, Srikanth Krishnamurthy, and Bradley Huffaker. On routing asymmetry in the internet. In *Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE*, volume 2, pages 6–pp. IEEE, 2005.

[81] J. Heidemann, A. Mankin, D. Wessels, P. Hoffman, Z. Hu, and L. Zhu. Specification for DNS Over Transport Layer Security (TLS). Internet Engineering Task Force (IETF), 2016.

[82] Dominik Herrmann, Karl-Peter Fuchs, Jens Lindemann, and Hannes Federrath. EncDNS: A lightweight privacy-preserving name resolution service. In *European Symposium on Research in Computer Security*, pages 37–55. Springer, 2014.

[83] Amir Herzberg and Haya Shulman. Retrofitting security into network protocols: The case of dnssec. *IEEE Internet Computing*, 18(1):66–71, 2014.

[84] Cristian Hesselman, Giovane CM Moura, Ricardo de Oliveira Schmidt, and Cees Toet. Increasing DNS security and stability through a control plane for top-level domain operators. *IEEE Communications Magazine*, 55(1):197–203, 2017.

[85] Paul Hill. Comcast begins man-in-the-middle attacks to show copyright notices on websites. `https://www.neowin.net/news/comcast-begin-man-in-the-middle-attacks-to-show-copyright-notices-on-website`, 2015.

[86] P. Hoffman and P. McManus. DNS Queries over HTTPS. `https://tools.ietf.org/html/draft-ietf-doh-dns-over-https-04`, 2018.

[87] How the NSA & FBI made Facebook the perfect mass surveillance tool. https://venturebeat.com/2014/05/15/how-the-nsa-fbi-made-facebook-the-perfect-mass-surveillance-tool/.

[88] How Brazil Crowdsourced a Landmark Law. http://foreignpolicy.com/2016/01/19/how-brazil-crowdsourced-a-landmark-law/, 2016.

[89] Zi Hu, John Heidemann, and Yuri Pradkin. Towards Geolocation of Millions of IP Addresses. In *ACM Conference on Internet Measurement Conference*, pages 123–130. ACM, 2012.

[90] Cheng Huang, Angela Wang, Jin Li, and Keith W Ross. Measuring and evaluating large-scale CDNs. In *ACM IMC*, volume 8, 2008.

[91] Bradley Huffaker, Marina Fomenkov, and K Claffy. Geocompare: A Comparison of Public and Commercial Geolocation Databases. *Proc. NMMC*, pages 1–12, 2011.

[92] Investigatory Powers Bill: Snooper's Charter Lacks Clarity, MPs Warn. http://www.theguardian.com/law/2016/feb/01/investigatory-powers-bill-snoopers-charter-lacks-clarity-mps-warn.

[93] Internet-Wide Scan Data Repository. https://scans.io/study/washington-dns.

[94] Investigatory Powers Act: Britain's online surveillance laws explained. http://www.pocket-lint.com/news/142641-investigatory-powers-act-britain-s-online-surveillance-laws-explained.

[95] MH Jalalzai, WB Shahid, and MMW Iqbal. DNS security challenges and best practices to deploy secure DNS with digital signatures. In *Applied Sciences and Technology (IBCAST), 2015 12th International Bhurban Conference on*, pages 280–285. IEEE, 2015.

[96] Yaoqi Jia, Guangdong Bai, Prateek Saxena, and Zhenkai Liang. Anonymity in peer-assisted CDNs: Inference attacks and mitigation. *Privacy Enhancing Technologies*, 2016(4):294–314, 2016.

[97] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries. In *CCS*. ACM, 2013. http://www.ohmygodel.com/publications/usersrouted-ccs13.pdf.

[98] Jaeyeon Jung, Balachander Krishnamurthy, and Michael Rabinovich. Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites. In *International Conference on World Wide Web*, pages 293–304. ACM, 2002.

[99] David Karger, Eric Lehman, Tom Leighton, Rina Panigrahy, Matthew Levine, and Daniel Lewin. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *ACM Symposium on Theory of Computing*, pages 654–663. ACM, 1997.

[100] Josh Karlin, Stephanie Forrest, and Jennifer Rexford. Nation-state Routing: Censorship, Wiretapping, and BGP. *arXiv preprint arXiv:0903.3218*, 2009.

[101] Ethan Katz-Bassett, John P John, Arvind Krishnamurthy, David Wetherall, Thomas Anderson, and Yatin Chawathe. Towards IP Geolocation Using Delay and Topology Measurements. In *ACM SIGCOMM conference on Internet Measurement*, pages 71–84. ACM, 2006.

[102] Ethan Katz-Bassett, Harsha V Madhyastha, Vijay Kumar Adhikari, Colin Scott, Justine Sherry, Peter Van Wesep, Thomas E Anderson, and Arvind Krishnamurthy. Reverse traceroute. In *NSDI*, volume 10, pages 219–234, 2010.

[103] Kazakhstan Will Require Internet Surveillance Back Doors. `http://www.engadget.com/2015/12/05/kazakhstan-internet-back-door-law/`, 2015.

[104] Anne-Marie Kermarrec and Maarten Van Steen. Gossiping in distributed systems. *ACM SIGOPS Operating Systems Review*, 41(5):2–7, 2007.

[105] Sheharbano Khattak, David Fifield, Sadia Afroz, Mobin Javed, Srikanth Sundaresan, Vern Paxson, Steven J Murdoch, and Damon McCoy. Do You See What I See? Differential Treatment of Anonymous Users. In *Network and Distributed System Security Symposium*, 2016.

[106] Jeffrey Knockel, Masashi Crete-Nishihata, Jason Q Ng, Adam Senft, and Jedidiah R Crandall. Every rose has its thorn: Censorship and surveillance on social video platforms in china. In *5th USENIX Workshop on Free and Open Communications on the Internet (FOCI 15)*, 2015.

[107] Srinivas Krishnan and Fabian Monrose. DNS prefetching and its privacy implications: when good things go bad. In *USENIX Conference on Large-Scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More*, pages 10–10, 2010.

[108] Albert Kwon, Henry Corrigan-Gibbs, Srinivas Devadas, and Bryan Ford. Atom: Scalable anonymity resistant to traffic analysis. *arXiv preprint arXiv:1612.07841*, 2016.

[109] Albert Kwon, David Lazar, Srinivas Devadas, and Bryan Ford. Riffle. *Privacy Enhancing Technologies*, 2016(2):115–134, 2016.

[110] Sir Stephen Lander. International Intelligence Cooperation: An Inside Perspective 1. *Cambridge Review of International Affairs*, 17(3):481–493, 2004.

[111] Matt Larson, Dan Massey, Scott Rose, Roy Arends, and Rob Austein. DNS security introduction and requirements. 2005.

[112] Law Enforcement Requests Report. `https://www.microsoft.com/en-us/about/corporate-responsibility/lerr`.

[113] Chris Lesniewski-Laas and M Frans Kaashoek. SSL splitting: Securely serving data from untrusted caches. *Computer Networks*, 48(5):763–779, 2005.

[114] Dave Levin, Youndo Lee, Luke Valenta, Zhihao Li, Victoria Lai, Cristian Lumezanu, Neil Spring, and Bobby Bhattacharjee. Alibi Routing. In *The 2015 ACM Conference on Special Interest Group on Data Communication*, pages 611–624. ACM, 2015.

[115] Amit Levy, Henry Corrigan-Gibbs, and Dan Boneh. Stickler: Defending against malicious CDNs in an unmodified browser. *arXiv preprint arXiv:1506.04110*, 2015.

[116] Daniel Mark Lewin. *Consistent hashing and random trees: Algorithms for caching in distributed networks*. PhD thesis, Massachusetts Institute of Technology, 1998.

[117] Jinjin Liang, Jian Jiang, Haixin Duan, Kang Li, Tao Wan, and Jianping Wu. When HTTPS meets CDN: A case of authentication in delegated service. In *Security and Privacy (S&P), 2014 IEEE Symposium on*, pages 67–82. IEEE, 2014.

[118] Harsha V Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iPlane: An Information Plane for Distributed Services. In *The 7th Symposium on Operating Systems Design and Implementation*, pages 367–380. USENIX Association, 2006.

[119] S. Marchal, J. François, C. Wagner, R. State, A. Dulaunoy, T. Engel, and O. Festor. Dnssm: A large scale passive dns security monitoring framework. In *2012 IEEE Network Operations and Management Symposium*, pages 988–993, April 2012.

[120] MaxMind. `https://www.maxmind.com/en/home`.

[121] David Folkman Mazières. *Self-certifying file system*. PhD thesis, Massachusetts Institute of Technology, 2000.

[122] Nikolaos Michalakis, Robert Soulé, and Robert Grimm. Ensuring content integrity for untrusted peer-to-peer content distribution networks. In *USENIX Conference on Networked Systems Design & Implementation*, pages 11–11. USENIX Association, 2007.

[123] Netherlands New Proposal for Dragnet Surveillance Underway. https://edri.org/netherlands-new-proposals-for-dragnet-surveillance-underway/, 2015.

[124] Daiyuu Nobori and Yasushi Shinjo. VPN gate: A Volunteer-organized Public VPN Relay System with Blocking Resistance for Bypassing Government Censorship Firewalls. In *The 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 229–241, 2014.

[125] NSA infiltrates links to Yahoo, Google data centers worldwide, Snowden documents say. https://www.washingtonpost.com/world/national-security/nsa-infiltrates-links-to-yahoo-google-data-centers-worldwide-snowden-document 2013/10/30/e51d661e-4166-11e3-8b74-d89d714ca4dd_story.html?utm_term=.ad3fb06d0888.

[126] Jonathan A Obar and Andrew Clement. Internet Surveillance and Boomerang Routing: A Call for Canadian Network Sovereignty. In *TEM 2013: The Technology & Emerging Media Track-Annual Conference of the Canadian Communication Association (Victoria)*, 2012.

[127] Eric Osterweil, Dan Massey, and Lixia Zhang. Deploying and monitoring DNS security (DNSSEC). In *Computer Security Applications Conference, 2009. ACSAC'09. Annual*, pages 429–438. IEEE, 2009.

[128] Andriy Panchenko, Fabian Lanze, Andreas Zinnen, Martin Henze, Jan Pennekamp, Klaus Wehrle, and Thomas Engel. Website fingerprinting at internet scale. In *Network & Distributed System Security Symposium (NDSS). IEEE Computer Society*, 2016.

[129] Vern Paxson. End-to-end routing behavior in the internet. *IEEE/ACM transactions on Networking*, 5(5):601–615, 1997.

[130] Simon Peter, Umar Javed, Qiao Zhang, Doug Woos, Thomas Anderson, and Arvind Krishnamurthy. One tunnel is (often) enough. *ACM SIGCOMM Computer Communication Review*, 44(4):99–110, 2015.

[131] Ania Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The loopix anonymity system. *arXiv preprint arXiv:1703.00536*, 2017.

[132] PlanetLab. http://planet-lab.org/.

[133] Ingmar Poese, Steve Uhlig, Mohamed Ali Kaafar, Benoit Donnet, and Bamba Gueye. IP Geolocation Databases: Unreliable? *ACM SIGCOMM Computer Communication Review*, 41(2):53–56, 2011.

[134] Promoting the Use of Internet Exchange Points (IXPs): A Guide to Policy, Management and Technical Issues. https://www.internetsociety.org/sites/default/files/Promoting%20the%20use%20of%20IXPs.pdf, 2012.

[135] Charles Rackoff and Daniel R Simon. Cryptographic defense against traffic analysis. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 672–681. ACM, 1993.

[136] RAN. https://bitbucket.org/ransom_research/ran/.

[137] T. Reddy, D. Wing, and P. Patil. Specification for DNS over Datagram Transport Layer Security (DTLS). https://tools.ietf.org/html/draft-ietf-dprive-dnsodtls-15, 2016.

[138] Report of the Special Rapporteur on the right to privacy. http://www.ohchr.org/Documents/Issues/Privacy/A-HRC-31-64.doc.

[139] Requests for user information. https://transparencyreport.google.com/user-data/overview?t=table.

[140] Revealed: How US and UK Spy Agencies Defeat Internet Privacy and Security. https://www.theguardian.com/world/2013/sep/05/nsa-gchq-encryption-codes-security.

[141] RIPE Atlas. https://atlas.ripe.net/.

[142] Hal Roberts, David Larochelle, Rob Faris, and John Palfrey. Mapping Local Internet Control. In *Computer Communications Workshop (Hyannis, CA, 2011), IEEE*, 2011.

[143] Ira S Rubinstein, Gregory T Nojeim, and Ronald D Lee. Systematic government access to personal data: a comparative analysis. *International Data Privacy Law*, 4(2):96–119, 2014.

[144] David Ruiz. Responsibility Deflected, the CLOUD Act Passes. https://www.eff.org/deeplinks/2018/03/responsibility-deflected-cloud-act-passes, 2018.

[145] Christine Runnegar. Encryption and Law Enforcement Can Work Together. https://www.internetsociety.org/blog/2017/10/encryption-law-enforcement-can-work-together/, 2017.

[146] Russia Needs More Internet Security Says Putin. http://www.wsj.com/articles/russia-needs-more-internet-security-says-putin-1412179448, 2014.

[147] Lidija Sabados. Indian government wiretapping. https://citizenlab.ca/2013/01/indian-government-wiretapping/.

[148] Schrems v. Data Protection Commissioner. https://epic.org/privacy/intl/schrems/.

[149] Will Scott, Thomas Anderson, Tadayoshi Kohno, and Arvind Krishnamurthy. Satellite: Joint analysis of CDNs and network-level interference. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, pages 195–208. USENIX Association, 2016.

[150] Adam Senft, Jakub Dalek, Irene Poetranto, Masashi Crete-Nishihata, , and Aim Sinpeng. Information Controls during Thailand's 2014 Coup. `https://citizenlab.ca/2014/07/information-controls-thailand-2014-coup/`.

[151] Anant Shah and Christos Papadopoulos. Characterizing International BGP Detours. Technical Report CS-15-104, Colorado State University, 2015.

[152] Vitaly Shmatikov and Ming-Hsiu Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *European Symposium on Research in Computer Security*, pages 18–33. Springer, 2006.

[153] Haya Shulman. Pretty bad privacy: Pitfalls of DNS encryption. In *Workshop on Privacy in the Electronic Society*, pages 191–200. ACM, 2014.

[154] Simple Hit-Metering and Usage-Limiting for HTTP. `https://www.ietf.org/rfc/rfc2227.txt`.

[155] Amitpal Singh. Christopher Parsons On Pakistan's Blackberry Information Requests. `https://citizenlab.ca/2015/12/christopher-parsons-on-pakistans-blackberry-information-requests/`.

[156] Andrei Soldatov and Irina Borogan. Russia's surveillance state. *World Policy Journal*, 30(3):23–30, 2013.

[157] Ao-Jan Su, David R Choffnes, Aleksandar Kuzmanovic, and Fabián E Bustamante. Drafting behind Akamai: Inferring network conditions based on CDN redirections. *IEEE/ACM Transactions on Networking (TON)*, 17(6):1752–1765, 2009.

[158] Ao-Jan Su and Aleksandar Kuzmanovic. Thinning Akamai. In *ACM SIGCOMM Conference on Internet Measurement*, pages 29–42. ACM, 2008.

[159] Yixin Sun, Anne Edmundson, Laurent Vanbever, Oscar Li, Jennifer Rexford, Mung Chiang, and Prateek Mittal. Raptor: Routing attacks on privacy in tor. In *USENIX Security*, pages 271–286, 2015.

[160] TeleGeography Submarine Cable Map. `http://www.submarinecablemap.com/`.

[161] The East African Marine System. `http://www.teams.co.ke/`.

[162] Matthew Thomas and Aziz Mohaisen. Measuring the leakage of onion at the root: A measurement of tor's. onion pseudo-tld in the global domain name system. In *Workshop on Privacy in the Electronic Society*, pages 173–180. ACM, 2014.

[163] Sipat Triukose, Zakaria Al-Qudah, and Michael Rabinovich. Content delivery networks: Protection or threat? In *European Symposium on Research in Computer Security*, pages 371–389. Springer, 2009.

[164] The Trouble with Tor. `https://blog.cloudflare.com/the-trouble-with-tor/`.

[165] Lokman Tsui. The panopticon as the antithesis of a space of freedom control and regulation of the internet in china. *China information*, 17(2):65–82, 2003.

[166] Nirvan Tyagi, Yossi Gilad, Matei Zaharia, and Nickolai Zeldovich. Stadium: A distributed metadata-private messaging system. *IACR Cryptology ePrint Archive*, 2016:943, 2016.

[167] UK Targets WhatsApp Encryption After London Attack. `http://www.rappler.com/technology/news/165288-uk-targets-whatsapp-encryption-london-parliament-attack`.

[168] Jelle Van Den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Symposium on Operating Systems Principles*, pages 137–152. ACM, 2015.

[169] Werner Vogels, Robbert Van Renesse, and Ken Birman. The power of epidemics: robust communication for large-scale distributed systems. *ACM SIGCOMM Computer Communication Review*, 33(1):131–135, 2003.

[170] Matthias Wählisch, Sebastian Meiling, and Thomas C Schmidt. A Framework for Nation-centric Classification and Observation of the Internet. In *The ACM CoNEXT Student Workshop*, page 15. ACM, 2010.

[171] Matthias Wählisch, Thomas C Schmidt, Markus de Brün, and Thomas Häberlen. Exposing a Nation-centric View on the German Internet–A Change in Perspective on AS-level. In *Passive and Active Measurement*, pages 200–210. Springer, 2012.

[172] Shaojung Sharon Wang and Junhao Hong. Discourse behind the Forbidden Realm: Internet surveillance and its implications on China's blogosphere. *Telematics and Informatics*, 27(1):67–78, 2010.

[173] Patrick Wendell and Michael J Freedman. Going viral: Flash crowds in an open CDN. In *ACM SIGCOMM Conference on Internet Measurement Conference*, pages 549–558. ACM, 2011.

[174] What ISPs Can See: Clarifying the Technical Landscape of the Broadband Privacy Debate. `https://www.teamupturn.com/reports/2016/what-isps-can-see`.

[175] What is 1.1.1.1? `https://www.cloudflare.com/learning/dns/what-is-1.1.1.1/`, 2018.

[176] Whatsapp Encryption Keeps Us Safe: Attacking it is Wrong. http://www.telegraph.co.uk/technology/2017/03/27/whatsapps-encryption-keeps-us-safe-attacking-wrong/.

[177] Scott Wolchok, Owen S Hofmann, Nadia Heninger, Edward W Felten, J Alex Halderman, Christopher J Rossbach, Brent Waters, and Emmett Witchel. Defeating Vanish with Low-Cost Sybil Attacks Against Large DHTs. In *NDSS*, 2010.

[178] David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. Dissent in numbers: Making strong anonymity scale. In *OSDI*, pages 179–182, 2012.

[179] World's Biggest Internet Hub Sues German Government Over Surveillance. http://fortune.com/2016/09/16/de-cix-surveillance-germany/.

[180] Charles V Wright, Scott E Coull, and Fabian Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *NDSS*, volume 9, 2009.

[181] Jingan Xue, David Choffnes, and Jilong Wang. CDNs Meet CN: An Empirical Study of CDN Deployments in China. *IEEE Access*, 2017.

[182] The Zettabyte Era – Trends and Analysis – Cisco. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html.

[183] Xin Zhang, Hsu-Chun Hsiao, Geoffrey Hasker, Haowen Chan, Adrian Perrig, and David G Andersen. Scion: Scalability, control, and isolation on next-generation networks. In *Security and Privacy (S&P), 2011 IEEE Symposium on*, pages 212–227. IEEE, 2011.

[184] Dave Levin Zhihao Li, Stephen Herwig. Detor: Provably avoiding geographic regions in tor. In *USENIX Security 2017*, 2017.

[185] Shi Zhou, Guo-Qing Zhang, and G-Q Zhang. Chinese Internet AS-level topology. *Communications, IET*, 1(2):209–214, 2007.

[186] Liang Zhu, Zi Hu, John Heidemann, Duane Wessels, Allison Mankin, and Nikita Somaiya. Connection-oriented dns to improve privacy and security. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 171–186. IEEE, 2015.

[187] Hadi Zolfaghari and Amir Houmansadr. Practical censorship evasion leveraging content delivery networks. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1715–1726. ACM, 2016.