Network Traffic Obfuscation and Automated Internet Censorship

Lucas Dixon | Jigsaw
Thomas Ristenpart | Cornell Tech
Thomas Shrimpton | University of Florida

Increasingly, censors deploy advanced networking tools such as deep-packet inspection to identify objectionable connections. This survey reviews network traffic obfuscation and its role in circumventing Internet censorship.

Around the world, well-resourced governments seek to censor the Internet. Censorship is driven by governments' desires to control access to information deemed politically or socially sensitive, or to protect national economic interests. For example, in China, performing Internet searches for information on Tiananmen Square reveals no information about the events in 1989, and communication platforms run by companies outside China, such as Gmail and Twitter, are among those that are routinely blocked. Although China's "Great Firewall" is perhaps the best known example of Internet censorship by a nation-state, similar controls are enacted in Turkey, Iran, Malaysia, Syria, Belarus, and many other countries. (An excellent survey of the censorship policies by various governments around the world is provided by the OpenNet Initiative; opennet.net.)

In this article, we focus on the technical underpinnings of automated censorship and of the network traffic obfuscation tools aimed at circumventing it. By automated censorship, we mean government-deployed or -mandated network monitoring equipment typically installed at ISPs. These systems detect and disrupt the flow of censored information without any direct human involvement and are applied broadly to enforce government policies over all citizens. This differs from targeted approaches to taking down Internet content, such as the US government's takedown of the Silk Road, which was an Internet marketplace for illicit goods such as drugs and weapons.

Background: From Censorship to Protocol Obfuscation

In this article, we use terms like "censor," "adversary," and "sensitive flow" repeatedly. To aid understanding, we provide a summary of such key terms in Table 1.

The technical means of automated censorship assumes that the censor enjoys privileged access to network infrastructure. Governments typically have this via direct or indirect control of ISPs. In the security community, this is called an *on-path adversary*: it's on the path of network packets between the user seeking to access sensitive content and the provider of that content (for example, a webserver).

Traffic Identification Methods

Network censors face two main tasks: to accurately identify network traffic that carries or attempts to

Table 1. Summary of often-used technical terms.

Term	Definition
Sensitive flow	Network communications targeted for being offensive or counterregime or for using obfuscation tools
Censor, adversary	The party that controls the communication network and is attempting to identify and restrict sensitive flows of information
Obfuscation tool	Software designed to prevent censors from identifying or blocking communications
Background traffic	Nonsensitive flows coexisting (potentially) with sensitive flows on the network
Collateral damage	Background traffic adversely affected by a censor's efforts to control sensitive flows
Proxy	Party that will forward traffic to the broader Internet on behalf of a censored client; external to censor-controlled network

access sensitive content, and (potentially) to perform a censoring action on that traffic. Examples of the latter are dropping packets to degrade the user's experience, forcing one or both of the endpoints to drop the connection (for instance, by sending TCP reset messages), and redirecting web connections to a censor-controlled webserver. But as censoring actions must be preceded by identification of targeted traffic, we focus only on the censor's identification task.

Via address. Originally, attempts to access censored Internet content were identified by first associating addressing information—IP addresses, port numbers, and domain names—with sensitive content, resulting in a blacklist of addresses. Any network connections to these addresses were deemed as attempts to access sensitive content. For blacklisted domain names, the censor can direct ISPs to run modified DNS software. Connections to these domains are then typically blocked or misdirected. For IP addresses, the censor can install hardware mechanisms at ISPs that compare information in a packet's IP header against the list. As the IP headers appear early in the packet, one needs only to perform "shallow" packet inspection, disregarding information encapsulated deeper in the packet. TCP or User Datagram Protocol (UDP) port information is similarly available via shallow inspection.

A user can avoid identification via domain name and IP/port by using a proxy, a cooperative machine whose address information isn't blacklisted. The operation of proxy-based circumvention is summarized in Figure 1. Many tools such as Tor, uProxy, Anonymizer, and Psiphon rely on the use of proxies. Recently, domain fronting,[1] a form of decoy routing[2–4] that operates at the application layer, has also been successfully deployed.

Via deep-packet inspection. The success of proxy systems in practice has led censors to deploy new deep-packet inspection (DPI) mechanisms that identify traffic based on information deeper inside the network packets. For example, application-layer content carried in (unencrypted) packet payloads can divulge user-generated data, such as keywords within search URLs. For example, China's Great Firewall blocks traffic containing blacklisted keywords.

In reaction to DPI, modern circumvention tools encrypt content to proxies, preventing this kind of keyword search. The result has been that sophisticated modern censors now use DPI to attempt to identify and block traffic created by circumvention tools. For example, China used DPI to detect Tor connections by looking for a specific sequence of bytes in the first application-layer message from client to server. This sequence corresponds to the TLS ciphersuites that Tor clients selected, which were unlike the other TLS implementations selected. As another example, Iran used DPI to determine the expiration date of TLS certificates. At the time, Tor servers used certificates with relatively near expiration dates, while most commercial TLS servers chose expiration dates that were many years away.

These blocking policies used fairly sophisticated information about the circumvention tool—the particular implementation of TLS—being carried in packet payloads. But even a crude scan of application-layer content can discover that standard protocols (for example, SSH and TLS) are being transported. This might be enough to trigger insensitive blocking or logging of source/destination IP addresses for more detailed analysis or later action.

Network Traffic Obfuscation

The use of DPI to perform tool identification motivated the circumvention tool community to seek countermeasures, culminating in an entire area of research and development: network traffic obfuscation. The goal is to force DPI-using censors to abandon attempts at protocol identification, because accurate identification at scale requires prohibitive resources, for example, by developing tools whose traffic can reliably force DPI to misclassify the traffic as that of a nonblacklisted tool.

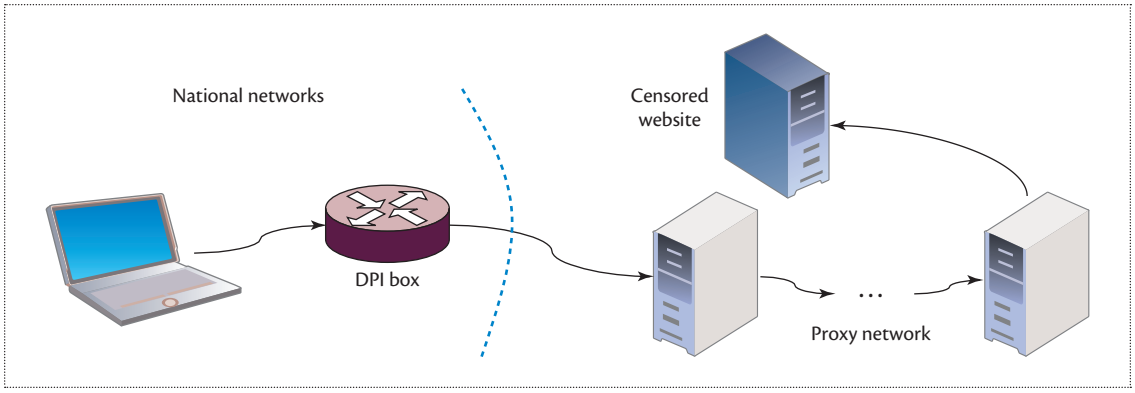

Figure 1. Proxy-based censorship circumvention. A client connects to a proxy server that relays its traffic to the intended destination. Should anonymity be desired, the server might send the traffic through multiple proxies before sending it to the destination. DPI is deep-packet inspection.

We focus on network traffic obfuscation in the remainder of this article.

We want to emphasize that our focus on obfuscation doesn't belie the importance of other aspects of censorship circumvention. IP- and DNS-based filtering are still more prevalent than DPI-based mechanisms, and circumventing such filtering remains a challenge in practice. Ultimately, circumvention requires a variety of mechanisms that work in concert to defeat a censor, and a system is only as good as its weakest link.

Threat models. A threat model is a particular combination of censor capabilities and goals. We provide only informal description of threat models for the purposes of this article. We emphasize that the threat models presented here apply only to gathering information from network flows (IP information, application-layer content, and so on) and not to the censoring actions (blocking or otherwise disrupting the communication) that this information enables. Information might be gathered by using passive or active techniques.

A passive technique simply observes traffic flows. One high-profile example of passive techniques occurred during the February 2012 anniversary of the 1979 Iranian revolution. The Islamic Republic of Iran mounted a large-scale censorship campaign, blocking various IP addresses associated with anticensorship tools, such as torproject.org, and using DPI-based identification of TLS network flows.[5]

An active technique modifies existing packet contents or injects completely new packets. Examples of active attacks that manipulate connections by modifying or dropping specific packets have appeared in the academic literature.[6] Although far less common than passive techniques, active techniques have been used by the Chinese government to identify Tor proxies.[7] Table 2 lists examples of passive and active attacks. For brevity, in the remainder of the article we focus primarily on passive attackers.

So far, we've spoken mostly about the capabilities part of the threat model. In terms of the goals, the typical assumption is that censors minimally want to accurately classify traffic as either targeted (for application of policy) or not. Accuracy has two facets. The first is a high true-positive rate, meaning that the fraction of targeted traffic marked as such is close to one. The second is a low false-positive rate, meaning that the fraction of nontargeted traffic incorrectly marked as targeted is close to zero. Obfuscation tools are often said to be unobservable if they can cause sufficiently low true-positive and high false-positive rates. What constitutes sufficient error rates, however, is largely unknown. Tools with the potential to induce high false-positive rates are preferred because censors will be less likely to block them.

Sensitivity to collateral damage. Collateral damage occurs when a censor's detection mechanisms suffer from false positives, leading them to mislabel network flows as associated to a blacklisted tool, protocol, or content when, in fact, the flow is not. Why should censors be sensitive to collateral damage? The answer is highly context dependent, varies over time, and is ultimately a question of sociopolitical factors. Iran's February 2012 censorship campaign is an example of insensitive censorship because the vast majority of TLS traffic was almost certainly not used for circumvention or bound for hosts deemed offensive by the regime. We note that this particularly broad censorship policy lasted only briefly, and that it's typical for censorship policies to change, in terms of targets and sensitivity, over time.

Obfuscation Approaches

Censors' DPI use has inspired researchers and activists to explore various countermeasures aimed

Table 2. Examples of censorship by nation-states.

Country	Date	Technique and target*	Passive or active
China	2007–	IP/DNS blacklisting, **Tor TLS header fingerprinting,** and **Tor-relay handshake probe**	Both
Syria	2011–	IP/DNS blacklisting, **instant message fingerprinting,** and **social media keyword blacklisting**	Passive
Iran	Feb. 2012	IP blacklisting, **keyword flagging,** and **TLS handshake fingerprinting**	Passive
Ethiopia	2012	**Skype detection** and **Tor TLS fingerprinting**	Passive
Kazakhstan	2012	**Encryption fingerprinting** and **Tor TLS handshake fingerprinting**	Passive
Turkey	Early 2014	IP/DNS blacklisting and Border Gateway Protocol hijack of Twitter and YouTube	Both
Belarus	2014–	IP/DNS blacklisting and **filtering according to packet content**	Passive
Egypt	2014–	IP/DNS blacklisting, **instant message fingerprinting,** and **social media keyword blacklisting**	Passive
Pakistan	Intermittent	IP/DNS blacklisting, **URL, keyword,** and **filetype flagging**	Passive

** Techniques and targets in bold rely on data-packet inspection.*

at obfuscating packet payloads. We break down approaches into four categories: encryption, randomization, mimicry, and tunneling.

Encryption

A natural first thought is that conventional encryption is already a good obfuscator. Indeed, encryption renders unintelligible the data that's encrypted. But most encryption protocols have plaintext headers in each packet payload that identify the protocol and sometimes indicate the particular application that's implementing the protocol. We've seen in Iran and China that plaintext headers contain fingerprintable parameters that can help a censor distinguish between, say, Tor's TLS implementations from non-Tor implementations. In short, just encrypting data doesn't guarantee a level of unobservability sufficient for the goal of making DPI unreliable for censors.

Randomization

If conventional encryption, like plaintext protocols, can leave exploitable fingerprints for DPI, why not randomize every byte in the packet payload? This strategy, which has been described as attempting to make network flows "look like nothing" by the obfs family of obfuscators,[8] is used by Dust and ScrambleSuit.[9,10] ScrambleSuit and the obfs methods are designed specifically for the Tor pluggable transport framework Obfsproxy.

The standard approach to randomizing a payload is to apply a stream cipher to every byte. Assuming a passive censor that doesn't have access to the stream cipher secret key, this approach makes protocol identification difficult because there are no fixed fingerprints to observe. Because stream ciphers don't provide the strong confidentiality and authenticity guarantees sought, they're typically applied on top of modern encryption methods that do.

Underlying the use of randomizing "look-like-nothing" obfuscators is the assumption that censors use only blacklisting: they allow all traffic except that which conforms to an explicit set of disallowed traffic types. If censors use whitelisting, exclusively or in addition to blacklisting, then fully randomized payloads won't avoid censorship by virtue of looking like no actual network traffic.

Mimicry

To address whitelisting censors, a class of obfuscation methods attempt to evade censorship by masquerading as a whitelisted protocol. Loosely, rather than trying to look like nothing, mimicry-based obfuscators try to make packet payloads look like something that they're not—specifically, something that will be identified by DPI as allowable. A common example is to make payloads look like HTTP, which is rarely blocked because of its ubiquity. Mimicry in the context of anticensorship originates with the Tor project prepending HTTP headers to Tor packets,[8] though mimicry can be thought of as steganography against a particular type of adversary (DPI systems). StegoTorus and SkypeMorph, for example, attempted to use steganographic approaches to look like HTTP and Skype traffic, respectively.[11,12] Unfortunately, they have prohibitively poor performance.

A minimalist but flexible approach to mimicry can be found in format-transforming encryption (FTE).[13] As implemented in Tor, uProxy, and elsewhere, FTE

lets programmers use regular expressions to specify the format of ciphertexts. Because DPI also uses regular expressions, one can often precisely force misclassification of the protocol.

Mimicry obfuscators attempt to make packet payloads look like a particular "cover" protocol but don't actually use an implementation of the targeted cover. This means that the syntax and semantics of messages emanating from mimicry systems might deviate, often significantly, from that of messages conforming to the cover protocol. As one example, a mimicking obfuscator might produce a message purporting to be an `HTTP GET` after having just received a message purporting to be an `HTTP GET`, thereby violating proper HTTP semantics.

Tunneling

Instead of custom obfuscation implementations of mimicry that reproduce some aspects of the cover protocol, tunneling obfuscators transmit messages produced by an actual implementation of the cover protocol. Tunneling is of course not something new to obfuscation: all proxy systems use tunneling over encrypted channels. So any VPN or HTTPS proxy might be considered a tunneling obfuscator. The key difference is that tunneling obfuscators are meant to obfuscate the fact that an anticensorship tool is being used, and sometimes even obfuscate that encryption is being used. A VPN or proxy might not attempt to hide such facts.

Two challenges arise in tunneling obfuscators. The first is functional and occurs when there's a mismatch between the desired network transport semantics of the tunneled data (for instance, Tor) and the offered transport semantics of the tunneling protocol. As an example, HTTP is tuned in many ways for high bandwidth downstream and low bandwidth upstream, while someone using Tor might be uploading a large file. Or one might want to tunnel Tor over a tunneling protocol built on top of an unreliable network transport such as UDP. Because the latter disregards packet drops, one will have to work to implement logic to achieve the reliability that Tor implementations expect. Functionality must be achieved while also ensuring suitably high performance.

The second challenge is that tunneling actually might fall short of hiding the true nature of the tunneled traffic. Consider again that Iran and China distinguished Tor TLS from non-Tor TLS first-hop connections. Tor was using an actual implementation of the TLS protocol to tunnel its traffic—that is, Tor was using tunneling obfuscation. Ultimately, obfuscation designers face the challenge of ensuring that, when tunneling, they take care to match the behavior of common implementations and usage of the cover protocol.

A notable example of a tunneling obfuscator is the Tor pluggable transport Meek.[1] Meek tunnels via TLS to HTTP load balancers operated by Google, Amazon CloudFront, or Microsoft Azure. This is called *domain fronting* because the ostensible domain revealed at the TCP layer (and so, in the clear) is google.com, say, but the underlying encrypted request is for a domain that routes to a Meek-enabled Tor relay. (Interestingly, a similar use of domain fronting was employed by the GoAgent system and was popular in China in 2013.) As with Tor TLS, the Meek obfuscator is at pains to ensure that tunneled Tor connections to the load balancer look like conventional TLS connections to the load balancer. It's worth noting that this approach relies on an undocumented feature of load balancer implementations, and it's conceivable that companies might modify future load balancers in a way that prevents domain fronting.

Table 3 summarizes a range of obfuscators that have been deployed and compares them qualitatively in terms of deployment, key exchange (should encryption be used to perform obfuscation), and performance. It also reviews attacks that have been used against them.

Seeing through Obfuscation

Can real censors detect obfuscators? Researchers have begun preliminary investigations into this question; we summarize their approaches here, focusing on passive techniques. The takeaway message from what follows is that the answer isn't yet clear.

Syntactic and Semantic Mismatch

The first intensive investigation into obfuscator detection was the work of Amir Houmansadr and his colleagues.[6] They focused on mimicry systems in particular and claimed that all such systems are fundamentally flawed because they'll never perfectly replicate all aspects of a target protocol implementation.

They looked specifically at the extent to which messages emitted by currently implemented systems abide by the proper syntax or semantics of the cover protocol. By syntax, we mean that the emitted messages are parsable as cover protocol messages—required headers are in the right places, the correct character sets are used, and so forth. Semantics refers instead to whether messages are meaningful relative to the protocol's proper functioning.

Most mimicry systems explicitly made no attempt to mimic all aspects of the cover protocol, making the search for syntactic and semantic mismatches a foregone conclusion. As examples, StegoTorus doesn't implement most of HTTP, and the packets sent when using FTE (in a way that favors performance) contain large numbers of random, unformatted bits. The authors

Table 3. Comparison of network obfuscation approaches.*

Obfuscator	Type	Deployment	Key exchange	Performance	Attack	Notes
Cirripede	Tunneling	—	In band (TLS) with preregistration	High	—	Requires on-path cooperating ISP/ routers
CloudTransport[18]	Tunneling	—	In band (TLS)	Moderate	—	Google or Amazon HTTPS; cloud storage as rendezvous
Collage[19]	Mimicry	—	N/A	Low	—	Embed in user-generated content; use content servers as rendezvous
Decoy Routing	Tunneling	—	Out of band	High	—	Requires on-path cooperating routers
Domain Fronting (for example, Meek)	Tunneling	Tor, Psiphon, Lantern, uProxy, GoAgent	In band (TLS)	Moderate	See "Seeing through Network-Protocol Obfuscation"[14]	Google or Amazon HTTPS
Dust	Randomization	uProxy	Out of band	High	—	Stream cipher with transmitted key material
FTE	Mimicry	Lantern, Tor, uProxy	Out of band	High	See "Seeing through Network-Protocol Obfuscation"[14]	Flexible DPI-regex matching of HTTP, SSH, SMTP, and so on
Flashproxy[20]	Tunneling	Tor	N/A	Moderate	—	Plaintext tunnel using websockets through browser-based proxies
Infranet[21]	Mimicry	—	In band (private)	Low	—	Modulate sequence of HTTP requests to send data

show how to take advantage of such mismatches. In particular, they detail a number of ways to detect StegoTorus, SkypeMorph, and CensorSpoofer. None of these obfuscators were actually in use (although for reasons unrelated to security), and so these attacks were tested only against simulated stand-ins for deployed systems. It's possible that real deployments would be more (or less!) vulnerable to their attack methods.

We also note that the authors didn't measure the collateral damage of their tests—that is, to what extent the attacks they develop would misclassify benign cover protocol traffic as obfuscated. Subsequent work showed that some of their tests would lead to a false-positive rate of 37 percent if naively deployed, whereas others appear effective at avoiding collateral damage.[14]

Entropy Tests

The semantic and syntactic checks above won't help one identify the use of randomizing obfuscators, as these explicitly try to look like no actual protocol. Liang Wang and his colleagues evaluated the use of efficient entropy tests in the context of detecting randomizing obfuscators.[14] Their techniques focus on a small window of bytes occurring in the first few messages of an observed flow. The intuition for why this works is that the packets emitted by all nonobfuscated protocols always contain some plaintext headers, whereas randomizers by definition have random data in these locations. Their test against obfsproxy achieved a relatively low false-positive rate of only 0.2 percent when used against nonobfuscated network traces collected from a university campus. Whether this constitutes a sufficiently small false-positive rate in practice is both context dependent and unclear.

Traffic Analysis

Even without parsing packet payloads to exploit the actual byte values being carried, obfuscator-produced

Table 3. Comparison of network obfuscation approaches.* (cont.)

Obfuscator	Type	Deployment	Key exchange	Performance	Attack	Notes
Marionette[22]	Mimicry	—	Out of band	Moderate	—	Programmable mimicry including stateful and statistical properties
Non-PT Tor	Tunneling	Tor	In band (TLS)	High	See "How the Great Firewall of China Is Blocking Tor"[7]	Non-Tor TLS
obfs2 and obfs3	Randomization	Tor	Out of band	High	See "Seeing through Network-Protocol Obfuscation"[14]	Full encryption of messages
obfs4	Randomization	Tor, Lantern	In band (private)	High	See "Seeing through Network-Protocol Obfuscation"[14]	Uniform key exchange and full encryption of messages
ScrambleSuit	Randomization	Tor	In band (private)	High	See "Seeing through Network-Protocol Obfuscation"[14]	Uniform key exchange and full encryption of messages
SkypeMorph	Mimicry	—	Out of band (Skype)	Low	See "The Parrot Is Dead: Observing Unobservable Network Communications"[6]	Mimics Skype connections
StegoTorus	Mimicry	—	In band (private)	Low	See "The Parrot Is Dead: Observing Unobservable Network Communications"[6]	Mimics HTTP connections
Telex	Tunneling	—	In band (modified TLS)	High	—	Requires on-path cooperating ISP/ routers

** Types correspond to our classifications of randomizer, mimicry, or tunneling. Any notable deployments are listed. Performance is qualitative and based on numbers reported in the original work.*

traffic might exhibit packet lengths, interpacket timings, and directional traffic patterns that deviate from the background network traffic. Using such metadata to infer information about communications is *traffic analysis*.

One example of deviant traffic behavior arises in Meek. Conventionally, HTTPS connections involve two message flows—one smaller upstream request (client to server) and another longer downstream message containing the requested content. However, with Meek, the client must frequently send requests to see whether new data is available. To a network DPI system, this might stand out as an uncharacteristically large number of upstream messages. The ability to exploit this by a DPI was investigated by Wang and his colleagues, whose machine learning–based approach detected Meek with a true-positive rate of 0.98 and false-positive rate of 0.0002 or less. But their detector might not be easy to use in practice as it only worked well when trained and tested on traces from the same network environment.

Fearing traffic analysis, some obfuscators include mechanisms to try to randomize metadata such as packet lengths and packet interarrival time. Such countermeasures borrow techniques from the extensive literature on website fingerprinting and its countermeasures.[15] How effective these countermeasures are in the obfuscation context is unknown.

Challenges and Open Questions

So far, we've provided an overview of the rapid evolution of censorship and censorship evasion technologies in the context of network traffic obfuscation. Now

we turn to the many open questions and challenges that remain.

Understanding Users

Beyond anecdotes and folklore, very little is understood about the users of censorship circumvention tools, and this hamstrings the development of effective tools. Research that takes a principled approach to understanding users and their situations could lead to significant improvements. But performing user studies in this context is particularly challenging. The use of censorship circumvention tools is illegal in some places, and in extreme cases, exposing users could put them in physical danger.

Develop guidelines for user studies. The research community might produce guidelines for performing ethical human-subject research in this space. These guidelines would do well to account for regional and cultural differences, reflect the challenge of obtaining informed consent, and perform measurement in a transparent yet privacy-sensitive way.

Perform user studies. The guidelines could be put to use in the design of ethical research studies. Key questions that might be addressed include: What makes users decide to use a system or not? How do users first learn of the availability of tools and then obtain them? What are user perceptions regarding risk, ethics, and legality of tool use? How do users react to different kinds of performance or functionality loss when using circumvention tools? How important is it to users that they be able to convincingly deny that they used a tool?

Understanding Censors

Perhaps the most important theme emerging from this article is that researchers don't yet have a clear understanding of the adversary. This prevents us from knowing whether a proposed obfuscation method is truly "broken" or targets appropriate design tradeoffs among security, performance, and usability.

Characterize "normal" traffic. Circumvention technologies depend on sensitivity to collateral damage, but at present we lack models for how normal, unobfuscated traffic appears to DPI systems. This is especially important for mimicry-based obfuscators which, by design, try to avoid detection by appearing to be normal traffic.

So far, researchers have either generated synthetic datasets or, in one case, gathered traces from a university's networks.[14] The problem with synthetic traces, and even university network captures, is that they're unlikely to properly represent the wide diversity of traffic that censors observe.

One approach here would be to develop partnerships with large network operators (for instance, ISPs) to perform studies. We note that for obfuscation research, one unfortunately needs what's referred to as *full-packet capture*—that is, entire packet contents, including application-layer data. This could have significant privacy implications for network users, so special care is demanded. Perhaps researchers could develop new techniques for doing studies without ever storing packets, or incorporate other privacy mechanisms (for example, see "Bunker: A Privacy-Oriented Platform for Network Tracing"[16]). This would improve the privacy of such studies and, one hopes, reduce reluctance of network operators to help perform them.

One powerful artifact of such studies would be useful models of "normal" traffic in various settings. These could be used to build more accurate simulations to support further analysis and tool testing. What would make a model "useful" isn't currently well defined.

Models of collateral damage. Clearly our threat models must account for the sensitivity of censors to collateral damage. Underlying all censorship circumvention tools is the assumption that censors want to limit collateral damage, at least to some extent. There are a number of open questions here.

First is characterizing what types of collateral damage are of concern to censors. We expect that tolerance to false positives varies from country to country, and across time within a single country. Moreover, not all collateral damage is the same: it might be that causing collateral damage of certain forms is more costly to censors in terms of political or economic repercussions. Domain fronting systems, such as CloudTransport and Meek, rely on the hope that blocking popular cloud services, for instance, those provided by Google or Amazon, is untenable. However, some countries like China have blocked such services entirely for periods of time. A methodology for systems to reason about collateral damage is clearly needed.

The challenge is for models to incorporate parameters in a way that captures informal statements of the form "the censor won't tolerate more than an n percent false-positive rate." Few existing works attempt to measure false-positive rates methodically; thus, we have no way to state that a given technique will induce collateral damage that's intolerable to the modeled censor.

A starting point would be to build threat models parameterized by false-positive rates and standardized methods to identify network traffic. Researchers could then present receiver operating characteristic (ROC) curves with their approach. ROC curves and measures, such as area under the curve (AUC), are already standard practice in machine learning, and we recommend

bringing these forms of analysis to circumvention technology development.

Accessible censorship adversary labs. Given the widespread and increasing use of commercial DPI devices, good threat models should be informed by what devices can reasonably be expected to do now and in the near future—what computational tasks they can perform (regular expression matching, full parsing of packet payloads, or intensive statistical tests) and how much state they can keep (a single packet, all packets in a TCP connection, or all connections from a particular IP address). This assessment will be context dependent: it will be impacted by the volume of traffic and the assumed sensitivity to collateral damage. This suggests that a hierarchy of threat models should be developed to guide building useful systems, rather than attempting to get consensus on a single model.

A hurdle has been obtaining access to the kinds of equipment used by censors. DPI systems are expensive proprietary devices, and they come with restrictive licensing agreements that typically prevent any benchmarking. This means that researchers, even if they purchase a device, might be contractually obligated not to perform research with them. Although hardware can often be purchased secondhand on sites like eBay, it might contain out-of-date software and not fully reflect the abilities of fully updated deployments.

> **A hierarchy of threat models should be developed to guide building useful systems, rather than attempting to get consensus on a single model.**

The community would benefit from a concerted effort to overcome these hurdles, with the goal of building a widely usable censorship lab. The lab would need to obtain realistic, preferably state-of-the-art, devices with realistic configurations. Less restrictive licensing would need to be negotiated. We might look to previous scientific infrastructure projects for inspiration, such as PlanetLab, CloudLab, and the National Cyber Range. We envision such a lab acting as a repository for reference datasets of "normal" traffic (as discussed earlier) in addition to providing a battery of existing censorship tools. Ideally, the lab would support remote access for researchers.

Future-Proofing Obfuscation Systems

We expect that user studies and better understanding of censors will yield significant insights into how to build better circumvention systems. Eventually, our goal should be to adapt to, or even defuse, the current arms race and give developers the techniques needed for future-proof obfuscation mechanisms. By this we mean mechanisms that can maintain connectivity for users even in the face of adaptive censors that know the design of the system. We summarize the research directions needed to achieve this.

Develop formal foundations for obfuscation research. Currently, there's a dearth of formal abstractions—formal threat models in particular—on which to lay principled foundations for obfuscation. One approach is to follow modern cryptographic theory's lead, where precise adversarial models, security goals, and syntax for primitives are primary research considerations. This approach enables clear scientific discussion and objective comparison of systems relative to specified adversarial models. In short, good models are a significant aid in new tool development.

Unlike traditional cryptographic problems, where efficient constructions have been shown to meet very pessimistic adversarial models, the censorship circumvention setting might benefit from more realistic models, informed and evolved by experimental evidence and measurements from the field.

Build general frameworks. Obfuscation mechanisms have mostly been built with a particular circumvention system in mind. But obfuscation methods designed for a specific circumvention system might not be easily adapted for another. For example, Tor is TCP based, so Tor-oriented obfuscation methods need not consider unreliable transports, but I2P and uProxy must.

To have broader impact, the obfuscation community would benefit from general libraries and frameworks. In addition to cutting down on aggregate implementation effort, having broadly useful obfuscation libraries could facilitate rapid rollouts of countermeasures to new attacks—not every tool would need to have a different patch developed. Of course, building more general libraries will require community consensus on typical architectures, APIs, and protocols for negotiating which obfuscation to use.

Increase the use of encryption. Recent efforts to expand the use of standard encryption tools like TLS make it harder to perform censorship. If most of the Internet is end-to-end encrypted using TLS, then censors are harder pressed to use DPI to detect content they want to block. Indeed, we view improved censorship

circumvention as a nice ancillary benefit to the already laudable goal of ubiquitous encryption on the Internet.

However, recall our earlier warning that encryption isn't a silver bullet for censorship circumvention. Particular applications that use common encryption protocols can still be fingerprinted, often by information sent in the clear such as headers, packet sizes, and timing.

Censors are also sometimes positioned to break end-to-end security, performing man-in-the-middle (MITM) attacks either by directly abusing the web's public-key infrastructure, as occurred in Turkey in 2012,[17] or by forcing clients to install root certificates that trick systems into allowing MITM proxies. The latter is common in corporate settings already, and retooling those solutions to nation-state levels is part of the threat landscape that should be considered.

Simultaneously, encryption standards could do more to support obfuscation. For example, they could encrypt more of the handshake required during connection establishment. Ideally, all bits emitted would be completely random-looking to the censor. (This is targeted by the obfs4 system.) If all TLS connections achieved this, then the entropy-based attacks discussed earlier would no longer be effective.

Censorship circumvention technology is a relatively young field of research but already has a wide diversity of approaches and systems. This suggests the need to reflect and consolidate what's been learned and to propose open questions to facilitate the field's development.

Perhaps the most striking observation is that researchers don't yet have a clear and common model of the adversary. We also noted significant challenges to understanding users and making obfuscation systems robust to future attacks. These limitations prevent researchers from knowing whether a proposed obfuscation method is vulnerable in practice (versus just in theory) and whether a design has made appropriate tradeoffs among security, performance, and ease of use, or it's failing to address user needs.

These open questions are in large part due to a surfeit of methodological challenges that researchers in this area face. Chief among these are the difficulty of obtaining access to network traffic without violating privacy, procuring DPI systems used by censors, and performing user studies in relevant populations. Such challenges make principled progress difficult but also offer opportunities for creative workarounds.

Unfortunately, Internet censorship will be around for the foreseeable future. Researchers, activists, and governments supporting the right to unfettered information online should continue to work toward a future in which network traffic obfuscation prevails in the face of sophisticated, well-provisioned censors. This will add a critical piece to the technosocial puzzle enabling people around the world access to information. ■

Acknowledgments

We thank the participants of the 2014 Obfuscation Workshop, which was organized by Google, for their valuable contributions.

References

1. D. Fifield et al., "Blocking-Resistant Communication through Domain Fronting," *Proc. Privacy Enhancing Technologies* (PETS 15), 2015, pp. 46–64.
2. A. Houmansadr et al., "Cirripede: Circumvention Infrastructure Using Router Redirection with Plausible Deniability," *Proc. 18th ACM Conf. Computer and Communications Security* (CCS 11), 2011, pp. 187–200.
3. J. Karlin et al., "Decoy Routing: Toward Unblockable Internet Communication," *USENIX Free and Open Communications on the Internet* (FOCI 11), 2011; www.usenix.org/legacy/event/foci11/tech/final_files/Karlin.pdf.
4. E. Wustrow et al., "Telex: Anticensorship in the Network Infrastructure," *Proc. 20th USENIX Conf. Security* (USENIX Security 11), 2011, p. 30.
5. "Iran Partially Blocks Encrypted Network Traffic," Tor blog, 10 Feb. 2012; blog.torproject.org/blog/iran-partially-blocks-encrypted-network-traffic.
6. A. Houmansadr, C. Brubaker, and V. Shmatikov, "The Parrot Is Dead: Observing Unobservable Network Communications," *IEEE Symp. Security and Privacy*, 2013, pp. 65–79.
7. P. Winter and S. Lindskog, "How the Great Firewall of China Is Blocking Tor," *USENIX Free and Open Communications on the Internet* (FOCI 12), 2012; www.usenix.org/conference/foci12/workshop-program/presentation/winter.
8. "Tor: Pluggable Transports," Tor, 2011; www.torproject.org/docs/pluggable-transports.html.en.
9. B. Wiley, *Dust: A Blocking-Resistant Internet Transport Protocol*, tech. report, 2011; blanu.net/Dust.pdf.
10. P. Winter, T. Pulls, and J. Fuss, "Scramblesuit: A Polymorphic Network Protocol to Circumvent Censorship," *Proc. ACM Workshop Privacy in the Electronic Society* (WPES 13), 2013, pp. 213–224.
11. Z. Weinberg et al. "Stegotorus: A Camouflage Proxy for the Tor Anonymity System," *Proc. ACM Conf. Computer and Communications Security* (CCS 12), 2012, pp. 109–120.
12. H. Mohajeri Moghaddam et al., "Skypemorph: Protocol Obfuscation for Tor Bridges," *Proc. ACM Conf. Computer and Communications Security* (CCS 12), 2012, pp. 97–108.
13. D. Luchaup et al., "LibFTE: A Toolkit tor Constructing Practical, Format-Abiding Encryption Schemes," *USENIX Security Symp.* (USENIX Security 12), 2014; www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-luchaup.pdf.

14. L. Wang et al., "Seeing through Network-Protocol Obfuscation," *Proc. ACM SIGSAC Conf. Computer and Communications Security* (CCS 15), 2015, pp. 57–69.
15. M. Juarez et al., "A Critical Evaluation of Website Fingerprinting Attacks," *Proc. ACM SIGSAC Conf. Computer and Communications Security* (CCS 14), 2014, pp. 263–274.
16. A.G. Miklas et al., "Bunker: A Privacy-Oriented Platform for Network Tracing," *USENIX 6th Symp. Networked Systems Design and Implementation* (NSDI 09), 2009, pp. 29–42.
17. A. Langley, "Enhancing Digital Certificate Security," Google blog, 3 Jan. 2013; googleonlinesecurity.blogspot.com/2013/01/enhancing-digital-certificate-security.html.
18. C. Burbaker, A. Houmansadr, and V. Shmatikov, "CloudTransport: Using Cloud Storage For Censorship-Resistant Networking," *Privacy Enhancing Technologies*, LNCS 8555, Springer, 2014, pp. 1–20.
19. S. Burnett, N. Feamster, and S. Vempala, "Chipping Away at Censorship Firewalls with User-Generated Content," *Proc. 19th USENIX Conf. Security* (USENIX Security 10), 2010, pp. 463–468.
20. D. Fifield et al., "Evading Censorship with Browser-Based Proxies," *Privacy Enhancing Technologies*, LNCS 7382, Springer, 2012, pp. 239–258.
21. N. Feamster, "Infranet: Circumventing Web Censorship and Surveillance," *Proc. 11th USENIX Security Symp.* (USENIX Security 02), 2002, pp. 247–262.
22. K.P. Dyer, S.E. Coull, and T. Shrimpton, "Marionette: A Programmable Network-Traffic Obfuscation System," *Proc. 24th USENIX Security Symp.* (USENIX Security 15), 2015; www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-dyer.pdf.

Lucas Dixon is a chief scientist at Jigsaw. His research interests include machine learning, automated reasoning, and quantum information. Dixon received a PhD in informatics from the University of Edinburgh. Contact him at ldixon@google.com.

Thomas Ristenpart is an associate professor at Cornell Tech. His research interests include security, privacy, and cryptography. Ristenpart received a PhD in computer science from the University of California, San Diego. Contact him at ristenpart@cornell.edu.

Thomas Shrimpton is an associate professor at the University of Florida. His research interests include cryptography and security. Shrimpton received a PhD in electrical engineering from the University of California, Davis. Contact him at teshrim@cise.ufl.edu.