CloudTransport: Using Cloud Storage for Censorship-Resistant Networking

Chad Brubaker[1,2], Amir Houmansadr[2], and Vitaly Shmatikov[2]

[1] Google
[2] The University of Texas at Austin

Abstract. Censorship circumvention systems such as Tor are highly vulnerable to network-level filtering. Because the traffic generated by these systems is disjoint from normal network traffic, it is easy to recognize and block, and once the censors identify network servers (e.g., Tor bridges) assisting in circumvention, they can locate all of their users.
CloudTransport is a new censorship-resistant communication system that hides users' network traffic by tunneling it through a cloud storage service such as Amazon S3. The goal of CloudTransport is to increase the censors' economic and social costs by forcing them to use more expensive forms of network filtering, such as large-scale traffic analysis, or else risk disrupting normal cloud-based services and thus causing collateral damage even to the users who are not engaging in circumvention. CloudTransport's novel passive-rendezvous protocol ensures that there are no direct connections between a CloudTransport client and a CloudTransport bridge. Therefore, even if the censors identify a CloudTransport connection or the IP address of a CloudTransport bridge, this does not help them block the bridge or identify other connections.
CloudTransport can be used as a standalone service, a gateway to an anonymity network like Tor, or a pluggable transport for Tor. It does not require any modifications to the existing cloud storage, is compatible with multiple cloud providers, and hides the user's Internet destinations even if the provider is compromised.

1 Introduction

Internet censorship is typically practiced by governments [3,45,53] to, first, block citizens' access to certain Internet destinations and services; second, to disrupt tools such as Tor that help users circumvent censorship; and, third, to identify users engaging in circumvention. There is a wide variety of censorship technologies [30]. Most of them exploit the fact that circumvention traffic is easy to recognize and block at the network level. Traffic filtering is cheap, effective, and has little impact on other network services and thus on the vast majority of users in the censorship region who are not engaging in circumvention. Another problem with the existing censorship circumvention systems is that they cannot survive partial compromise. For example, a censor who learns the location of a Tor bridge [6] can easily discover the locations of all of its users simply by enumerating the IP addresses that connect to the bridge.

While there is no comprehensive, accurate data on the technical capabilities of real-world censors, empirical evidence suggests that they typically perform only line-speed or close-to-line-speed analysis of Internet traffic. In particular, they neither store huge Internet traces for a long time, nor carry out resource-intensive statistical analysis of all observed flows. Furthermore, many state-level censors appear unwilling to annoy regular users, who are not engaged in circumvention, by significantly disrupting popular services—even if the latter employ encrypted communications. This is especially true of services used by businesses. For example, Chinese censors are not blocking GitHub because of its popularity among Chinese users and the gigantic volume of traffic they generate [17], nor are they blocking some of Google's encrypted services [19].

Some censors are willing to risk popular discontent by taking more drastic measures. Ethiopia has been reported to block Skype [13] (denied by the Ethiopian government [14]), Iran occasionally blocks SSL [26], and the Egyptian government cut the country off the Internet entirely during an uprising [12]. We focus on the more common scenario where, instead of blocking all encrypted communications, the censors aim to distinguish censorship circumvention traffic from "benign" encrypted traffic and block only the former.

Our contributions. We design, implement, and evaluate CloudTransport, a new system for censorship-resistant communications. CloudTransport is based on the observation that public cloud storage systems such as Amazon S3 provide a very popular encrypted medium accessible from both inside and outside the censor-controlled networks. For example, Amazon's cloud services are already used to host mirrors of websites that are censored in China, yet Chinese censors are not blocking Amazon because doing so would disrupt "thousands of services in China" with significant economic consequences [20].

CloudTransport is a general-purpose networking system that uses cloud storage accounts as passive rendezvous points in order to hide network traffic from censors. Since censors in economically developed countries like China are not willing to impose blanket bans on encrypted cloud services—even if these services are known to be used for censorship circumvention [20]—they must rely on network filters to recognize and selectively block circumvention traffic. CloudTransport uses exactly the same cloud-client libraries, protocols, and network servers as any other application based on a given cloud storage (we refer to this property as *entanglement*). Consequently, simple line-speed tests that recognize non-standard network protocols are not effective against CloudTransport.

CloudTransport's passive-rendezvous protocol helps survive partial compromise. Because CloudTransport clients never connect to a CloudTransport bridge directly, a censor who discovers a CloudTransport connection or learns the IP address of a bridge can neither block this bridge, nor identify its other users. The bridge can also transparently move to a different IP address without any disruption to its clients (e.g., if it experiences a denial of service attack). Our rendezvous protocol may be useful to other censorship resistance systems, too.

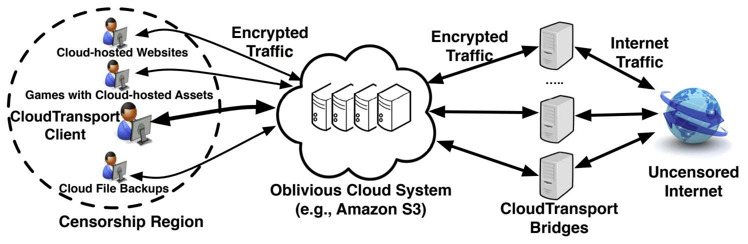


Fig. 1. High-level architecture of CloudTransport.

CloudTransport is versatile and lets the user select a trusted cloud storage provider in a jurisdiction of the user's choice. On the user's machine, it presents a universal socket abstraction that can be used as a standalone communication system, a gateway for accessing proxies or Tor, or a pluggable transport for Tor.

The goal of CloudTransport is to raise the economic and social costs of censorship by forcing the censors to use statistical traffic analysis and other computationally intensive techniques. False positives of statistical traffic classification may cause the censors to disrupt other cloud-backed services such as enterprise applications, games, file backups, document sharing, etc. This will result in collateral damage, make censorship tangible to users who are not engaging in circumvention, and increase their discontent.

We analyze the properties provided by CloudTransport against ISP-level censors, cloud providers, and compromised bridges. We also show that its performance is close to Tor pluggable transports on tasks such as Web browsing, watching videos, and uploading content.

2 Protocol Design

The overall architecture of CloudTransport is shown in Fig. 1. The user installs CloudTransport client software on her machine and creates a *rendezvous account* with a cloud storage provider such as Amazon S3 in a jurisdiction of her choice outside the censor's control. The user must also choose a CloudTransport bridge and send the rendezvous account's access credentials to the bridge via the bootstrapping protocol described in Section 3. We envision CloudTransport bridges being run by volunteers in uncensored ISPs. A natural place to install CloudTransport bridges is on the existing Tor bridges [6], so that CloudTransport users benefit from Tor's anonymity properties in addition to the censorship circumvention properties provided by CloudTransport.

On the user's machine, the CloudTransport client presents a socket that can be used by any application for censorship-resistant networking. For example, the user may run a Web browser or a conventional Tor client over CloudTransport. The CloudTransport client uses the cloud storage provider's standard client library to upload application-generated network packets to the rendezvous account; the bridge collects and delivers them to and from their destinations.

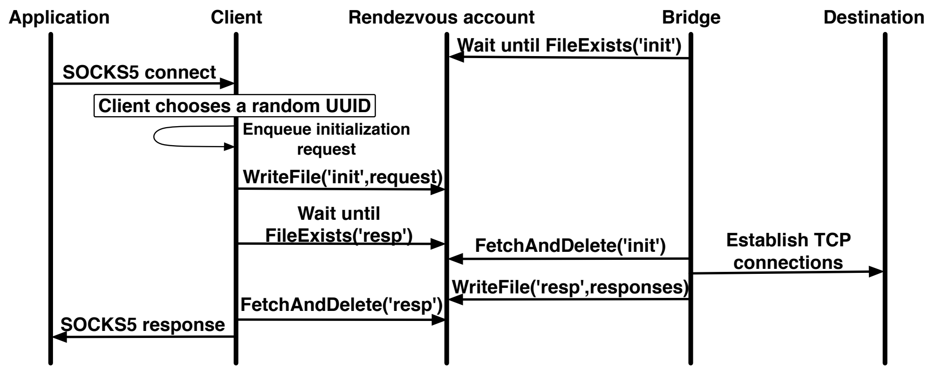

Application
Client
Rendezvous account
Bridge
Destination
Wait until FileExists('init')
SOCKS5 connect
Client chooses a random UUID
Enqueue initialization request
WriteFile('init',request)
Wait until FileExists('resp')
FetchAndDelete('init')
Establish TCP connections
WriteFile('resp',responses)
FetchAndDelete('resp')
SOCKS5 response

Fig. 2. Cirriform: connection initialization.

CloudTransport uses existing cloud storage services "as is," without any modifications. This is a challenge because cloud-storage APIs are designed for occasional file uploads with many downloads, not for fast sharing of data between two parties. They do not typically support file locking or quick notification of file changes. CloudTransport clients and bridges, on the other hand, write to cloud storage often and must learn as quickly as possible when the other party has uploaded data to the shared account. To solve this challenge, each file used by CloudTransport is written by only one connection and read by only one connection. Writes happen only if the file does not already exist and all reads delete the file, to signal that it is safe to create the file anew and write into it.

We designed and implemented two variants of CloudTransport, Cirriform and Cumuliform. The protocol flow is the same, the only difference is how often they write into the cloud-based rendezvous account and poll for updates.

Cirriform. Cirriform uses one file in the rendezvous account per connection per direction, plus one file per direction for connection setup.

Figure 2 shows the protocol for setting up a new Cirriform connection. Connection requests and responses are queued and uploaded in batches. The client and the bridge periodically check the rendezvous account for pending messages. Once the connection is established, Figures 3 and 4 show how data is transferred from the application and the destination, respectively.

Typical cloud-storage API does not support pushing storage updates to customers, thus the client and the bridge must poll the rendezvous account. In our prototype, the polling rate for initialization requests and responses is set randomly and independently by each client, with the expected value of once per 0.5 seconds. For maximum performance, polling for data connections starts at once per 0.1 seconds, halves after every 20 failed checks, and resets to once per 0.1 seconds after every successful check. To avoid generating a regular signal, random jitter is added or subtracted to the interval after each poll.

Cumuliform. Applications such as Web browsing create many parallel connections, and polling cloud storage on all of them can incur a non-trivial cost

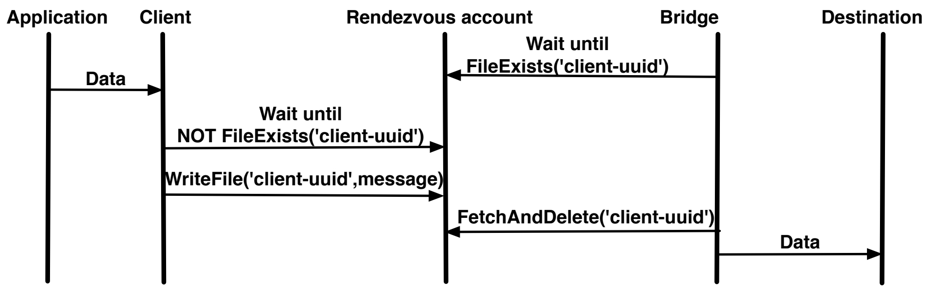

Application
Client
Rendezvous account
Bridge
Destination
Wait until FileExists('client-uuid')
Data
Wait until NOT FileExists('client-uuid')
WriteFile('client-uuid',message)
FetchAndDelete('client-uuid')
Data

Fig. 3. Cirriform: client sending data.

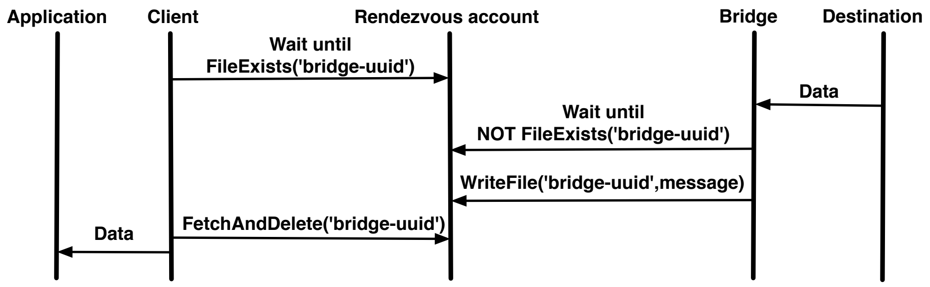

Application
Client
Rendezvous account
Bridge
Destination
Wait until FileExists('bridge-uuid')
Data
Wait until NOT FileExists('bridge-uuid')
WriteFile('bridge-uuid',message)
FetchAndDelete('bridge-uuid')
Data

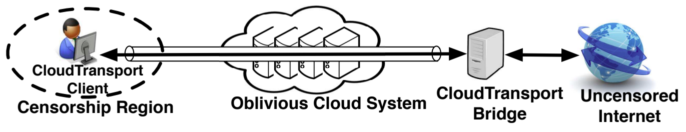
Fig. 4. Cirriform: destination sending data.

Table 1. Prices charged by cloud storage providers (2013).

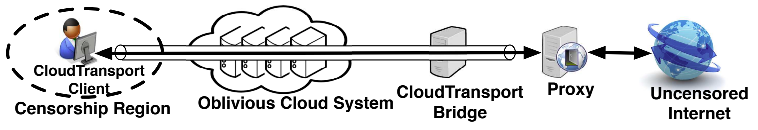
Provider	Bandwidth cost	Storage cost	Operation cost
Amazon S3	\$0.12/GB after first GB	\$0.0950/GB	\$0.004/10000 GET \$0.005/1000 PUT
Rackspace CloudFiles	\$0.12/GB after first GB	\$0.1000/GB	None
Google Cloud Storage	\$0.12/GB (USA/Europe) \$0.21/GB (Asia/Pacific)	\$0.0865/GB	\$0.01/10000 GET \$0.01/1000 PUT

if the provider charges per operation (see Table 1). To reduce the polling cost, Cumuliform uses one file per direction rather than per connection. All requests are enqueued; the client and the bridge check 5 times a second for pending requests. Unlike Cirriform, which uploads data as soon as it is ready, Cumuliform uploads in batches, which can add extra delays.

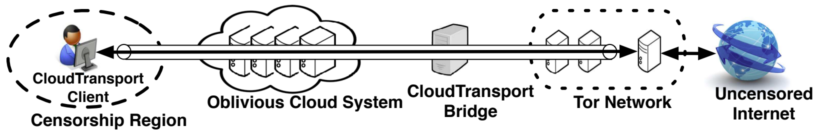
Usage modes. CloudTransport can be used directly to send and receive network packets. We refer to this as the *transport mode*. The transport mode does not provide any privacy against the cloud storage provider since the provider can observe all of the user's packets in plaintext. To provide some protection against



(a) Tunnel mode



(b) Proxified-light mode



(c) Proxified-Tor mode

Fig. 5. Usage modes of CloudTransport.

malicious or curious cloud providers and CloudTransport bridges, we developed three usage modes illustrated in Figure 5. These modes represent different points in the tradeoff space between performance and censorship resistance.

The *tunnel mode* of CloudTransport hides the user's Internet destinations—but not the fact that she is using CloudTransport —from the cloud provider. In this mode, the user uses a CloudTransport bridge as a gateway to censored destinations. The traffic between the user's CloudTransport client and the bridge is encrypted, preventing the cloud provider from observing traffic contents. The bridge runs an OpenSSH server and authenticates the client using the temporary public key from the client's bootstrapping ticket (see Section 3.2). The client connects to this server via the rendezvous account, as described in Section 2, and tunnels all of its traffic over SSH.

In the *proxified-light mode*, the client uses CloudTransport to access a one-step proxy, e.g., Anonymizer [2]. The user's activities are thus hidden from the bridge if the traffic between the client and the proxy is encrypted end-to-end.

For strongest privacy, the client can use a system that aims to provide protection against itself, e.g., the Tor anonymity network in conjunction with CloudTransport. In the *proxified-Tor mode*, the client either runs a conventional Tor client and forwards Tor traffic over CloudTransport, or else uses CloudTransport as a pluggable transport [39] for Tor.

3 Bootstrapping

Bootstrapping is a critical part of any circumvention system. Many systems [4,7,25,35,37,39,51] must send their clients some secret information—for example, IP addresses of circumvention servers or bridges, URLs of websites covertly serving censored content, etc.—and hope that this information does not fall into the censors' hands. As shown in [33, 34], censors can easily obtain these secrets by pretending to be genuine users and then block the system. Existing, trusted clients can help bootstrap new clients [49, 50], but this limits the growth of the system, especially in the early stages. Another way for the clients to discover circumvention servers is by probing the Internet [23, 54].

By contrast, bootstrapping in CloudTransport is initiated by users and performed "upstream": clients send information to the bridges without needing to obtain any secrets first. Therefore, insider attacks cannot be used to block CloudTransport bridges or discover other users.

3.1 Selecting a cloud provider and a bridge

To start using CloudTransport, the user must set up a rendezvous account with a cloud storage provider. The user should select a cloud storage provider which is (1) outside the censor's jurisdiction, (2) already used by many diverse applications unrelated to censorship circumvention, and (3) unlikely to cooperate with the censor. We believe that using a cloud storage account for CloudTransport does not violate the typical terms of service, e.g., Amazon S3's "Conditions of Use" [1] or Dropbox's "Acceptable Use Policy" [9], since CloudTransport does not cause harm to other users or the provider itself.

Global providers such as Amazon S3 let customers specify a region for their data, e.g., "US West (Oregon)", "Asia Pacific (Tokyo)", etc. To evade flow correlation attacks discussed in Section 4.4, a CloudTransport bridge should access its clients' rendezvous accounts through the cloud provider's servers located outside the censorship region.

Due to the distributed nature of cloud storage, there is a delay between uploading a file and this file becoming visible for download, as well as other temporary inconsistencies between customers' views of the same account. This is typically a non-issue for conventional uses of cloud storage, but the primary source of delays for CloudTransport. Delays are much smaller and consistency achieved much faster by services such as Amazon S3 that charge per storage operation, as opposed to services such as Google Drive that simply charge per amount of storage regardless of how frequently this storage is accessed.

The monetary costs of using cloud storage is another consideration (see Table 1). We hope that some providers would be willing to donate their storage services (e.g., in the form of free accounts) to support censorship resistance.

The user must also select a CloudTransport bridge. Unlike Tor bridges [6], which must remain hidden from the censors, the list of CloudTransport bridges, along with other information needed for their usage, can be publicly advertised. It can be hosted on a directory server similar to the directory server of Tor relays [48]. For each CloudTransport bridge, this public directory should contain (1) a certificate with the bridge's public key, and (2) the URL of the bridge's *dead drop*, whose purpose is explained in Section 3.3.

We distinguish between the *login credentials* (e.g., username and password) and *access credentials* (e.g., API Key and Access Key in Amazon S3) for the rendezvous account. Access credentials allow reading and writing files, but do not give access to management data such as the billing information, IP addresses from which the account was accessed, etc. Only the access credentials for the rendezvous account should be sent to the bridge. The user can do this via one of the methods described in Section 3.3.

3.2 Creating a bootstrapping ticket

To use a bridge, a CloudTransport client first obtains the bridge's public key K_B from CloudTransport's directory server. The client then creates a *bootstrapping ticket* with (1) the name of the cloud provider chosen by the user, (2) the access credentials for the rendezvous account (API Key and Access Key in the case of Amazon S3), and (3) optionally, the client's temporary public key, which is used in the tunnel mode (Section 2) to authenticate the client. The ticket is encrypted using K_B as an S/MIME [42] message in the `EnvelopedData` format.

3.3 Delivering the ticket to the bridge

Dead drop. A bridge can set up its own cloud storage account, create a "dead drop" in it as a world-readable and -writable file directory, and advertise its URL in the bridge directory. Clients will write their tickets into the dead drop as files with arbitrary names and the bridge will periodically collect them.

To protect tickets in network transit from tampering, the dead drop should be accessible via HTTPS only (most cloud storage services use HTTPS by default). Unlike rendezvous accounts used for actual networking, bootstrapping is not latency-sensitive, thus free services like Dropbox, SkypeDrive, or Google Drive can be used to set up the dead drop.

Out-of-band channels. Since latency is not critical for bootstrapping, a user can deliver her bootstrapping ticket to the bridge by asking a trusted friend who is already using CloudTransport, or by posting the ticket to an anonymous chat room, social network, or public forum.

4 Analysis

Table 2 shows what information CloudTransport aims to hide from, respectively, the censoring ISP, cloud storage provider, and CloudTransport bridges. The cloud storage provider is trusted not to reveal to the censors the identities and network locations of its customers who are using CloudTransport. The bridges are trusted not to perform flow correlation (see Section 4.4). In the tunnel mode, the bridges must also be trusted not to reveal the contents and destinations of CloudTransport traffic; this assumption is not required in the proxified modes.

In the rest of this section, we discuss how CloudTransport resists different types of attacks that may violate these properties.

Table 2. Intended properties of CloudTransport.

	Users' ISP	Cloud storage provider	CloudTransport bridge
Network locations of CloudTransport users	Hidden	Known	Hidden
Destinations of Cloud-Transport traffic	Hidden	Hidden	Known (tunnel mode) Hidden (proxified modes)
Content of Cloud-Transport traffic	Hidden	Hidden	Known (tunnel mode) Hidden (proxified modes)

4.1 Recognizing CloudTransport network traffic

CloudTransport aims to increase the technological complexity of censorship and, in particular, to force censors into using computationally expensive techniques such as statistical traffic analysis [10] as opposed to simple network-level tests.

Protocol discrepancies. CloudTransport's encrypted tunnels use exactly the same clients, same protocols, and same network servers as any other application based on a given cloud storage API. Due to this "entanglement" property, CloudTransport is immune to attacks that find discrepancies [21, 47] between genuine protocols like SSL and Skype and the imitations used by systems such as Tor and SkypeMorph [35]. This significantly raises the burden on the censors because simple line-speed tests based on tell-tale differences in protocol headers, public keys, etc. cannot be used to recognize CloudTransport. Also, CloudTransport's reaction to active perturbations such as dropping and delaying packets is similar to any other application based on the same cloud API.

The network servers used by Tor, SkypeMorph, Obfsproxy [37] and similar systems are disjoint from those used by other services. Once these servers are discovered, censors can block them without zero impact on non-circumvention users and their traffic. By contrast, blocking the network servers used by CloudTransport would effectively disable all uses of a given cloud provider, causing economic damage to users and businesses in the censorship region [20].

Statistical analysis. We do not claim that no statistical classification algorithm can distinguish CloudTransport traffic from the traffic generated by other cloud applications. We believe, however, that it will be technically challenging for the censors to develop an algorithm that simultaneously achieves low false negatives (to detect a significant fraction of CloudTransport traffic) and low false positives (to avoid disrupting non-CloudTransport cloud services).

First, note an important difference between the encrypted cloud traffic and the encrypted traffic generated by Skype and other standalone applications. All of Skype traffic is generated by copies of the same client or, at most, a few variations of the same client. Therefore, censors can whitelist typical Skype patterns and block all traffic that deviates from these patterns (this includes traffic generated by Skype imitators such as SkypeMorph or Stegotorus [21]).

By contrast, encrypted traffic to the cloud provider's servers is generated by thousands of diverse applications. This makes it difficult to create an accurate whitelist of traffic patterns and block all deviations without disrupting permitted services. Instead, censors must rely on blacklisting and use statistical analysis to positively recognize traffic patterns characteristic of CloudTransport. Furthermore, this analysis must be performed on every cloud connection, increasing the censors' computational burden.

Detailed analysis of traffic patterns generated by CloudTransport vs. all the diverse uses of cloud storage is beyond the scope of this paper. The main challenge for accurate statistical recognition of CloudTransport traffic is that CloudTransport is unlikely to account for more than a tiny fraction of all monitored connections. Due to the base-rate fallacy inherent in detecting statistically rare events, we expect that even an accurate classifier will either fail to detect many CloudTransport connections, or occasionally confuse CloudTransport with another cloud service. In the former case, some CloudTransport traffic will escape detection. In the latter case, censorship will cause collateral damage to at least some non-CloudTransport cloud applications. This will make censorship visible to non-circumvention users and potentially disrupt cloud-based business services, thus increasing the economic and social costs of censorship.

4.2 Abusing the CloudTransport bootstrapping protocol

The dead-drop variant of the CloudTransport bootstrapping protocol described in Section 3.3 can be potentially abused by censors to deny service to bona fide CloudTransport users. Since bridges publicly advertise their dead drops, censors can read and write them like any other user.

Even though reading other users' tickets does not reveal who these users are because the tickets are encrypted under the bridge's public key, censors may delete or tamper with them in order to deny service to genuine users. Fortunately, many cloud storage providers store all versions of each file (e.g., a free Dropbox account keeps all file versions for 30 days[1]). Therefore, the bridge should collect the first version of every file in the dead drop.

Censors may also stuff the dead drop with tickets that contain credentials for non-existing rendezvous accounts or real rendezvous accounts that are never used. The bridge will be forced to repeatedly poll these accounts, potentially exhausting its resources. To partially mitigate these attacks, the bridge backs off on polling if the account remains inactive (see Section 2). If the rate at which the censors can stuff the dead drop with fake tickets is significantly higher than the rate at which the bridge can check and discard them, this attack may hinder the bootstrapping process.

[1] https://www.dropbox.com/help/11/en

4.3 Attacking a CloudTransport bridge

It is relatively easy for the censors to discover the IP addresses of CloudTransport bridges. For example, a censor can pretend to be genuinely interested in circumvention, pick a bridge, set up a rendezvous account, and find out the bridge's IP address from the account's access logs.

CloudTransport clients do not connect to bridges directly. Therefore, the censors cannot discover CloudTransport clients by simply enumerating all IP addresses inside the censorship region that connect to the bridges' addresses. For the same reason, blacklisting the addresses of known bridges has no effect on CloudTransport if these addresses are outside the censorship region. Unless the censors take over a bridge, they cannot observe or disrupt the connections between this bridge and the cloud provider because these connections take place entirely outside the censorship region (see Fig. 1 and Section 3.1).

Censors may stage a denial-of-service attack by flooding the IP address of a known bridge with traffic. In addition to standard defenses against network denial of service, some operators may be able to move their bridges to another IP address. This change is completely transparent to the users: as long as the bridge is hosted at an address from which it can access the cloud storage, CloudTransport remains operational even if the users don't know this address. Censors may also pose as genuine clients and send large volumes of requests via CloudTransport, but this involves heavy use of rendezvous accounts and will incur significant monetary costs. Furthermore, a bridge can throttle individual clients.

A denial-of-service attack on the bridge may cause a correlated drop in traffic on CloudTransport connections utilizing that bridge, and thus help the censors recognize CloudTransport connections by finding these correlations. This attack requires large-scale traffic analysis, which will be more expensive for the censors than simply enumerating all clients connecting to a bridge.

Finally, the censors may create their own bridge or take over an existing bridge. In either case, they gain full visibility into the traffic passing through this bridge, including the access credentials for the rendezvous accounts of all CloudTransport users communicating through the bridge. These credentials do not directly reveal these users' identities or network locations. Furthermore, the proxified modes of CloudTransport (see Section 2) encrypt traffic end-to-end between the client and the apparent destination: either a proxy, or a Tor entry node. Consequently, the censors in control of a bridge do not learn the true destinations or contents of CloudTransport traffic.

By controlling the bridge, the censors gain the ability to perform flow correlation attacks—see Section 4.4. Furthermore, the censors in control of a bridge can write content into rendezvous accounts that is legally prohibited in the cloud provider's jurisdiction. They can then use the presence of such content to shut down the accounts and/or convince the cloud provider to ban CloudTransport.

4.4 Performing large-scale flow correlation

A censor who observes all traffic to and from the cloud storage provider may attempt to identify flows that belong to the same CloudTransport connection by correlating packet timings and sizes [8, 22] In particular, the censor may look for flows between a user and the cloud provider that are correlated with the flows between the provider and a known or suspected CloudTransport bridge. A precondition for this attack is the ability to observe the traffic between the provider and the bridge. As explained in Section 3.1, we assume that the bridge is connecting to the provider through a server located outside the censorship region. That said, flow correlation can be feasible if the censors set up their own bridges or compromise an existing bridge.

Flow correlation is resource-intensive. Passive correlation attacks [8] require recording hundreds of packets from each flow and cross-correlating them across all flows. Active correlation [22] requires fine-grained perturbations and delays to be applied to all suspected flows. Furthermore, correlation must be done separately and independently for each flow reaching a given bridge.

The censor may attempt a side-channel attack such as website fingerprinting [5, 38, 44] to infer websites being browsed over CloudTransport. This attack exploits patterns in object sizes which are preserved by encryption. Random padding used by some SSH[2] [43] (respectively, TLS) implementations greatly complicates this attack against CloudTransport's tunnel (respectively, proxified-light) mode. Tor's use of equal-sized cells mitigates this attack in the proxified-Tor mode, but may not completely prevent it [5, 38]. To address this, Tor pluggable transports use traffic morphing [28], replaying old traffic traces [35, 51], and format-transforming encryption [11]. A CloudTransport client, too, can deploy these countermeasures, which can be hosted on users' machines [31, 32] or network proxies [31, 41], at the cost of additional bandwidth overhead.

5 Performance

We evaluated CloudTransport on four use cases: browsing the front pages of the Alexa Top 30 websites, uploading 300 KB images via SCP to a remote server, watching 5 minutes of 480p streaming video from Vimeo, and uploading a 10MB video to YouTube. All experiments involved a single client and a single bridge. The client was running on a machine with 16 Mb down- and 4 Mb up-bandwidth, while the bridge was running in a datacenter 2,400 kilometers (1,500 miles) away. Evaluating the performance of CloudTransport in a realistic, large-scale deployment is a topic of future work.

Table 3. *Browsing*, per-page costs.

Provider	Cirriform	Cumuliform	Cirriform Profixied	Cumuliform Profixied
S3	0.00240¢	0.00100¢	0.00300¢	0.00430¢
CloudFiles	0	0	0	0
Cloud Storage	0.00570¢	0.00234¢	0.00600¢	0.00900¢

[2] http://www.gnutls.org/manual/gnutls.html#On-Record-Padding

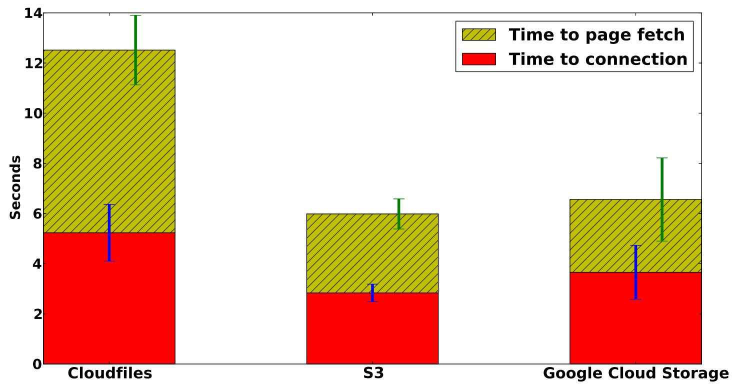


Fig. 6. *Browsing* (different providers).

Fig. 6 compares different cloud storage providers with CloudTransport operating in the tunnel mode. Table 3 shows the corresponding costs. Amazon S3 and Google Cloud Storage have similar performance and costs; S3 is slightly cheaper and quicker to propagate changes. RackSpace CloudFiles does not charge per operation and is thus much cheaper, but also significantly slower.

All of the following experiments were performed with a rendezvous account hosted on Amazon S3.

Performance. Fig. 7 shows that the performance of Cirriform in tunnel and profixied-Tor modes is similar to Tor with Obfsproxy [37]. Note that in the proxified-Tor mode, CloudTransport traffic enters the Tor network after passing through the bridge and is therefore subject to the same performance bottlenecks as any other Tor traffic. Unlike CloudTransport, Tor+Obfsproxy is easily recognizable at the network level and thus marked "(observable)" in the charts.

Cumuliform is noticeably slower because it buffers messages for all connections (as many as 30 when browsing). The variance for CloudTransport is much lower than for Tor+Obfsproxy, mainly because delays in CloudTransport are due to waiting for data to become available in the rendezvous account and S3 has fairly consistent delays in propagating small files used by CloudTransport.

Uploading files involves a lot of back-and-forth communication to set up the SCP connection. This puts CloudTransport at a disadvantage because of its per-message overheads, but Fig. 8 shows that it still outperforms Tor+Obfsproxy in all modes but one. Uploading a video to Youtube has similar issues to uploading small images, but with larger data sizes and more back-and-forth communication. Fig. 9 shows that CloudTransport still outperforms Tor+Obfsproxy in all Cirriform modes. Cumuliform in tunnel and proxified-Tor modes is, respectively, similar to and slower than Tor+Obfsproxy.

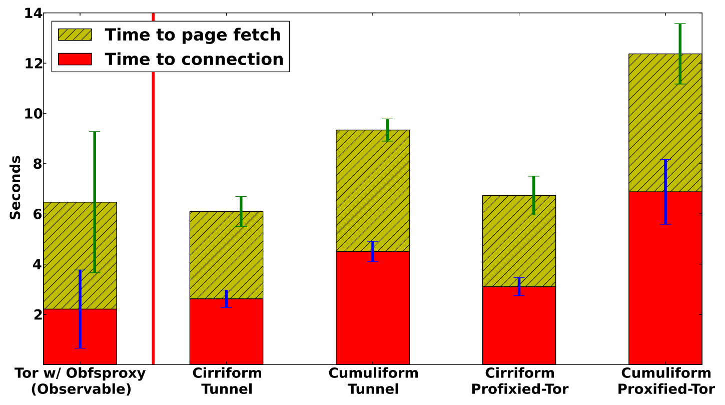


Fig. 7. *Browsing* (different usage modes).

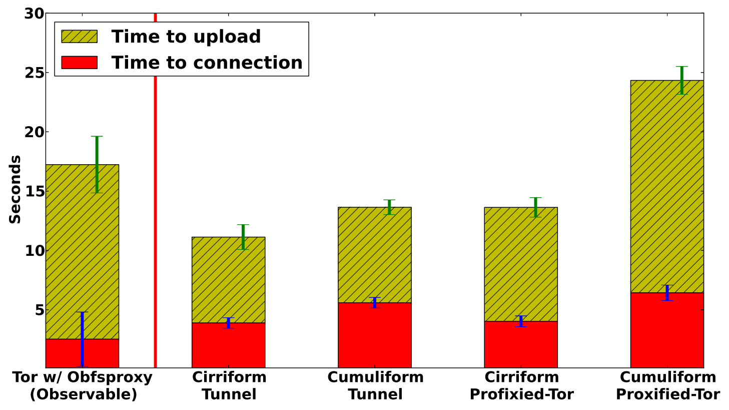


Fig. 8. *Image uploading.*

CloudTransport in all modes consistently plays streaming videos without pause after some initial buffering. Tor+Obfsproxy starts playing earlier but often buffers again later in the clip. Fig. 10 shows the average time spent buffering.

Bandwidth. CloudTransport connections have minimal bandwidth overhead per message: 350-400 bytes for S3, 700-800 for CloudFiles, and 375-450 for Google Cloud Storage. HTTPS uploads and downloads have extra 2-3% overhead. When Cirriform polls an S3 account 3 times per second and 5 times per second per connection, its total overhead is 1.2KB + 2KB/connection per second.

Costs. Cirriform's performance is consistently superior to Cumuliform in all modes, but Cumuliform uses many fewer operations and is thus almost half as cheap when using providers who charge per operation (Table 3). In profix-

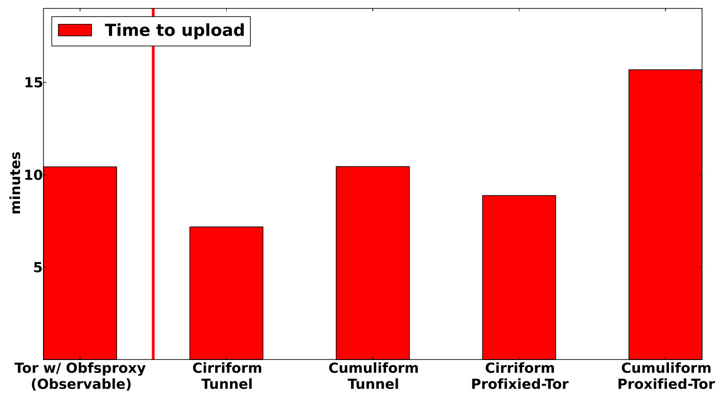


Fig. 9. *Youtube Uploading.*

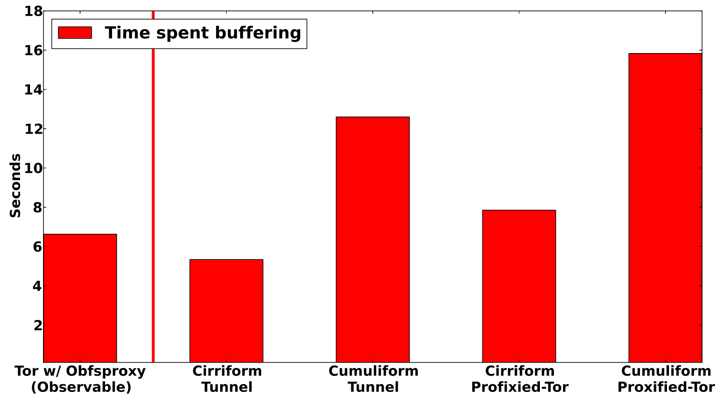


Fig. 10. *Streaming Video.*

ied modes, connections are re-used, thus Cumuliform no longer enjoys the cost advantage. Cirriform's polling costs are higher because it takes longer to run.

Table 4. Idle-polling costs.

Provider	Cirriform	Cumuliform
S3	\$0.185/day + \$0.34/day/connection	\$0.34/day
CloudFiles	0	0
Cloud Storage	\$0.215/day + \$0.86/day/connection	\$0.86/day

The costs of idle-polling the rendezvous account regardless of whether communication is taking place are shown in Fig. 4. These assume one write per every poll and are thus worst-case estimates. Real costs will be lower because uploads to cloud storage propagate slower than CloudTransport's polling rate.

6 Related Work

Proxy-based systems. IP address blacklisting is the most basic technique used by many censors [30]. A natural way to circumvent the filter is to access blacklisted destinations via a proxy, e.g., Psiphon [40]. GoAgent [18] is an HTTP proxy implemented as a cloud-hosted application in Google App Engine [16]. In contrast to CloudTransport, GoAgent has access to all of the user's traffic in plaintext and must be fully trusted.

The main challenge for proxies is how to securely distribute their locations to genuine users while keeping them secret from *insider attackers*, i.e., censors pretending to be genuine users [33, 50]. As soon as the censor learns the proxy's location, he can blacklist it, identify past users from network traces, or even leave the proxy accessible in order to identify and punish its future users [34].

Tor bridges. Tor is a popular anonymity network [7]. Cloud-based Onion Routing (COR) [27] is a proposal to host Tor relays in the cloud. Whether hosted in the cloud or not, the addresses of Tor relays are public, thus censors can and do block them. A Tor bridge is a hidden proxy that clients can use as a gateway to the Tor network [6]. The Tor Cloud project [46], currently deployed by Tor, allows donors to run Tor bridges inside Amazon EC2. This idea was previously proposed by Mortier et al. [36] in a position paper.

CloudTransport does not involve running relays or bridges in the cloud; it uses the cloud solely as a passive rendezvous point for data exchange. This gives CloudTransport several advantage over Tor bridges, Tor Cloud, COR, etc.

First, Tor traffic is easily recognizable at the network level because Tor clients and bridges run their own unique protocol. Iranian censors were able to block Tor by exploiting the difference between the Diffie-Hellman moduli in "genuine" SSL and Tor's SSL [47, Slide 27], as well as the expiration dates of Tor's SSL certificates [47, Slide 38]. By contrast, CloudTransport uses exactly the same protocol, cloud-client library, and network servers as any other application based on a given cloud storage service.

Second, blacklisting the IP address of a Tor bridge completely disables this bridge with zero impact on other network services. By contrast, blacklisting the IP addresses of CloudTransport bridges has no effect on CloudTransport, while blacklisting the IP addresses of cloud servers used by CloudTransport disrupts other cloud-based applications using the same servers.

Third, a censor who discover the IP address of a Tor bridge (e.g., via a probe [52, 53] or insider attack [33, 34, 50]) can easily enumerate the network locations of clients who connect to this bridge. By contrast, even a censor in complete control of a CloudTransport bridge does not learn the locations of its clients without computationally intensive flow correlation analysis.

Fourth, when a Tor bridge changes its IP address (e.g., when it is attacked or blacklisted), all of its clients must be securely notified about the new address. By contrast, when a CloudTransport bridge changes its IP address, this change is completely transparent to its clients.

Fifth, bootstrapping Tor bridges is challenging because their addresses must be distributed only to genuine users but not to censors pretending to be users. By contrast, bootstrapping in CloudTransport is initiated by clients. Even if a censor pretends to be a user, he cannot discover who the other users are.

CloudTransport's reliance on rendezvous accounts hosted by cloud storage providers has some disadvantages, too. Unlike Tor clients, which only require Internet access, CloudTransport clients require every user to set up a cloud storage account outside her region. This may negatively impact usability, impose financial costs, generate a pseudonymous profile, and disclose the user's identity and the fact that she is using CloudTransport to the cloud storage provider, as well as the financial institutions processing her payments.

Imitation systems. To remove characteristic patterns from Tor traffic, Tor deployed *pluggable transports* [39]. For example, Obfsproxy [37] re-encrypts Tor packets to remove content identifiers. Systems such as SkypeMorph [35], StegoTorus [51], and CensorSpoofer [49] proposed pluggable transports that aim to imitate popular network protocols like Skype and HTTP. A recent study showed multiple flaws in the entire approach of unobservability-by-imitation [21].

Hide-within systems. A promising alternative to imitation is to actually run a popular protocol and hide circumvention traffic within its network channels, thus entangling circumvention and non-circumvention traffic. This ensures that the circumvention system is "bug-compatible" with a particular implementation of the chosen protocol and therefore immune to tests that find discrepancies between actual protocol implementations and partial imitations [21].

We call such systems *hide-within*. CloudTransport is a hide-within system that tunnels circumvention traffic through cloud storage protocols. Other hide-within designs include FreeWave [24], which encodes circumvention traffic into acoustic signals sent over VoIP connections, and SWEET [25], which tunnels circumvention traffic inside email messages.

Steganography-based systems. In Infranet [15], the client pretends to browse an unblocked website that has secretly volunteered to serve censored content. Requests for censored content are encoded in HTTP requests, the responses are encoded in images returned by the site. By contrast, CloudTransport uses cloud storage obliviously, without any changes to the existing services.

Collage [4] hides censored content in user-generated photos, tweets, etc. on public, oblivious websites. It does not support interactive communications such as Web browsing.

In decoy routing [23, 29, 54], ISPs voluntarily help circumvention by having their routers recognize covert, steganographically marked traffic generated by users from the censorship region and deflect it to the blocked destinations specified by the senders. Unlike CloudTransport, decoy routing is not deployable without cooperation from at least some ISPs in the middle of the Internet.

7 Conclusions

We presented the design and implementation of CloudTransport, a new system for censorship-resistant communications. CloudTransport hides network traffic from censors by reading and writing it into rendezvous accounts on popular cloud-storage services. It can be used as a standalone networking medium or as a pluggable transport for Tor, enhancing Tor's censorship resistance properties.

Unlike Tor, SkypeMorph, and other systems utilizing network bridges to assist in circumvention, CloudTransport can survive the compromise of one or more of its bridges because its rendezvous protocol does not reveal the locations and identities of CloudTransport users even to the bridge.

CloudTransport aims to increase the economic and social costs of censorship. Empirical evidence shows that censors in relatively developed countries like China are not willing to impose a blanket ban on encrypted cloud services even when these services are used for censorship circumvention [20]. Because CloudTransport uses exactly the same network tunnels and servers as the existing cloud-based applications, censors can no longer rely on simple line-speed tests of protocol-level discrepancies to recognize and selectively block CloudTransport connections. Instead, they must perform statistical classification of every cloud connection. In contrast to systems like Tor, which can be recognized and blocked with zero impact on the vast majority of users, any false positives in the censors' recognition algorithms for CloudTransport will disrupt popular and business-critical cloud services. This will make censorship visible and increase discontent among the users who are not engaging in censorship circumvention.

Acknowledgements

This research was supported by the Defense Advanced Research Projects Agency (DARPA) and SPAWAR Systems Center Pacific, Contract No. N66001-11-C-4018, NSF grant CNS-0746888, and a Google research award.

References

1. Amazon: Conditions of Use. http://www.amazon.com/gp/help/customer/display.html?ie=UTF8&nodeId=508088.
2. Anonymizer. https://www.anonymizer.com/.
3. Joining China and Iran, Australia to Filter Internet. http://www.foxnews.com/scitech/2009/12/15/like-china-iran-australia-filter-internet.
4. S. Burnett, N. Feamster, and S. Vempala. Chipping Away at Censorship Firewalls with User-Generated Content. In *USENIX Security*, 2010.
5. X. Cai, X. Zhang, B. Joshi, and R. Johnson. Touching from a Distance: Website Fingerprinting Attacks and Defenses. In *CCS*, 2012.
6. R. Dingledine and N. Mathewson. Design of a Blocking-Resistant Anonymity System. https://svn.torproject.org/svn/projects/design-paper/blocking.html.
7. R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-generation Onion Router. In *USENIX Security*, 2004.

8. D. Donoho, A. Flesia, U. Shankar, V. Paxson, J. Coit, and S. Staniford. Multi-scale Stepping-Stone Detection: Detecting Pairs of Jittered Interactive Streams by Exploiting Maximum Tolerable Delay. In *RAID*, 2002.
9. Dropbox: Acceptable Use Policy. https://www.dropbox.com/terms#acceptable_use.
10. M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli. Tunnel Hunter: Detecting Application-layer Tunnels with Statistical Fingerprinting. *Computer Networks*, 53(1):81–97, 2009.
11. K. Dyer, S. Coull, T. Ristenpart, and T. Shrimpton. Protocol Misidentification Made Easy with Format-transforming Encryption. In *CCS*, 2013.
12. Egypt Leaves the Internet. http://www.renesys.com/blog/2011/01/egypt-leaves-the-internet.shtml.
13. Ethiopia Bans Skype, Other VoIP Services. http://www.sudantribune.com/spip.php?article42946.
14. Ethiopia: Govt Denies Banning Skype and Other Internet Communication Services. http://allafrica.com/stories/201206250202.html.
15. N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. Karger. Infranet: Circumventing Web Censorship and Surveillance. In *USENIX Security*, 2002.
16. Google App Engine. https://developers.google.com/appengine/.
17. China's GitHub Censorship Dilemma. http://mobile.informationweek.com/80269/show/72e30386728f45f56b343ddfd0fdb119/.
18. GoAgent proxy. https://code.google.com/p/goagent/.
19. Google Transparency Report. http://www.google.com/transparencyreport/traffic/.
20. Activists Say They Have Found Way Round Chinese Internet Censorship. http://www.theguardian.com/world/2013/nov/18/activists-chinese-internet-censorship-mirror-sites.
21. A. Houmansadr, C. Brubaker, and V. Shmatikov. The Parrot Is Dead: Observing Unobservable Network Communications. In *S&P*, 2013.
22. A. Houmansadr, N. Kiyavash, and N. Borisov. RAINBOW: A Robust And Invisible Non-Blind Watermark for Network Flows. In *NDSS*, 2009.
23. A. Houmansadr, G. Nguyen, M. Caesar, and N. Borisov. Cirripede: Circumvention Infrastructure Using Router Redirection with Plausible Deniability. In *CCS*, 2011.
24. A. Houmansadr, T. Riedl, N. Borisov, and A. Singer. I Want My Voice to Be Heard: IP over Voice-over-IP for Unobservable Censorship Circumvention. In *NDSS*, 2013.
25. A. Houmansadr, W. Zhou, M. Caesar, and N. Borisov. SWEET: Serving the Web by Exploiting Email Tunnels. In *PETS*, 2013.
26. Iran Reportedly Blocking Encrypted Internet Traffic. http://arstechnica.com/tech-policy/2012/02/iran-reportedly-blocking-encrypted-internet-traffic.
27. N. Jones, M. Arye, J. Cesareo, and M. Freedman. Hiding Amongst the Clouds: A Proposal for Cloud-based Onion Routing. In *FOCI*, 2011.
28. G. Kadianakis. Packet Size Pluggable Transport and Traffic Morphing. Tor Tech Report 2012-03-004, 2012.
29. J. Karlin, D. Ellard, A. Jackson, C. Jones, G. Lauer, D. Mankins, and W. Strayer. Decoy Routing: Toward Unblockable Internet Communication. In *FOCI*, 2011.
30. C. Leberknight, M. Chiang, H. Poor, and F. Wong. A Taxonomy of Internet Censorship and Anti-censorship. http://www.princeton.edu/~chiangm/anticensorship.pdf, 2010.

31. Z. Li, T. Yi, Y. Cao, V. Rastogi, Y. Chen, B. Liu, and C. Sbisa. WebShield: Enabling Various Web Defense Techniques without Client Side Modifications. In *NDSS*, 2011.
32. X. Luo, P. Zhou, E. Chan, W. Lee, R. Chang, and R. Perdisci. HTTPOS: Sealing Information Leaks with Browser-side Obfuscation of Encrypted Flows. In *NDSS*, 2011.
33. D. McCoy, J. A. Morales, and K. Levchenko. Proximax: A Measurement Based System for Proxies Dissemination. In *FC*, 2011.
34. J. McLachlan and N. Hopper. On the Risks of Serving Whenever You Surf: Vulnerabilities in Tor's Blocking Resistance Design. In *WPES*, 2009.
35. H. Moghaddam, B. Li, M. Derakhshani, and I. Goldberg. SkypeMorph: Protocol Obfuscation for Tor Bridges. In *CCS*, 2012.
36. R. Mortier, A. Madhavapeddy, T. Hong, D. Murray, and M. Schwarzkopf. Using Dust Clouds to Enhance Anonymous Communication. In *IWSP*, 2010.
37. A Simple Obfuscating Proxy. https://www.torproject.org/projects/obfsproxy.html.en.
38. A. Panchenko, L. Niessen, A. Zinnen, and T. Engel. Website Fingerprinting in Onion Routing Based Anonymization Networks. In *WPES*, 2011.
39. Tor: Pluggable Transports. https://www.torproject.org/docs/pluggable-transports.html.en.
40. Psiphon. http://psiphon.ca/.
41. C. Reis, J. Dunagan, H. Wang, O. Dubrovsky, and S. Esmeir. BrowserShield: Vulnerability-Driven Filtering of Dynamic HTML. *TWEB*, 1(3):11, 2007.
42. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification. http://www.ietf.org/rfc/rfc3851.txt, 2004.
43. The Secure Shell (SSH) Transport Layer Encryption Modes. http://www.ietf.org/rfc/rfc4344.txt, 2006.
44. Q. Sun, D. R. Simon, Y. Wang, W. Russell, V. Padmanabhan, and L. Qiu. Statistical Identification of Encrypted Web Browsing Traffic. In *S&P*, 2002.
45. Syria Tightens Control over Internet. http://www.thenational.ae/news/world/middle-east/syria-tightens-control-over-internet.
46. The Tor Cloud Project. https://cloud.torproject.org/.
47. How Governments Have Tried to Block Tor. https://svn.torproject.org/svn/projects/presentations/slides-28c3.pdf.
48. Tor Directory Servers and Their URLs. https://silvertunnel.org/doc/tor-directory-server-urls.html.
49. Q. Wang, X. Gong, G. Nguyen, A. Houmansadr, and N. Borisov. CensorSpoofer: Asymmetric Communication Using IP Spoofing for Censorship-Resistant Web Browsing. In *CCS*, 2012.
50. Q. Wang, Z. Lin, N. Borisov, and N. Hopper. rBridge: User Reputation Based Tor Bridge Distribution with Privacy Preservation. In *NDSS*, 2013.
51. Z. Weinberg, J. Wang, V. Yegneswaran, L. Briesemeister, S. Cheung, F. Wang, and D. Boneh. StegoTorus: A Camouflage Proxy for the Tor Anonymity System. In *CCS*, 2012.
52. T. Wilde. Knock Knock Knockin' on Bridges' Doors. https://blog.torproject.org/blog/knock-knock-knockin-bridges-doors, 2012.
53. P. Winter and S. Lindskog. How the Great Firewall of China Is Blocking Tor. In *FOCI*, 2012.
54. E. Wustrow, S. Wolchok, I. Goldberg, and J. Halderman. Telex: Anticensorship in the Network Infrastructure. In *USENIX Security*, 2011.