Ephemeral Exit Bridges for Tor

Zhao Zhang, Tavish Vaidya
Georgetown University

Kartik Subramanian
Jericho High School

Wenchao Zhou, Micah Sherr
Georgetown University

***Abstract*—This paper examines an existential threat to Tor—the increasing frequency at which websites apply discriminatory behavior to users who arrive via the anonymity network.**

Our main contribution is the introduction of Tor *exit bridges*. Exit bridges, constructed as short-lived virtual machines on cloud service providers, serve as alternative egress points for Tor and are designed to bypass server-side censorship. Due to the proliferation of managed cloud-based desktop services (e.g., Amazon Workspaces), there is already a surprisingly large fraction of web requests that originate in the cloud. Trivially disrupting exit bridges by blocking requests from the cloud would thus lead to significant collateral damage.

Our experiments demonstrate that exit bridges effectively circumvent server-side blocking of Tor with low overhead. Additionally, we perform a cost-analysis of exit bridges and show that even a large-scale deployment can be done at low cost.

***Keywords*-Tor; Bridge; Exit; Server-side; Blocking;**

I. Introduction

Tor [12] is used by millions of daily users [27, 49], most of whom use the anonymity network to privately browse the web [27]. Blocking access to the relays that comprise the Tor network is trivial: the network locations of the relays are public (to allow for source routing) and can be straightforwardly enumerated and subsequently blocked. To prevent such blocking, the Tor Project has responded by developing new obfuscation protocols [16, 31, 54] that allow Tor clients to covertly communicate with *bridge relays*—Tor relays whose network locations are not advertised by the directory servers.

This paper explores a complementary threat to Tor. Rather than study how censors prevent their citizenries from accessing Tor, we focus on attempts to block Tor *at the destination*. We are interested in the degree to which websites and hosting providers discriminate against Tor traffic (for example, by blocking it) and how users can circumvent such techniques by concealing that their traffic has been relayed through Tor.

As prior work has shown [26, 41], some websites prevent access from Tor. Although motivations differ, such blocking is commonly done due to the inclusion of Tor exit relays in subscribed IP blacklists (that is, Tor relays are the collateral damage of subscribing to such lists) or the site operators' beliefs that a disproportionate amount of attack traffic flows through Tor [26]. Our measurements show that the rate of server-side filtering has increased since these prior studies were conducted, and that that approximately 8% of Alexa top 10,000 sites [4] either significantly alter content for Tor users or block Tor traffic entirely.

We argue that this trend represents an existential threat to Tor. Simply put, as more sites block access from Tor users, we posit that more users will abandon the anonymity service. Our belief is buoyed by recent work that shows that approximately 80% of Tor visits are to the Alexa top 1 million sites [27]; based on their measurements, we estimate that more than 4.8% of Tor traffic would go to the blocked sites. We anticipate that a large fraction of Tor users will indeed be disenfranchised as top sites continue to block access and Tor becomes increasingly unable to provide access to the desired content.

The main contribution of this paper is the design, implementation, and roll-out of *ephemeral exit bridges* (or simply, *exit bridges*). The aim of an exit bridge is to disguise the fact that Tor is being used by providing a temporary egress point for Tor traffic. Exit bridges are ephemeral in that they are short-lived and thus difficult to blacklist.

Exit bridges are inspired by, but are distinct from, traditional (ingress) Tor bridges. Traditional bridges operate at fixed (albeit unadvertised) network locations, and are the target of adversaries intent on preventing people from using Tor.

In contrast, exit bridges are more similar to domain fronting [21] and operate as ephemeral virtual machines (VMs) on popular cloud service providers. They assume users who can access the Tor network (perhaps using a traditional bridge) but are otherwise stymied by server-side discrimination against requests originating from the Tor network. The threat model for exit bridges is thus more constrained and assumes a more corporate (as opposed to nation-state) adversary who operates or hosts a website. Whereas traditional bridges are required for all Tor connections when direct access to Tor is blocked, exit bridges can be applied in a more ad hoc fashion, as needs dictate; e.g., their use can be reserved for sites that otherwise block access from Tor.

We perform an extensive evaluation of exit bridges and show that they *enable* access while imposing little overhead. The use of exit bridges permitted us to access nearly all tested sites, even those that block connections from Tor. Exit bridges incur additional latency; however, this cost is overwhelmed by Tor's overall end-to-end latency.

At first blush it may seem trivial for sites to effectively block exit bridges by preventing incoming requests from cloud providers since sites could expect few requests that originate *from* the cloud. However, we argue that such a strategy will likely cause significant collateral damage given the proliferation of virtualized desktop services such as Amazon Workspace. We show that a surprising amount of web traffic already originates from the cloud, thus making it a good mixing ground for exit bridge traffic.

We consider the anonymity implications of exit bridges, and

find that coalescing egress traffic at cloud service providers increases vulnerability to traffic correlation [25, 33, 37] attacks. Such a threat is roughly analogous to that faced by domain fronting techniques (e.g., meek [45]), although in our case a rogue (or honest-but-curious) cloud operator learns the sites being visited as opposed to the clients who request them.

As a final contribution, we explore the operational costs of deploying exit bridges. Unlike domain fronting systems that are expensive to operate, we argue that the cost of operating an exit bridge is sufficiently low that it can be fully-funded by its users. Here, we use nascent web revenue services [23] that pay website site operators a small amount each time their visitors complete a short online task. We demonstrate that the revenue is sufficient for operating an exit bridge, while imposing only a modest time commitment on the bridge user.

II. Background and Related Work

Tor is an overlay network that provides anonymity by routing user requests through volunteer operated nodes called *relays*. The Tor client typically forms anonymous *circuits*, each of which consists of a *guard*, *middle*, and *exit* relay; the guard and exit respectively serve as ingress and egress points for the anonymity network. On receiving the user's requested destination, the exit relay establishes a TCP connection to the desired destination and forwards traffic over this TCP connection. The destination perceives the Tor exit relay to be the requesting client, rather than the actual Tor user.

When a Tor relay comes online, it publishes various information about itself, including its public key fingerprint and network address, to the Tor *directory authorities*. This information is then consolidated into signed *consensus documents* that can be retrieved by Tor clients to discover the relays that comprise the network, enabling source routing.

Blocking traffic from Tor. Online services can easily block traffic coming from Tor. By simply creating a blacklist of IP addresses belonging to exit relays (obtained via the consensus document), sites can effectively block requests from Tor.

Existing research has looked into the extent of blocking of Tor users by online services. Khattak et al. [26] provide a systematic evaluation of Tor exit blocking and the differential treatment of Tor users by online services. Their results show that at least 1.3 million IPs block traffic from Tor exits at the TCP/IP layer. Khattak et al. also found that 3.67% of top Alexa 1000 websites perform blocking or discrimination of Tor traffic. Their work points out that most online services inherit the blocking behavior of their hosting providers while only a few sites employ their own blocking mechanisms.

Singh et al. [41] extend Khattak et al.'s work by measuring the extent of blocking of sites' search and login functionalities. Their findings show that 20% of Alexa top 500 websites discriminate against traffic coming from Tor users.

Related work. Tor was originally intended to allow users to more privately browse the web by separating a user's identity from its network location [12]. However, with various actors trying to block access to the Tor network [55], Tor has extended its original focus to also take on the role of a censorship-circumvention tool.

Most of the relevant literature focuses on censors' efforts to block access to the Tor network. Preventing access to guards is trivial since the list of relays is publicly available from the directory consensus. To mitigate such blocking, the Tor Project maintains a separate list of relays, called *bridges*, whose IPs are not publicly advertised and are thus more difficult to enumerate. (However, existing work has shown that discovering bridge IP addresses is not especially difficult [11, 15].)

Even if the locations of the bridges are not public, censors can easily identify Tor protocol traffic through traffic analysis or active probing techniques and can thus still restrict access to Tor bridges [24, 51, 55]. As a countermeasure, Tor supports pluggable transports [48] that obfuscate Tor traffic between Tor users and bridges. Various pluggable transports have been proposed [16, 17, 20, 31, 46, 47, 54, 56] and deployed [16, 20, 46, 48, 56], each of them using different schemes to conceal Tor traffic patterns. For example, the meek [20] pluggable transport uses HTTP to relay Tor traffic via cloud-based domain fronting [21], using TLS to hide the underlying Tor protocol. Using pluggable transports requires both the Tor user and the bridge to support a particular transport. (We impose similar requirements for exit bridges.)

Davidson et al. [10] examine how some users are treated unfairly by content providers because they happen to share a common IP address with a large pool of users, some of whom may be malicious. For example, it was previously the case that Cloudflare imposed multiple CAPTCHAs on Tor users due to the relatively high volume of malicious activity that it received from the Tor network [40]. Davidson et al. propose PrivacyPass, a cryptographic protocol that allows users to earn "tokens" by providing solutions to a challenge; these tokens can then be exchanged in the future without interacting with the challenge again. However, PrivacyPass requires the participation of the service provider. In contrast, exit bridges are designed as a more general solution and do not require the support of either the requested website or its hosting provider.

III. Design Principles

We next present an overview of exit bridges, starting with a motivating analysis of the state of server-side discrimination against Tor.

A. The state of server-side blocking of Tor

To understand the status quo of how severely Tor traffic is blocked or censored at various destination websites, we conduct an analysis of Tor's accessibility to Alexa's top 10,000 websites. For each website, we perform three consecutive HTTP/S requests and collect their responses: one direct request without using Tor (**Direct**$_1$) from a local machine, one request through Tor (**Tor**), and finally another direct request (**Direct**$_2$) from same local machine. The reason for performing two direct requests is to identify sites that serve significantly different content for each request (see Figure 1).

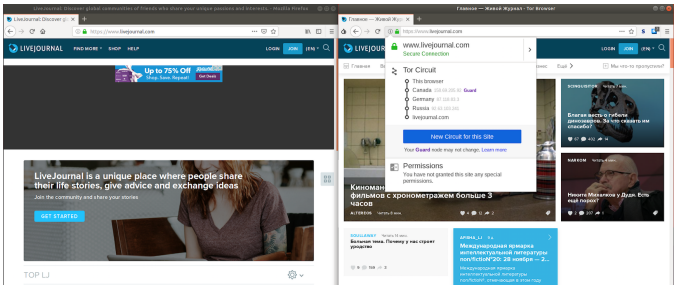


Fig. 1: A webpage (livejournal.com) that renders different dynamic content. Comparing the left and right screenshots, different cover stories, article feeds, and languages are presented for different users.

We use two techniques to measure the frequency at which sites block Tor. We first consider the HTTP response codes returned by the server in the **Direct**$_1$ and **Tor** configurations. We treat a web request as successful if and only if we receive a 2xx or 3xx HTTP response code (which signifies success). We consider a site to block Tor if we receive an HTTP response code that indicates success for **Direct**$_1$, but not for **Tor**.

We also consider the similarity between a page fetched directly (**Direct**$_1$) or via **Tor**. Here, we quantify similarity using the *HTML similarity score* metric [22, 28], where a score of 1 indicates that two HTML documents are identical, and a score of 0 reflects completely different content (e.g., one request returns an empty response). We consider only sites that returned 2xx or 3xx HTTP response codes when their top pages were fetched via **Direct**$_1$ and whose returned content was valid HTML. We choose a conservative threshold and consider a site to be blocked if (i) the HTML similarity score of the site's top page between **Direct**$_1$ and **Tor** is below 0.01, indicating that the two returned HTML pages are almost entirely different and (ii) the HTML similarity score of the site's top page between **Direct**$_1$ and **Direct**$_2$ is above 0.01. The latter check reduces false positives by not considering sites that serve radically different content based on geolocations.

We use both pycurl and Selenium [44, 53] to simulate a user's activity of browsing a webpage. pycurl generates simple HTTP/S requests and does not process Javascript. We instrument Selenium with Firefox's geckodriver (version 0.24.0) and fully render pages in the headless browser.

Table I reports the *block rates*—the percentage of top Alexa 10,000 sites that we found to block Tor—for both our HTTP response header and HTML similarity-based techniques. We find that the two techniques yield similar block rates, but that the block rates differ substantially between pycurl and Selenium. This latter difference is likely due to the Cloudflare Onion Service (COS) [40], which allows TorBrowser to bypass CAPTCHAs that Cloudflare otherwise imposes on Tor users that access its hosted sites.[1] Selenium (using the Firefox driver) is compatible with COS; pycurl is not.

[1]Cloudflare introduced the Cloudflare Onion Service (COS) in September 2018 [40]. Prior to its release, sites hosted on the popular Cloudflare platform served CAPTCHAs to users arriving via Tor circuits. COS uses HTTP Alternative Services [35], which is available in recent versions of the Tor Browser, to redirect Tor users to hidden service versions of Cloudflare-hosted sites and avoid CAPTCHAs.

	Header	HTML Similarity
pycurl	26.86%	27.94%
Selenium	7.99%	9.86%

TABLE I: Tor block rates for Alexa's top 10,000 sites, including sites that present CAPTCHAs.

	Header	HTML Similarity
pycurl	8.01%	8.97%
Selenium	7.39%	9.21%

TABLE II: Tor block rates for Alexa's top 10,000 sites, after removing sites that present CAPTCHAs.

When assessing the HTML similarity, only static page content has been examined. Is it possible that the embedded objects or Javascripts exhibit the discrimination of anonymous access? It would be useful to clarify if the static content is sufficient.

We conducted experiments using both pycurl and selenium. While pycurl returns only the static HTML contents, selenium in fact renders any Javascript inside the HTML. The results are presented in Table I and Table II. We observed that the results for the pycurl and selenium experiments are not identical, which is reasonable since the Javascript may heavily modify a page's DOM. However, the main observation result stays consistent: the two methods report similar blocking rates and the ephemeral exit bridges are effective in bypassing the server-side blocking.

We further refine our results by excluding sites that serve CAPTCHAs to Tor users, since presumably users could still access these sites if they are willing to solve puzzles. Table II shows these filtered results. Here, our results are generally consistent both between web clients and block detection techniques. *Overall, we find that approximately 8% of Alexa top sites blocks Tor.* The modest differences in HTML similarity scores between the two crawlers is due to the manner in which pages are constructed: pycurl doesn't retrieve the embedded web objects, while Selenium does a full rendering of the page (including Javascript).

To sanity-check our automated technique, we additionally randomly sampled 100 sites from the Alexa list and then manually evaluated whether the sampled sites discriminated against Tor. To conduct our manual analysis, we loaded each site directly without using Tor and contemporaneously requested the site through the TorBrowser. We found that 11% of the sampled sites either blocked Tor or timed-out when using the TorBrowser, which generally agrees with the findings of our more comprehensive automated analysis.

Need to explain more about the motivation of the work. What will be the impact of the work for the industry and users?

Server side blocking is becoming popular and is a potential existential threat for Tor. Tor effectively becomes useless if it cannot be used to access the web. This work presents a viable solution to mitigating server-side blocking, ensuring that Tor users will continue to have unfettered access to information.

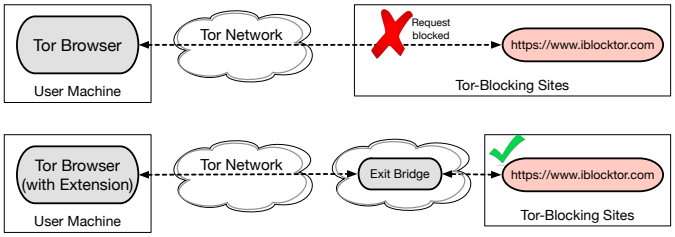


Fig. 2: Visiting a Tor-blocking website through Tor *(top)* without exit bridges and *(bottom)* with exit bridges.

Our findings agree with prior studies [26, 41] that similarly show high rates of server-side blocking of Tor traffic. As such blocking becomes more commonplace, we posit that more users may turn away from using Tor, since they will become increasingly unable to access content on the Internet. This, in turn, will endanger users' anonymity as they either decide to directly access content (at the expense of losing anonymity) or move to less studied (and sometimes dubious) methods of achieving anonymity.

B. Threat model

We consider the threat model adopted by Tor [12]. Briefly, we assume an adversary whose goal is to compromise the anonymity of Tor users. The adversary can observe, inject, modify, delete, or delay the traffic within its reach.

The adversary may also operate or compromise some of the Tor relays. Finally, we allow the adversary to compromise the entire exit bridge infrastructure.

As a focus of this paper, we additionally consider the threat posed by adversaries that, as content providers, aim to discriminate against Tor users by providing *differentiated services* between Tor and non-Tor users. (In the remainder of this paper, we use the shorthand "block" to denote any instance in which a website discriminates against Tor users.) We assume this adversary has full knowledge of the operation of the Tor network, has access to publicly available information, such as the list of all Tor relays, but has no control and limited visibility as to how network traffic is routed on the Internet. This is a reasonable assumption given that such adversaries are typically web content providers rather than ISPs or other network-level adversaries, only the latter of which may be willing or able to attack the network's routing infrastructure [42, 43]. In our threat model, the adversary can block all traffic that comes *directly* from the Tor network by filtering all traffic originating from Tor exit relays.

C. Design goals

The intuition behind our system design is simple: since Tor-blocking websites mostly rely on blacklisting the IPs of Tor exit relays, we can simply add one (or more) extra hops between the Tor exit relays and the destination website (as shown in Figure 2) such that the destination website cannot reliably differentiate traffic from Tor and non-Tor users. We name such extra hops *exit bridges*.

Our high-level goal is to enable practical exit bridges that are accepted by Tor users as a viable solution to access websites that are otherwise unavailable due to server-side blocking. Specifically, we consider the following design goals:

Usability. The exit bridges should be compatible with existing Tor, to keep Tor users in a familiar and, more importantly, privacy-preserving environment. To ease deployment, exit bridges should not require changes to Tor protocols and should only require the installation of patched Tor Browsers that are configured to support their use. To achieve this, we implement the exit bridges as an optional service on top of Tor that can be accessed by installing an open-source Tor Browser extension. Our extension recognizes user-maintained Tor-blocking websites and switches to route Tor traffic through exit bridges when the user attempts to browse to these sites.

Safety and unlinkability. The use of Tor exit bridges should not compromise the user's anonymity or unlinkability. Connections between the user and bridge are always tunneled through Tor to ensure that the user's anonymity is protected. More detailed analyses of the security and anonymity properties of exit bridges are provided in §IV-B and §VI.

Ephemerality. The bridges should be ephemeral such that they are not easily enumerated; bridges with fixed IPs will likely eventually be exposed and adversaries can block such bridges just as they block Tor exit relays. Even if an individual exit bridge is detected and blocked, ephemerality ensures that the overall system is not significantly impacted. Each exit bridge is dedicated to serving only one Tor user, and, by design, it has a short life cycle—an exit bridge self-destructs after a predefined period.

High collateral damage. Additionally, the exit bridges should have a wide IP range, such that if the adversaries decide to block the whole range, as a means to block the ephemeral bridges, they will suffer significant collateral damage. Practically, the exit bridges should share IP space with services that frequently communicate with Tor-blocking sites. For example, we note that Amazon Workspaces [2], a service that provides cloud-hosted virtual desktops, uses IP addresses associated with AWS. Blocking AWS would thus prevent access from potential customers who use Amazon Workspaces. We provide further insight on the potential collateral damage due to blocking clients from cloud service providers in §V-D.

Low overhead. Exit bridges should not add significant overhead on top of Tor. Tor is designed as a low-latency anonymity network, and exit bridges should not incur latency and bandwidth penalties that impact its usability.

Scalability. The cost of running exit bridges should be minimal such that new bridges can be inexpensively spawned. The infrastructure should be sufficiently scalable and cost effective to support Tor's millions of daily users [27, 49].

IV. Implementation

Having presented the high-level design principles of the exit bridge infrastructure, we next describe our experiences implementing exit bridges (§IV-A), and discuss our solutions for addressing two particular challenges: (i) retaining unlinkability (§IV-B) and (ii) evaluating and covering the system's operating costs (§IV-C).

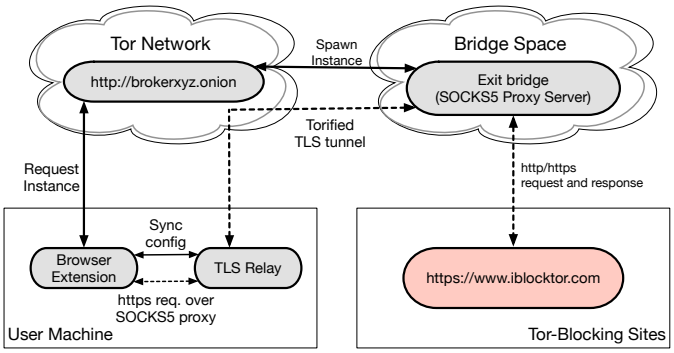


Fig. 3: A typical workflow of a Tor user visiting Tor-blocking websites through an exit bridge. Solid line indicates control flow and dotted line indicates data flow.

A. Exit Bridge Architecture

Tor exit bridges are created, maintained, and utilized for Tor users to visit web resources on demand. A typical workflow is depicted in Figure 3; a concrete demonstration is presented in §IV-D. The workflow follows the following steps:

1) Request exit bridge: As introduced in §III-C, a Tor Browser extension serves as the client's point of entry to using exit bridges. When a Tor user attempts to visit a URL, the Tor Browser extension intercepts each request and checks whether it is on the user-maintained server-side blocking list of known Tor-blocking websites. (This check occurs locally and the URL is never leaked.)

If the Tor user's intended destination is not on the blocking list, the Tor Browser will proceed as normal and no exit bridge will be used. Otherwise, the Tor Browser prompts the user that an exit bridge is needed for visiting the website, and redirects, if the user so chooses, to a *broker site* that is responsible for curating ephemeral exit bridges. The broker site is hosted within Tor as a hidden service, with the purpose of preserving the anonymity of the Tor user.

A user may use multiple exit bridges at the same time, where each exit bridge is dedicated to forwarding traffic only to a specific site for that user. That is, each exit bridge is specific to a single user and a single site the user is visiting. Additionally, we use a separate Tor circuit to connect to each exit bridge. This design prevents a rogue exit bridge from linking the user's activities across different sites. We discuss this design decision in more detail in §IV-B.

2) Spawn exit bridge: The main task of the broker site is to control the lifecycle of ephemeral exit bridges. As spawning and maintaining bridges on a cloud service provider incurs a financial cost, the broker may optionally ask Tor users to subsidize the service by contributing human work, for example, by completing an online image labeling task. This is advantageous not only for achieving the scalability design goal (by offsetting the cost of running the bridges), but also as a means to defend against naïve denial-of-service attacks. We provide more detailed cost and revenue analyses in §IV-C.

Once a user's contribution is confirmed, the broker then spawns an exit bridge. The exit bridge can be deployed on any IaaS cloud platform; in our current implementation, we chose AWS EC2 t3-nano instances for a concrete evaluation. To enforce ephemerality, the EC2 instance is configured to terminate after 15 minutes or after transmitting 50MB traffic, whichever comes first. If the user depletes the time or traffic budget, the user needs to contribute again to spawn another exit bridge.

Creating an exit bridge on Amazon EC2 takes approximately 50 seconds. Such a delay is likely intolerable to most web users, even if it enables access to an otherwise inaccessible website. To decrease this waiting time, the broker site adopts a *self-adaptive* buffering mechanism that maintains a pool of idle, never used exit bridges. When a user's request arrives, the broker associates it directly with one of the fresh exit bridges to minimize startup costs.

Suppose t_{start} is the time required to create an instance of an exit bridge on EC2, and t_{instance} is the average instance lifetime, that is, the time from when a user starts to use an exit bridge to the teardown of the exit bridge (either due to expiration of time or exhaustion of bandwidth budget). We further write N_{active} as the number of instances that are actively being used by users, and N_{spawn} as the number of instances that are currently being spawned.

Without the buffering mechanism, the ratio of N_{spawn} to N_{active} should roughly match the ratio of t_{start} to t_{instance}, assuming the user requests are flowing into the system at a relatively steady rate. In other words, $N_{\text{active}} \times \frac{t_{\text{start}}}{t_{\text{instance}}}$ users are waiting for the completion of spawning new exit bridges. Therefore, we use this to predict the number of idle exit bridges needed to accommodate all incoming requests. A benefit of this approach is its self-adaptiveness—the number of idle exit bridges flexibly scales up (or down) as the system receives an increasing (or decreasing) number of requests.

As the cost for optimizing away the waiting time for spawning new exit bridges, the buffering mechanism requires that $N_{\text{active}} \times \frac{t_{\text{start}}}{t_{\text{instance}}}$ exit bridges are kept idling, which generates additional operating cost. Considering the typical numbers of $t_{\text{start}} = 50$ seconds and $t_{\text{instance}} = 15$ minutes, the cost for the additional idling bridges is estimated as 5.5% of the total instance cost, which translates to $5,759 per year based on our cost estimates that we derive in the next section; we show there that this expense is easily covered by asking users to perform a small amount of work.

Once the exit bridge is ready to operate, it further goes through a number of initialization steps:

Create login credentials. The exit bridge is dedicated solely for the Tor user who requested the service. We use a random username/password pair as the login credentials.

Configure SSL/TLS certificate. When the Tor user connects through Tor to the exit bridge and presents its login credentials, the credentials should not be exposed to the Tor exit relay. This is enforced by protecting the communication between the Tor user and the exit bridge using TLS (wrapped within Tor's protocols). Each exit bridge uses a unique TLS certificate, signed by the broker.

After these initialization steps, the broker sends back to the Tor Browser extension the IP address and port of the bridge as

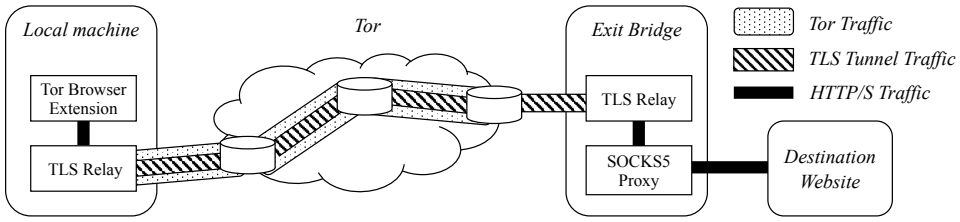


Fig. 4: Anonymous traffic is relayed via Tor with an exit bridge towards its intended destination.

well as the login credentials; this process is transparent to the actual human user. In short, after navigating to the broker site and electing to create an exit bridge instance, the process of initializing, configuring, and using the exit bridge is automated and handsfree.

3) Relay traffic through exit bridge: Once the Tor browser extension receives the IP, port number and login credentials of the exit bridge, it configures a local relay that encapsulates the user's traffic and tunnels it to the exit bridge through Tor.

Figure 4 depicts the end-to-end routing of the traffic from the Tor user to the destination website, which traverses over the following architectural components:

Local relay. A local relay accepts local TCP connections from the browser extension and decides which exit bridge to use based on the destination hostname. It then sends communication from the Tor Browser to the exit bridge, via the Tor-tunneled connection. Similarly, when it receives data from the exit bridge (via a Tor circuit), it forwards the data back to the Tor Browser.

Exit bridge. Each exit bridge consists of two modules. The first is a TLS endpoint, which operates similarly to the aforementioned local relay. Once a TLS connection has been established, it extracts the original HTTP/S traffic and forwards it to the second module, which can be any SOCKS5 proxy server (we use dante [5] in our implementation). The SOCKS5 proxy then relays traffic to the intended destination website. Responses from the destination website will be relayed back in the reverse direction.

The local relay and the exit bridge operate on the data plane, and form a completed torified TLS tunnel for securely transmitting HTTP/S requests and responses.

B. Per-destination Exit Bridges

Our design assigns dedicated exit bridges for each unique user. This potentially risks unlinkability, since a curious exit bridge operator (or an adversary that can observe traffic from the exit bridge) could associate traffic to multiple websites as originating from the same user.

This is more problematic for exit bridges than for exit relays, since the shared use of exit relays among many Tor users makes it more difficult for an eavesdropper to determine which exit streams belong to the same user.

To address this, we require that a user's traffic to different Tor-blocking second-level domains[2] be handled by different exit bridges. The Tor Browser extension effectively maintains a pool of exit bridges. Based on the destination of a web request, it chooses the corresponding exit bridge to route traffic. When it detects traffic to a new Tor-blocking site that has not been associated with any of the user's currently operating exit bridges, it sends a request to the broker to instantiate a new exit bridge. In this way, an exit bridge (or an eavesdropper) can only see the user's network activity to a single website. A cabal of curious exit bridges (or an eavesdropper that can observe traffic traversing multiple exit bridges) still cannot link web requests, since each exit bridge is accessed over independent Tor circuits; that is, it cannot discern whether web requests forwarded through two exit bridges belong to the same or different Tor users.

One potential concern is that multiple exit bridges are needed for browsing even just one website, if the site includes web objects from multiple second-level domains that block Tor. To determine how often this might happen in practice, we conducted a short experiment in which we configure the Tor Browser plugin to use exit bridges for all sites in the Alexa top 10,000 list that block Tor. We then visit 20 sites that block Tor, chosen uniformly at random from all such sites on the top 10,000 list. We find that 19 of the visits required just one exit bridge; the remaining site required the use of two exit bridges. In summary, we envision that users will require only one exit bridge to visit the vast majority of sites that block Tor.

C. Operating Cost

The operating cost of exit bridges is dominated by the instance cost and bandwidth usage. With the current pricing of AWS EC2 spot instances [1], a t3-nano instance with one vCPU and 1 GB memory, costing $0.0016 per hour, is sufficient to host an exit bridge. (All prices are in USD.) Amazon charges $0.05 per GB for outgoing traffic to the Internet (for monthly traffic over 100TB). Therefore, running an exit bridge for 15 mins and 50 MB data transmission costs under $0.0029.

Cost of running a global service. We further estimate the cost of providing an exit bridge service for *all* Tor users to freely visit *all* Tor-blocking websites. Hence, we provide conservative cost estimates in this section.

Is the "4.8%" in the 4th paragraph of the introduction supposed to be 6.4%, as 8%*80%?

*The 8% blocking rate reflects the percentage of *websites* that block traffic from Tor; however, it does not directly translate to the percentage of Tor *web visits* that suffer from server-side blocking (e.g., the more popular sites do not block Tor). Therefore, we estimated that 4.8% Tor traffic is*

[2]A second-level domain is a domain that is directly below the top-level domain, e.g., twitter.com.

blocked by adopting the following calculation: We assume that Tor users exhibit similar browsing behavior to general web users; that is, the distribution of website visits follows the power-law distribution. Based on this assumption, we estimate the percentage of website visits that experience server-side blocking of Tor, using the measurements we collected for the Alexa Top 10,000 websites. According to our fitted distribution, we calculated a block rate of 4.8% for Tor's web traffic.

We assume that Tor users exhibit similar browsing behavior to general web users [27]; that is, the distribution of website visits follows the power-law distribution [8]. Based on this assumption, we estimate the percentage of website visits that experience server-side blocking of Tor, using the measurements we collected for the Alexa Top 10,000 websites. Here, we fit the power-law distribution to the Alexa list since the latter specifies only the ranks of websites but not their respective popularities. According to our fitted distribution, we calculated a block rate of 4.8% for Tor's web traffic. For the month of May 2019, Tor's average daily measured bandwidth across all exit relays was 52.18 Gbps [49]. Given that AWS only charges for egress traffic to the Internet, the bandwidth subject to fees is calculated as 11.74 TB per day ($219,584 per year).

Estimating the instance cost is more challenging, since, to the best of our knowledge, there is no study measuring screen time on the Tor Browser. We instead take an indirect approach: we estimate the machine time based on the number of per-day Tor connections. Prior work has shown that the aggregated number of Tor connections per day is approximately 148 million [27]. Given that an esimated 4.8% of Tor traffic is blocked at the server side, We then estimate the number of connections blocked by target websites (and therefore benefitting from exit bridges) is 710K per day. We conservatively assume that each blocked connection needs the creation of a new 15-minute exit bridge, which yields $284 in machine-time cost per day, or $103,660 for a year.

As brokers only participate in spawning new exit bridges but not in the actual web communication, their operating cost is almost negligible. Taking the conservative estimation of 710K requests per day, we find that 250 AWS t3-nano instances are sufficient to sustain such a request rate, given that each request typically lasts under five minutes and communicates around 300 KB data (200 KB ingress and 100 KB egress). Therefore, the instance cost of running the global broker service is $3,285 per year, and the bandwidth cost is $1,236 per year.

The total yearly cost to allow *all* Tor users to visit *all* Tor-blocking sites is $328K.

Crowdsourcing the operating cost. In principle, the broker could recover its operating costs using traditional web monetization techniques—that is, by serving advertisements. However, since visitors arrive via Tor and are anonymous, it is both unclear whether ad networks would be willing to serve ads where they cannot identify users (e.g., via tracking cookies) and whether the traffic volume and click-through rates would be sufficient to recover the costs.

We instead consider revenue models that are better suited for anonymous browsing and do not require the identification of users. There is a nascent market of companies that present crowdsourced image labeling tasks to website visitors and provide some revenue to website operators.[3] For example, the hCaptcha service [23] serves a traditional CAPTCHA (e.g., labeling which animals are dogs or identifying dress sizes from photographs), with the results then being used as the ground-truth for some machine learning task by a third-party company. Unlike Google's popular reCAPTCHA service, hCaptcha pays the website operator for each completed task (approximately $0.0017 per task). In our experience, solving an hCaptcha puzzle takes between five and ten seconds.

Assuming 710K Tor connections are blocked per day (see the estimation of instance cost), the revenue collected from hCaptcha is $441K per year, which is enough to cover the running cost of the exit bridges (and, in fact, earn a small profit). In short, solving a single labeling task (requiring less than 10 seconds) provides sufficient funds for the infrastructure required to provide access to an otherwise inaccessible site.

D. Typical workflow

Finally we describe a typical workflow for using Tor exit bridges to access a website that otherwise blocks Tor. For illustrative purposes, we configure our Tor Browser extension to include http://whatsmyip.com in its server-side blocking list; that is, the Tor Browser extension considers the site to block Tor. (Choosing whatsmyip.com is intended to show that an exit bridge is indeed used to forward traffic, since the site lists the requesting IP. In actuality, whatsmyip.com does not block Tor traffic.)

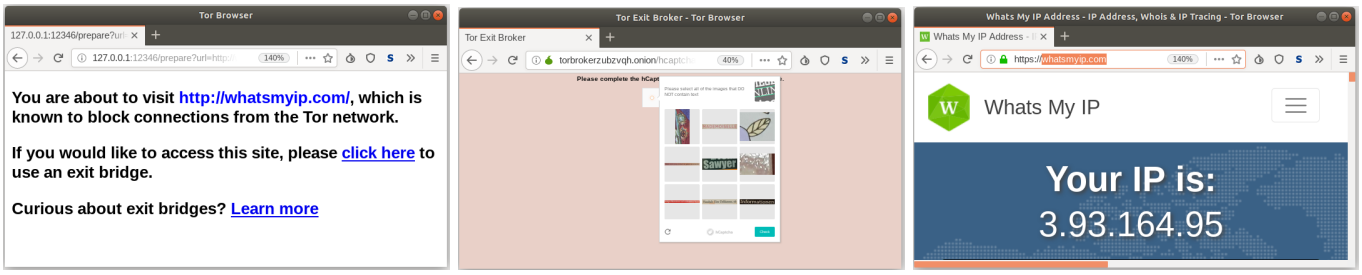
When the user attempts to browse to the site, the Tor Browser extension detects that it has been tagged as a site that blocks Tor. The Tor Browser will not load the webpage directly; instead, it offers the user the option of accessing the site through an exit bridge (Figure 5a). Should the user choose to, it will be redirected to the broker and asked to complete a CAPTCHA to instantiate an exit bridge (Figure 5b).

Once the CAPTCHA is completed, an exit bridge (an AWS EC2 instance in this case) will be instantiated and be configured to dedicatedly route traffic for the user to access http://whatsmyip.com. The instantiation and configuration process is transparent to the user. From the user's perspective, it will immediately be able to access the website (which would otherwise block Tor) through a Tor-circuit to the exit bridge and then to the target destination (Figure 5c). The IP address displayed on the webpage belongs to the exit bridge.

V. Evaluation

In this section, we focus on answering the following questions: (i) how much **collateral damage** is inflicted when site operators opt to block all requests originating from cloud service providers? (ii) what is the **effectiveness** of exit bridges in enabling Tor users to access previously blocked websites?

[3]This is very distinct from and should not be confused with unscrupulous sites that outsource the solving of CAPTCHAs (primarily to workers in inexpensive labor markets) in order to bypass site protections.



(a) (b) (c)

Fig. 5: (a) The Tor Browser extension detects that the user is attempting to access a site that blocks Tor, and offers the user the option of accessing the site through an exit bridge. (b) The exit bridge requires the user to complete a CAPTCHA to continue. (c) The site (that otherwise would block Tor) is accessed through the exit bridge. The IP address displayed in the screenshot is the (AWS-hosted) IP address of the exit bridge.

and (iii) how much **performance overhead** is introduced by exit bridges, in terms of the latency and total time spent for loading a website?

A. Experimental Setup

In our evaluation, we deployed the broker as a Tor hidden service. Exit bridges were hosted on t3-nano EC2 instances in Amazon's US East Region, configured as we described in §IV-A2. To enable automated evaluation on a large coverage of websites, we emulated the browsing of a destination website by calling PYCURL.PERFORM() with a Firefox USER-AGENT HTTP request header.

As a comparative study, we consider the following four different configurations:

- DIRECT serves as the baseline of the evaluation, where the client communicates with the destination website directly through the Internet.
- TOR represents that the client attempts to visit the destination website through Tor.
- In PROXYONLY, the client communicates with the destination website through the exit bridge only, but not through Tor. This configuration enables microbenchmarks to assess the overheads of exit bridges, but is not intended for real-world use.
- Finally, EXITBRIDGE is the complete deployment, where the client-website communication is relayed through both the Tor network and the exit bridge.

In our evaluation, as the destination websites, we used the top 1,000 websites in the Amazon Alexa's top one million site list [4]. For each website, we sent 10 consecutive requests using each configuration. As minor complications, we had to resolve issues caused by HTTP redirects and inconsistent DNS resolutions:

Handling HTTP redirections. We used Amazon Alexa's top 1,000 websites as the destination websites. Since this is a list of hostnames, a direct connection to http://hostname (or https://hostname) may be redirected, which can introduce unpredictable extra traffic and unexpected HTML responses. To address this, we generated a list of redirected URLs from the original hostname list before the evaluation: we directly connected to http://hostname (or https://hostname) and then noted any redirections.

Fixing DNS resolution. For many websites, a single hostname could resolve to multiple IP addresses. This can occur when reverse proxies, load balanced DNS, or (most commonly) content distribution networks (CDNs) are used. To ensure that we communicated to the same IP address across all four configurations, we performed DNS resolution before the evaluation and ensured that all subsequent experiments would use consistent IP addresses.

For each request, if a server response is received within 10 seconds, we save the response as well as pcap packet traces captured from the local machine and the exit bridge. (We emphasize that we record only our own traffic.) Note that we fetch the base HTML only and do not retrieve other web objects or execute embedded Javascript. The collected HTML and pcap files are used to analyze the HTML similarity between different configurations, and the latency incurred when accessing the destination websites.

When emulating the browsing of destination websites, has the distributed clients' location been considered?

In our evaluation, we fixed the location of the client. However, differing the clients' locations would have negligible impact on our evaluation results: in terms of the effectiveness of the exit bridges, whether a web request encounters server-side blocking is influenced almost exclusively by the IP or geolocation of the exit bridge that does the "last-hop" (instead of the client); the performance (i.e., latency) is dominated by the communication within the Tor network, the geolocation of the client contributes little in terms of the overall time-to-first-byte (or time-to-last-byte) latency.

Our results are obtained using a fixed client location. However, the location of the client should have negligible impact on our evaluation results since the client's network location is protected by Tor—and not available to the destination website—and thus does not affect server-side blocking of Tor traffic. In terms of the effectiveness of the exit bridges, whether a web request encounters server-side blocking is influenced almost exclusively by the IP or geolocation of the *last hop* in the circuit, i.e., the exit relay or exit bridge.

B. Effectiveness of Exit Bridges

By using exit bridges, Tor users should be able to access websites that block Tor traffic, and the responses received

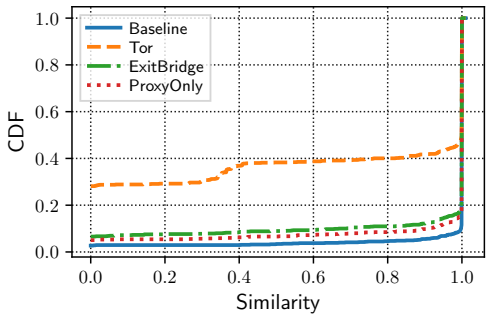


Fig. 6: CDF of HTML similarity scores.

should be the same or similar to those when the pages are fetched directly without Tor. To verify this, we calculate the HTML similarity [28] of DIRECT, TOR, PROXYONLY, and EXITBRIDGE (see §V-A) against pages fetched directly. Both this baseline and DIRECT use direct IP to fetch the webpages; we include DIRECT here to show how HTML similarity naturally shifts due to dynamic content.

Figure 6 plots the CDF of the similarity scores. A similarity score of 1 indicates an exact match and a score of 0 indicates little or no commonalities. As expected, DIRECT consistently has the highest HTML similarity: over 90% of the fetched websites are the exact match of their references. A significant portion of requests in the TOR configuration yield a similarity score of zero, meaning Tor traffic is blocked by these websites. We note that this block rate is inflated by including the websites hosted on Cloudflare—these websites require a CAPTCHA challenge for accesses from `pycurl` but would otherwise have rendered correct content on the Tor Browser [40]. Even excluding these Cloudflare-hosted websites, the accurate block rate of Tor is still well above 13.8% (the rate decreases to around 8.97% for the Alexa top 10,000 websites).

The request through Tor may be accomplished based on the location of the exit relay, resulting in the different page content and low similarity. Can such a case be effectively eliminated by the threshold of similarity or it is rare in reality? Please clarify.

*We did observe that the same URL results in different renderings from different geographic locations due to dynamic contents customized to different locales or users. To ensure that our evaluation results do not mistakenly count these cases as server-side blocking, we adopt an extremely conservative threshold (to be exact, an HTML similarity score of 0.01 or below). This threshold counts a server-side blocking only if the retrieved page is *almost completely* different from the reference webpage retrieved without using Tor.*

On the other hand, the curves for PROXYONLY and EXITBRIDGE track that of DIRECT closely, offering significantly improved access to the Tor-blocking websites. The relatively lower similarity scores (compared to DIRECT) are mainly due to the use of the exit bridge's geographic location, which differs from that of the machine in the DIRECT configuration, leading to differences in sites that customize content based on clients' perceived geographic locations. EXITBRIDGE receives marginally lower similarity scores compared to PROXYONLY. This is because, in rare cases, the use of Tor caused a significantly prolonged retrieval time that caused the request to time out.

C. Performance Overhead

To evaluate the performance overhead of exit bridges, we measure both the time-to-first-byte and time-to-last-byte latency for each "browsing action". We present the analysis results of time-to-last-byte; similar observations are made in the time-to-first-byte analysis below. Time-to-last-byte measures the time elapsed for the `pycurl.perform()` function call to complete—that is, for the entire page to be fetched. Time-to-last-byte additionally reveals the transmission goodput.

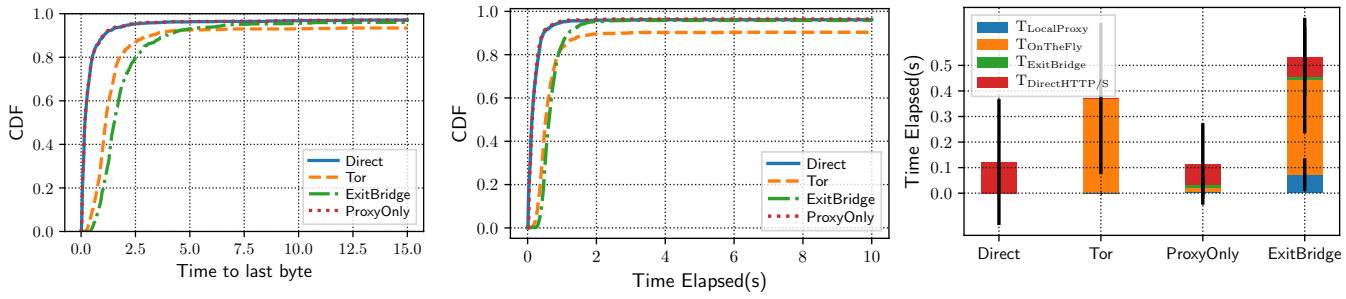
Figure 7a shows the CDF of the time-to-last-byte latency. We observe that PROXYONLY incurred negligible latency compared to DIRECT for web requests. Similarly, the comparison between EXITBRIDGE and TOR shows that the latency introduced by the extra hop to the exit bridge was insignificant and would not likely noticeably affect users' browsing experience. (This is mostly unsurprising since Tor is known to incur a large latency penalty [13, 49], which overwhelms the cost of including an additional hop for the exit bridge.) We further break down the latency to reveal the time spent in each step of the communication and our study confirms that Tor is the main contributing factor to the end-to-end latency. We also observe that the CDF of TOR plateaus at approximately 92%. This is mainly because some Tor-blocking websites did not return responses or returned error codes.

The cumulative distribution of the time-to-first-byte latencies is shown in Figure 7b. The Figure shows a similar trend as the time-to-last-byte measurements (presented in §V-C): the performance of EXITBRIDGE is close to that of TOR.

To confirm our conjecture that Tor is the main contributing factor to the end-to-end latency, we further break down the time-to-first-byte latency to reveal the time spent in each step of the communication. This is achieved by analyzing the pcap files collected at the local relay and the exit bridge (again, we capture only our own traffic). We consider the following four contributing factors:

- $T_{\text{LOCALPROXY}}$: the time required to relay traffic through the local relay;
- T_{ONTHEFLY}: the time required for traffic to traverse either via Tor (in either the TOR or EXITBRIDGE configuration) or direct IP communication (in the case of PROXYONLY). In the case of TOR, T_{ONTHEFLY} also includes the latency of direct HTTP/S requests/responses;
- $T_{\text{EXITBRIDGE}}$: the time required to relay traffic through an exit bridge; and
- $T_{\text{DIRECTHTTP/S}}$: the latency of direct HTTP requests/responses.

Figure 7c shows the breakdown of the time-to-first-byte latency. We observe that the latency is dominated by the traversal through the Tor network (T_{ONTHEFLY}). The performance



(a) (b) (c)

Fig. 7: Relative performance of Exit Bridges, including *(a)* the CDF of time-to-last-byte (in seconds); *(b)* the CDF of the time-to-first-byte latency (in seconds); and *(c)* contributing factors of the time-to-first-byte latency.

overheads added by the exit bridge (i.e., $T_{LocalProxy}$ and $T_{ExitBridge}$) are insignificant in comparison.

In summary, exit bridges enable Tor users to access most Tor-blocking sites, with similar overall performance as Tor.

D. Assessing Collateral Damage

A website operator may block traffic from exit bridges by blocking access from the cloud service providers that host the bridges. We consider the collateral costs of such blocking. The degree of collateral damage is dependent on the regularity at which non-Tor traffic to the website originates from these cloud service providers. Our findings suggest that websites may already see a surprisingly large fraction of requests that originate from the cloud, leading to significant collateral damage if they are blocked.

Amazon provides a managed, cloud desktop service called Amazon Workspaces [2]. We studied one instance of Amazon Workspaces by setting up our own cloud desktop and found its IP to be in the IP range of AWS [3]. This implies that blocking AWS will also harm the users of AWS-based virtual desktops. Similar services are also offered by Google Cloud [6] and Microsoft Azure [7].

We also examined a three-day snapshot of our institution's web logs to see if any requests originated from IPs belonging to AWS [3], Google Cloud, and Microsoft Azure [29]. We exclude all requests for robots.txt or those containing the (case-insensitive) substrings bot, crawler, spider, indexer or b-o-t in the USER-AGENT header. Among the 5.7M total requests seen by our institution's web server, we surprisingly find that 5.6% of client requests originated from IPs residing in AWS, with smaller amounts coming from Google Cloud and Microsoft Azure. More than two-thirds of these requests came from users using Chrome, Internet Explorer/Edge, Firefox, or Opera (based on the supplied USER-AGENT), suggesting that most of these cloud-based requests resulted from actual browsing activity. We posit that losing more than 5% of potential site requests for the purposes of preventing Tor users from accessing the site is too high a cost for the website operator. In addition, with the trend of increasingly more companies migrating their IT needs to cloud-based solutions, it will likely become even more costly to blindly block all accesses from the cloud.

VI. Anonymity and Security Considerations

We discuss the privacy and security implications of our exit bridge infrastructure, organized by the participating parties.

Exit relays. Exit bridges receive traffic from Tor circuits that terminate at exit relays (see Figure 3). Our design adds another hop to the anonymous path, reducing the role of an exit relay from an egress point to effectively a second middle relay. The TLS Relays that reside on the client and the exit bridge encrypt all traffic between them. This prevents the exit relay from learning the client's requested destination.

The exit relay could perform traffic analysis (e.g., throughput [30] or website fingerprinting [9, 38, 52]), just as any other middle relay. We consider such attacks orthogonal since they equally apply to Tor's current design.

Exit bridges. As with traditional exits, an exit bridge learns the client's destination, but not the identity of the client.

The risks of using exit bridges are similar to but distinct from those of using meek bridges. A meek bridge allows the cloud provider to observe ingress traffic and enumerate the users (or their IPs) of the meek bridge. This is arguably a greater anonymity threat than, in the case of exit bridges, the cloud provider learning the requested destinations. Although we use AWS in our deployment, multiple independently operated cloud providers can be used to host distinct exit bridge networks, offering some decentralization. However, such providers are limited, and funneling traffic through the cloud is a fundamental feature (and anonymity risk) of our design.

To maintain Tor's unlinkability, an exit bridge is dedicated for a single user visiting a single website (see §IV-B). Thus, a malicious exit bridge cannot link multiple sites visited by a user. Similarly, colluding exit bridges cannot reliably link traffic as belonging to the same user since each connection to a bridge uses an independent Tor circuit.

Finally, to prevent trivial traffic correlation attacks [25, 33, 37], the Tor Browser should ensure that the ingress point (e.g., a guard or bridge) is not hosted by the same provider as the exit bridge. (Popular cloud providers publish their IP address ranges.) We are planning on adding such checks to our implementation.

Broker. The broker operates as a hidden service, and all connections to the broker occur over independent Tor circuits. This prevents even a malicious broker from learning the network locations of the clients, or linking two requests as belonging to the same client. The broker assigns clients to exit bridges—forcing users to use a malicious exit bridge is equivalent to users selecting a malicious exit; this case is covered above.

To prevent tracking by the broker using client side cookies, the Tor Browser extension should prevent or delete cookies from the broker site (this feature is currently under development). We additionally rely on the extensive anti-fingerprinting techniques provided by the Tor Browser [39] to prevent other methods of breaking unlinkability.

VII. Ethics

We consider the ethical considerations of this paper from two dimensions: the ethics of our experiments and the ethics of a future Tor deployment of exit bridges.

Ethics of experimentation. We believe the experiments described in §V are well within the bounds of ethical and responsible research. As a guiding ethical framework, we consider the Menlo Report [14], an extension of the Belmont Report [34] for ethical research that is specifically tailored for computer security research. The Menlo Report describes four main principles of ethical studies: respect for persons, beneficence, justice, and respect for law and public interest. Since our experiments do not concern human users or data derived from human users (and thus is not covered by our institutional review board), they trivially achieve the first and third criteria. To the best of our knowledge, our experiments pose no significant risks and do not violate any laws (at least not in our jurisdiction); in general, we simply retrieved publicly accessible webpages via AWS. Notably, the effects of our experiments (visiting webpages) are identical to what would have occurred had we retrieved the webpages using virtual AWS Workspaces. In summary, we believe that our experiments meet the ethical criteria of the Menlo Report and additionally fall well within the norms of computer science research.

Ethics of an exit bridge deployment. We separately consider the ethics of a publicly accessible exit bridge deployment. As with many privacy preserving and censorship evading technologies, the ethics of such a system are multifaceted and complex. In the remainder of this section, we attempt to highlight some of the major ethical issues involved in allowing users to bypass site-based blocking of Tor.

Are we circumventing security? There are many reasons sites might block access from Tor, including fear of malicious traffic relayed through the anonymity network. By design, exit bridges conceal that the traffic traversed through Tor, and thus could permit malicious traffic to reach a destination which it otherwise could not have. However, there are myriad other ways in which attackers already can disguise the origins of attack traffic, including the use of open proxies, VPNs, and botnets. We are skeptical that preventing access from Tor provides websites with much security, but it is important to acknowledge that exit bridges do bypass such protections.

We note that the CAPTCHA puzzle-solving requirement provides some mitigation against automated activities (for example, spamming and crawling) that often irk site operators. That is, the (human) work required to connect via an exit relay—while not especially burdensome to an individual user—makes it more difficult (albeit not impossible) for an an attacker to use the exit bridge infrastructure to do automated activities.

If sites purposefully block access to Tor for philosophical reasons, is it appropriate to permit such access? This is the reciprocal to "if a country disallows access to Tor, is it ethical to provide access (e.g., through bridges) anyways?" Both traditional bridges and exit bridges purposefully violate policies, just at opposite ends of the communication. There is also some similarity to the case of ad and web tracker blockers, which can violate sites' acceptable use policies but which protect users' privacy.

The pertinent section of the Menlo Report [14, see §C.5] acknowledges that public interest (here, allowing users to freely and privately access public websites) may conflict with acceptable use and other policies.[4] The Report requires in such cases that there be "ethically defensible justification" [14, see §C.5.1], which is admittedly a very subjective criteria. In brief, we believe that allowing users to browse privately is of such immense public interest, that it justifies the use of exit bridges.

Are we imposing a burden on the cloud service provider if Tor users use exit bridges to perform illegal actions? Tor is used by criminals to access illegal content (e.g., child abuse imagery) and perform other illegal actions [36]. As happens with exit relays, the illegal activity could be misattributed to the exit bridge—and thus the cloud service provider—since traffic appears to be originating from the provider. From the (U.S.-centric) legal perspective, the cloud service provider has little legal liability. In particular, the Digital Millennium Copyright Act (DMCA) provides indemnification (i.e., so-called "safe harbor" protections) for entities that act solely as a "conduit" for forwarding traffic [50]. The Tor Project does not know of any individuals being sued or prosecuted for running exit relays [19], and the same would likely apply to exit bridges.

A potential mitigation that reduces the burden and exposure of the cloud service provider is to use a whitelisting strategy in which exit bridges are configured to only create connections to sites that block Tor. This would likely still permit access to blocked sites while substantially reducing or even eliminating abuse complaints. And, as mentioned above, the use of CAPTCHAs may stymie automated activities (e.g., crawling), potentially reducing the number of complaints sent to the cloud provider.

[4]The Menlo Report focuses on the ethics of computer security research. While the above paragraph considers the ethics of deploying exit bridges (and not on performing research), we believe the principles laid in the report are applicable here too.

VIII. Limitations

Exit bridges aim to circumvent server-side discrimination against Tor traffic by making it more difficult to determine that a given traffic flow traversed through the Tor network. However, services can still prevent the use of exit bridges by blocking all traffic that originates from cloud service providers. As discussed above, doing so could also block (i) users who either proxy or originate their traffic from cloud service providers and (ii) automated systems (e.g., web crawlers) that operate from the cloud service provider.

Rather than identify exit bridge traffic by IP, a site could attempt to detect the use of the Tor Browser via browser fingerprinting [18]. The Tor Browser is not intended to be a covert application. By design, it attempts to reduce the entropy of an individual user's browser fingerprint [32, 39] by making all Tor Browsers look identical—but not identical to other browsers. A determined website administrator can likely identify the Tor Browser with high accuracy. However, this requires the administrator to (i) include Javascript that performs measurements of the browser and (ii) add website logic to assess the measurements and produce a browser fingerprint. This requires making webpages larger and slower to load. We are skeptical that website operators would be willing to accept such tradeoffs.

Lastly, it is unclear that website operators would even want to disallow Tor Browser users if their goal is to eliminate attacks arriving from Tor. Attackers who use Tor to perform vulnerability scanning or send malicious content likely do not use the Tor Browser and instead attach their scripts and tools directly to the Tor client (e.g., via torsocks). We posit that Tor users who use the Tor Browser are much more likely to be non-malicious.

IX. Conclusion

This paper highlights the growing threat of server-side blocking of Tor traffic, and introduces the exit bridge architecture to counter such censorship efforts. Ephemeral exit bridges are difficult to block since they are short-lived and their network locations are largely indistinguishable from other cloud-based services (including those used by website users who do not use Tor). Our experiments show that exit bridges effectively circumvent server-side efforts to block Tor with a very modest 14.5% increase in latency relative to normal Tor usage. Additionally, based on current cloud service provider pricing models, exit bridges are inexpensive and cost under $0.01 per bridge per hour. Our implementation is available as free open-source software and can be downloaded at https://security.cs.georgetown.edu/tor-exit-bridges.

Acknowledgments

We thank the anonymous reviewers and our shepherd Yennun Huang for their valuable comments. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contracts Nos. FA8750-19-C-0500 and HR0011-16-C-0056 and the National Science Foundation (NSF) under grants CNS-1453392, CNS-1513734, CNS-1527401, CNS-1704189, and CNS-1718498. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA or NSF.

References

[1] Amazon EC2 Pricing, 2019. Available at https://aws.amazon.com/ec2/pricing/on-demand/.

[2] Amazon Workspaces, 2019. Available at https://aws.amazon.com/workspaces/.

[3] AWS IP Address Ranges, 2019. Available at https://ip-ranges.amazonaws.com/ip-ranges.json.

[4] AWS Top 1M Sites List, 2019. Available at http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[5] Dante Socks5 Server Implementation, 2019. Available at http://www.inet.no/dante/.

[6] Google Cloud solutions, 2019. Available at https://cloud.google.com/solutions/.

[7] Windows virtual desktop, 2019. Available at https://azure.microsoft.com/en-us/services/virtual-desktop/.

[8] Adamic, L. A., and Huberman, B. A. Zipf's Law and the Internet. *Glottometrics 3*, 1 (2002), 143–150.

[9] Cai, X., Zhang, X. C., Joshi, B., and Johnson, R. Touching from a Distance: Website Fingerprinting Attacks and Defenses. In *ACM Conference on Computer and Communications Security (CCS)* (2012).

[10] Davidson, A., Goldberg, I., Sullivan, N., Tankersley, G., and Valsorda, F. Privacy pass: Bypassing internet challenges anonymously. *Proceedings on Privacy Enhancing Technologies 2018*, 3 (2018), 164–180.

[11] Dingledine, R. Research Problems: Ten Ways to Discover Tor Bridges (Tor Blog Post), 2011. Available at https://blog.torproject.org/research-problems-ten-ways-discover-tor-bridges.

[12] Dingledine, R., Mathewson, N., and Syverson, P. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium (USENIX)* (August 2004).

[13] Dingledine, R., and Murdoch, S. Performance Improvements on Tor, or, Why Tor is Slow and What We're Going to Do About It. https://svn.torproject.org/svn/projects/roadmaps/2009-03-11-performance.pdf, March 2009.

[14] Dittrich, D., Kenneally, E., et al. The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research. Tech. rep., US Department of Homeland Security, August 2012.

[15] Durumeric, Z., Wustrow, E., and Halderman, J. A. ZMap: Fast Internet-wide Scanning and its Security Applications. In *USENIX Security Symposium (USENIX)* (2013).

[16] Dyer, K. P., Coull, S. E., Ristenpart, T., and Shrimpton, T. Protocol Misidentification Made Easy with Format-Transforming Encryption. In *ACM Conference on Computer and Communications Security (CCS)* (2013).

[17] DYER, K. P., COULL, S. E., AND SHRIMPTON, T. Marionette: A Programmable Network Traffic Obfuscation System. In *USENIX Security Symposium (USENIX)* (Aug. 2015).

[18] ECKERSLEY, P. How Unique is Your Browser? In *Privacy Enhancing Technologies Symposium (PETS)* (2010).

[19] ELECTRONIC FRONTIER FOUNDATION (EFF). The Legal FAQ for Tor Relay Operators, 2014. Available at https://www.torproject.org/eff/tor-legal-faq.html.en.

[20] FIFIELD, D. meek. https://trac.torproject.org/projects/tor/wiki/doc/meek.

[21] FIFIELD, D., LAN, C., HYNES, R., WEGMANN, P., AND PAXSON, V. Blocking-resistant Communication through Domain Fronting. In *Privacy Enhancing Technologies Symposium (PETS)* (2015).

[22] GOWDA, T., AND MATTMANN, C. A. Clustering Web Pages Based on Structure and Style Similarity. In *International Conference on Information Reuse and Integration (IRI)* (2016).

[23] HCAPTCHA. https://hcaptcha.com/.

[24] HOUMANSADR, A., BRUBAKER, C., AND SHMATIKOV, V. The Parrot is Dead: Observing Unobservable Network Communications. In *IEEE Symposium on Security and Privacy (Oakland)* (2013).

[25] JOHNSON, A., WACEK, C., JANSEN, R., SHERR, M., AND SYVERSON, P. Users Get Routed: Traffic Correlation on Tor By Realistic Adversaries. In *ACM Conference on Computer and Communications Security (CCS)* (November 2013).

[26] KHATTAK, S., FIFIELD, D., AFROZ, S., JAVED, M., SUNDARESAN, S., MCCOY, D., PAXSON, V., AND MURDOCH, S. J. Do You See What I See? Differential Treatment of Anonymous Users. In *Network and Distributed System Security Symposium (NDSS)* (2016).

[27] MANI, A., BROWN, T. W., JANSEN, R., JOHNSON, A., AND SHERR, M. Understanding Tor Usage with Privacy-Preserving Measurement. In *ACM SIGCOMM Conference on Internet Measurement (IMC)* (October 2018).

[28] MARCA, E. HTML Similarity Tool, 2018. Available at https://github.com/matiskay/html-similarity.

[29] MICROSOFT. Microsoft Azure Datacenter IP Ranges. Available at https://www.microsoft.com/en-us/download/details.aspx?id=41653.

[30] MITTAL, P., KHURSHID, A., JUEN, J., CAESAR, M., AND BORISOV, N. Stealthy Traffic Analysis of Low-latency Anonymous Communication using Throughput Fingerprinting. In *ACM Conference on Computer and Communications Security (CCS)* (2011).

[31] MOGHADDAM, H. M., LI, B., DERAKHSHANI, M., AND GOLDBERG, I. SkypeMorph: Protocol Obfuscation for Tor Bridges. In *ACM Conference on Computer and Communications Security (CCS)* (2012).

[32] MOWERY, K., AND SHACHAM, H. Pixel Perfect: Fingerprinting Canvas in HTML5. In *Web 2.0 Security & Privacy* (2012).

[33] MURDOCH, S. J., AND DANEZIS, G. Low-Cost Traffic Analysis of Tor. In *IEEE Symposium on Security and Privacy (Oakland)* (2005).

[34] NATIONAL COMMISSION FOR THE PROPTECTION OF HUMAN SUBJECTS OF BIOMEDICAL AND BEHAVIORAL RESEARCH. *The Belmont Report: Ethical Principles and Guidelines for the Protection of Human Subjects of Research*. U.S. Government Printing Office, 1978.

[35] NOTTINGHAM, M., MCMANUS, P., AND RESCHKE, J. HTTP Alternative Services. Tech. Rep. 7838, Internet Engineering Task Force, April 2016.

[36] O'NEILL, P. H. Tor's Ex-director: 'The Criminal Use of Tor has Become Overwhelming'. In *Cyberscoop (online news article)* (May 2017). Available at https://www.cyberscoop.com/tor-dark-web-andrew-lewman-securedrop/.

[37] ØVERLIER, L., AND SYVERSON, P. Locating Hidden Servers. In *IEEE Symposium on Security and Privacy (Oakland)* (2006).

[38] PANCHENKO, A., NIESSEN, L., ZINNEN, A., AND ENGEL, T. Website Fingerprinting in Onion Routing Based Anonymization Networks. In *ACM Workshop on Privacy in the Electronic Society (WPES)* (2011).

[39] PERRY, M., CLARK, E., MURDOCH, S., AND KOPPEN, G. The Design and Implementation of the Tor Browser [DRAFT], 2018. Available at https://2019.www.torproject.org/projects/torbrowser/design/#fingerprinting-linkability.

[40] SAYRAFI, M. Introducing the Cloudflare Onion Service (Blog post), September 2018. Available at https://blog.cloudflare.com/cloudflare-onion-service/.

[41] SINGH, R., NITHYANAND, R., AFROZ, S., PEARCE, P., TSCHANTZ, M. C., GILL, P., AND PAXSON, V. Characterizing the nature and dynamics of tor exit blocking. In *26th USENIX Security Symposium (USENIX Security). USENIX Association, Vancouver, BC* (2017).

[42] SUN, Y., EDMUNDSON, A., VANBEVER, L., LI, O., REXFORD, J., CHIANG, M., AND MITTAL, P. RAPTOR: Routing Attacks on Privacy in Tor. In *USENIX Security Symposium (USENIX)* (Aug. 2015).

[43] TAN, H., SHERR, M., AND ZHOU, W. Data-plane Defenses against Routing Attacks on Tor. In *Privacy Enhancing Technologies Symposium (PETS)* (July 2016), vol. 4, pp. 276–293.

[44] THE SELENIUM PROJECT. The Selenium Project, 2018. Available at https://www.seleniumhq.org/.

[45] THE TOR PROJECT. Meek Pluggable Transport Overview. https://trac.torproject.org/projects/tor/wiki/doc/meek#Overview.

[46] THE TOR PROJECT. obfs4 (The obfourscator). Available at https://gitweb.torproject.org/pluggable-transports/obfs4.git/tree/doc/obfs4-spec.txt.

[47] THE TOR PROJECT. List of Pluggable Transports, 2018. Available at https://trac.torproject.org/projects/tor/wiki/doc/PluggableTransports/list.

[48] THE TOR PROJECT. Tor: Pluggable Transports,

2018. Available at https://www.torproject.org/docs/pluggable-transports.html.

[49] TOR PROJECT, INC. Tor Metrics Portal. https://metrics.torproject.org/.

[50] U.S. GOVERNMENT. Digital Millennium Copyright Act (DMCA). U.S. Code Title 17, Chapter 5, §512. Available at https://www.law.cornell.edu/uscode/text/17/512.

[51] WANG, L., DYER, K. P., AKELLA, A., RISTENPART, T., AND SHRIMPTON, T. Seeing through Network-Protocol Obfuscation. In *ACM Conference on Computer and Communications Security (CCS)* (2015).

[52] WANG, T., CAI, X., NITHYANAND, R., JOHNSON, R., AND GOLDBERG, I. Effective Attacks and Provable Defenses for Website Fingerprinting. In *USENIX Security Symposium (USENIX)* (2014).

[53] WEBFP. A Python library to automate Tor Browser with Selenium, 2018. Available at https://github.com/webfp/tor-browser-selenium.

[54] WEINBERG, Z., WANG, J., YEGNESWARAN, V., BRIESEMEISTER, L., CHEUNG, S., WANG, F., AND BONEH, D. StegoTorus: A Camouflage Proxy for the Tor Anonymity System. In *ACM Conference on Computer and Communications Security (CCS)* (2012).

[55] WINTER, P., AND LINDSKOG, S. How the Great Firewall of China is Blocking Tor. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)* (2012).

[56] WINTER, P., PULLS, T., AND FUSS, J. ScrambleSuit: A Polymorphic Network Protocol to Circumvent Censorship. In *ACM Workshop on Privacy in the Electronic Society (WPES)* (2013).