

Improving the Performance and Security of Tor's Onion Services

Arushi Arora
Purdue University
arora105@purdue.edu

Christina Garman
Purdue University
clg@cs.purdue.edu

Abstract

Tor is one of the most widely used anonymous communication networks today. A popular feature of Tor is its onion services, anonymous network services that can only be accessed via the Tor network. This enables users to both host and access such services anonymously, protecting onion services from censorship and take-down. According to Tor Metrics, over 150,000 onion services collectively serve traffic at a rate of nearly 4 Gbps, with applications ranging from news services to chat to whistleblowing. Unfortunately, onion services also suffer from a variety of performance and security concerns. Latency can be extremely high, and many services face denial of service and deanonymization attacks due to the content and types of services that they host.

In this work we seek to help address these concerns *without making any changes to Tor*, thus making our improvements immediately useful and deployable. To do this, we leverage a recent advance in programmable anonymity networks, which allows one to deploy user-written functions on willing Tor relays. We use this architecture to design the first Content Delivery Network (CDN) for onion services, which we call *Centor*. *Centor* allows onion services to take advantage of many traditional CDN benefits, such as replication and load balancing and bringing content (geographically) closer to the client. These techniques and applications raise an interesting trade-off between performance and anonymity for users, which we rigorously explore and quantify. We implement, deploy, and evaluate our architecture on the Tor network, demonstrating how these techniques are immediately able to extend and improve the capabilities, performance, and defenses of onion services, without any changes to the Tor protocol.

Keywords

Tor, onion services, CDN

1 Introduction

Tor [29] is one of the most popular and widely used anonymous communication networks today. Tor enables anonymous communication by employing a volunteer overlay network to conceal a user's location and usage from anyone conducting network surveillance or traffic analysis. In addition, Tor's onion services offer an anonymous network service, permitting a web server to obfuscate its location. This makes it possible for Tor users to both host content and access these onion services anonymously. Onion services have supported freedom of speech, expression, and information to users living in oppressive regimes and have therefore experienced a

sharp increase in their number and amount of traffic relayed. In fact, according to Tor Metrics, over 150,000 onion services collectively serve traffic at a rate of nearly 4 Gbps, with applications ranging from news services to chat to whistleblowing [73].


One of the largest performance barriers that Tor faces in terms of practical client usage is that of latency [2, 6, 9, 39]. Interactive communication on Tor, which accounts for over 90% of connections in the Tor network, incurs latencies over 5x greater than on a direct (traditional) Internet path [61]. The delay factor further increases when a client attempts to reach an onion service, as then a Tor circuit is utilized on both the client- and server-side. For instance, [17] shows that three-hop circuits can provide acceptable performance for VoIP calls. However, when extending to six-hop circuits, the latency increases significantly. The average one-way delay in six-hop configurations often exceeds the acceptable threshold due to the additional relay nodes, which introduce more latency and degrade overall call quality. Similarly, [82] shows that a single Tor link cannot consistently provide the stable and low latency required for high-quality VoIP, supporting multiplexing traffic over circuits. This increased latency can discourage Tor users from adopting and utilizing onion services, both as a service operator and a client [33, 98].

From a security perspective, the Tor network, which uses a set of volunteer relays, is also prone to Denial of Service (DoS) attacks. For example, Jansen et al. [49] estimated that congestion attacks against all Tor onion routers could increase the median client download time by 47%. Due to the asymmetrical architecture of the Tor protocol, these DoS attacks can further result in the unavailability of an onion service under attack. Furthermore, the anonymity of an onion service's clients makes it harder to detect malicious ones, preventing the deployment of typical defenses that one might employ on the non-anonymous Internet.

Additionally, onion services also face many threats of deanonymization [55]. Attacks on an onion service, such as timing analysis, service location attacks, and distance attacks, can potentially expose the location of the server hosting it [65]. Circuit fingerprinting attacks [57] attempt to correlate the circuits involved in communicating with an onion service and then perform a website fingerprinting attack [96] on the identified circuits to deanonymize the target service with high accuracy.

Based on these considerations, we postulate that not only would providing onion service operators with easy-to-deploy means of protecting themselves against common attacks as well as realize performance improvements enrich the user experience, but they would also encourage more content hosts and distributors to use and own onion services. This would then in turn allow more users to reap the security and privacy benefits of onion services. We, therefore, seek to answer the following questions:

What performance improvements can an onion service client securely obtain today? How can onion services both operate with a reduced

This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license visit <https://creativecommons.org/licenses/by/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Proceedings on Privacy Enhancing Technologies 2025(1), 531–552
© 2025 Copyright held by the owner/author(s).
<https://doi.org/10.56553/popets-2025-0029>

risk of DoS and deanonymization attacks and provide a better experience for their users? Can we obtain these benefits in a way that is immediately deployable, without altering the Tor protocol?

We answer these questions by providing two immediately deployable security and performance improvements for onion services. We design and build **CenTor**, the first Content Delivery Network (CDN) for Tor onion services, providing them with enhanced resilience and security, as well as improved performance and reduced latency to clients through techniques for geographical-awareness. While the use of CDNs are quite common on the traditional Internet, and are a key component in rendering DoS attacks no longer an existential threat to Internet services, to date, anonymity networks (and onion services) have not been able to reap these same benefits. We also build μ Tor, an adaption of techniques from [3, 101, 102] for dynamic traffic splitting and multipath routing to onion services, that allows us to distribute the traffic load across multiple circuits, thus achieving better bandwidth for onion services and better utilization of often-overlooked low-bandwidth Tor relays.

The key insight that allows us to build **CenTor** and μ Tor without needing to change the Tor protocol in any way, is the use of Bento, a recently introduced architecture for achieving the features of programmable networks in Tor [77]. Bento is a Network Function Virtualization (NFV) service that allows Tor relays to act as programmable middleboxes. Users can write sophisticated functions in a high-level language that the Bento server can execute on their behalf. Furthermore, the usage of conclave [38] provides executing functions with strong confidentiality and anonymity guarantees. We implement **CenTor** as a sophisticated function for Bento which an onion service operator can install and run on the Bento servers, without changing the Tor architecture. Bento also allows us to compose functions (such as **CenTor** and μ Tor) for additional client-side performance improvements.

While neither the idea of a CDN nor of multi-path routing are themselves new, we provide the first immediately deployable instantiation of these techniques for Tor’s onion services. Additionally, while the programmability of Bento simplifies the process of building and implementing such improvements, it provides no guarantees that these improvements themselves are safe and secure to deploy, without any unintended or harmful side-effects on users. As such we must also carefully analyze our designs in the context of Tor to ensure that the deployment of such functions do not harm their users (particularly users’ anonymity) and to quantify any trade-offs that exist. To do this, we conduct a comprehensive evaluation to study these techniques in terms of both performance and anonymity. We analyze their security and present quantitative results showcasing the tradeoff between users’ anonymity and performance gains. Our performance analysis indicates that both of our proposals result in significant performance improvements for clients. Our anonymity analysis allows us to quantify any loss that would occur through the deployment of our CDN proposal, and allows us to conclude that by sacrificing just a small amount of anonymity, users can achieve significant performance guarantees.

The aforementioned performance and anonymity trade-offs point to one final benefit of our proposed architectures: they provide individual users of onion services with a new flexibility to tailor their anonymity, bandwidth overhead, and latency preferences. While

the notion of the anonymity trilemma [23] states that anonymous communication protocols can only achieve two out of the three properties (strong anonymity, low bandwidth overhead, and low latency overhead), the techniques we present in this paper allow individual users and/or onion service providers a new degree of flexibility to trade off in this space themselves. For example, **CenTor** lets an onion service user choose lower latency but slightly weaker anonymity by allowing them to specify a (geographically close) region in which they wish to access an onion service replica.

Our Contributions. In this paper, we demonstrate how to make Tor onion services more usable and performant for clients and more secure for service operators, without making any changes to the Tor source code or protocol. Because one of our primary goals is deployability, we make all of our code publicly available¹. In summary, we make the following contributions:

- We present **CenTor**, the first CDN for Tor onion services.
- We quantitatively analyze how geographical-awareness for an onion service CDN and its clients could impact a user’s anonymity.
- We implement **CenTor** using the Bento architecture and evaluate it on both the live Tor network as well as through a series of simulations, demonstrating that it can provide significant performance benefits to users.

2 Background and Related Work

2.1 Content Delivery Networks

At its core, a Content Delivery Network (CDN) aims to improve Internet service quality by replicating the data from an original host server to replica servers scattered (geographically) all over the world. Clients’ requests are then directed to nearby CDN servers, improving response time and reducing latency [35, 62, 70]. A CDN can also prove to be advantageous in adverse conditions, such as a host’s hardware failure, as its distributed nature ensures content availability and redundancy. CDNs’ global deployments and large amounts of bandwidth can also make websites more resilient to denial of service attacks [19, 89], and arguably are what has ultimately made DoS attacks no longer an existential threat to the Internet.

While CDNs have played a key part in the (non-anonymous) Internet, to date, anonymous networks have generally not enjoyed the same benefits. While not directly related to Tor, part of our work is similar in spirit to the non-anonymous CDN-on-Demand system by Gilad et al. [36], which allows any user to rapidly scale up or down its web servers via the cloud.

2.2 Tor Background

Tor [29] is the most popular and widely used anonymous communication network today. This low-latency circuit-based, application-level overlay network system is based on the concept of onion routing [37]. In general, every Tor “message”, called a *cell*, passes encrypted through a 3-hop circuit comprised of entry, middle, and exit nodes (as chosen by the client from a list of available Tor nodes), wherein each hop removes a layer of encryption before it finally reaches its target Internet destination. A client instructs the exit relay to connect to the desired external Internet destination by

¹<https://github.com/BARC-Purdue/CenTor/>

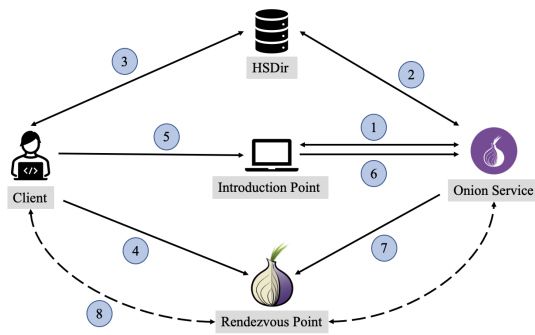


Figure 1: The Tor Onion Service Protocol. All steps and communication legs are done over Tor circuits.

creating a *stream* through the circuit and uses TCP at the network layer for every hop in the circuit.

Onion Services. Onion services (formerly hidden services) allow servers to anonymously host content and web services. At a high level (eliding details such as key establishment), the onion service protocol operates as follows (also shown in Figure 1).

A Tor user wishing to operate an onion service selects one or more onion relays, creates anonymous circuits to them, and requests that they serve as *introduction points* (IPs) for its onion service (*Step 1*). IPs are the publicly reachable “gate keepers” that clients first contact to reach the onion service. If the onion relays agree to serve as IPs, they allow the circuits from the onion service operator to persist, and the onion service operator publishes the list of IPs (along with relevant key material) on a distributed hash table (DHT), indexed by the onion service’s name (*Step 2*).

When a client wishes to connect to an onion service, it looks up the IPs (and key material) on the DHT (*Step 3*). Before connecting to the IPs, however, the client recruits a relay of its own (over another circuit) to act as a *rendezvous point* (RP) between itself and the onion service (*Step 4*). Whereas IPs serve to initially connect clients and services, RPs proxy communication between them. Once it recruits its RP, the client opens another circuit to one of the service’s IPs, and sends a request (encrypted with the onion service’s public key) for the service to “meet” the client at the RP (*Step 5 & 6*). The service then creates its own circuit to the RP, and the client terminates its circuit to the IP² (*Step 7*). At this point, the client has a circuit to the RP, as does the service. All communication between the two goes through the RP then, resulting in a longer, 6-hop circuit (3 chosen by client, including the RP, and 3 chosen by the server).

As a result, the client is able to communicate with the service, yet neither the client nor the service learns the other’s identity (*Step 8*). We will refer to traditional onion services that operate in this manner as *legacy* onion services. Onion services can additionally operate in a *non-anonymous* mode³ [27], which facilitates direct connections from the client’s RP to the onion service, thus cutting out three hops in the connection. We make use of this feature to improve performance, but demonstrate how, through our design, we still maintain the anonymity of the original onion service provider.

²This leads to very short-lived circuits between clients and IPs, which Kwon et al. showed made them easily fingerprintable [57].

³That is, set `HiddenServiceSingleHopMode - 1` in the `torrc` configuration file.

Programmable Anonymity Networks. Bento [77] enhances Tor’s capabilities by allowing relays to serve as user-programmable middleboxes and integrate network function virtualization (NFV) into anonymity networks like Tor. Bento allows Tor relays to opt-in to operate as Bento servers, offering users the ability to upload and execute custom functions and providing a safe and secure platform for them to customize their Tor experience and features. These functions are written in high-level, fully featured programming languages, such as Python, and can handle complex middlebox operations. Previous research has shown the advantages of integrating programmable middleboxes into the Tor network, including the ability to balance load through a dynamic `LoadBalancer` function [77]. However, we step well beyond these capabilities and showcase more sophisticated CDN functionality.

We choose Bento to showcase our design, implementation, and evaluation, due to its ability to quickly deploy new functionality within the Tor network without requiring any changes to the codebase, though our overall CDN design is more general and adaptable. `CenTor` could be utilized in other programmable networks such as FAN [78], which seeks to achieve application-level control. However, FAN involves modifying the Tor codebase, and only permits trusted plugin developers to deploy new plugins at this time.

2.3 Improving Tor’s Performance and Security

A significant amount of research has been done towards improving Tor’s performance. This includes work to improve Tor’s circuit construction algorithm [83, 84], congestion control [5, 76], transport protocol [11, 24, 91] and circuit scheduling [12, 88], as well as to perform real-time circuit classification [4].

Work has also been done on onion services specifically, as they suffer from a number of additional performance concerns [98]. Tang et al. [88] implement an advanced circuit scheduling algorithm based on the circuit’s recent activity. Øverlier et al. [67] proposed utilization of pre-distributed Diffie-Hellman values for constructing circuits to reduce the overhead of the server connections.

Prior work has also looked at improving the security of onion services, as they are prone to de-anonymization attacks [55] like fingerprinting [57, 68] and duster attacks [41]. `SGX-Tor` [56] works to enhance the security of Tor by running it within an Intel SGX enclave, at the cost of moderate performance loss. Øverlier et al. proposed the use of valet nodes to reduce an onion service’s vulnerability to DoS attacks [66], while `CLAPS` [79] provides security improvements in entry guard design.

Location-Awareness. Tor generally selects relays with a probability based on the node’s consensus weight. However, there has been a variety of research that proposes *client-location-awareness* while constructing client-side circuits [31, 54, 79]. This concept reduces the distance between clients and relays, thereby improving the client-side performance. Although such practices optimize path selection, they also have the potential to leak a client’s anonymity.

One such work, `CLAPS` [79], provided a method for location-aware path selection and load balancing, showcasing improvements in both performance and security when compared with prior proposals. `Counter-RAPTOR` selects more resilient relays for its guard nodes while constructing circuits to mitigate attacks [86, 87]. They considered relays to have high resilience if they lie in Autonomous

Systems (AS) with high Tor bandwidth. DeNASA, on the other hand, provides a destination-naive but AS-aware path selection algorithm that achieves time-to-first-byte close to vanilla Tor [10].

To analyze clients' anonymity when location-aware path selection is employed, CLAPS provides an anonymity function that takes into account the total client density in a region, the entropy of the distribution of users across locations (in an AS), and the entropy of the distribution of users across countries [79]. We base the framework of our anonymity analysis on this work, but modify the anonymity function with additional parameters to suit our case as explained in Section 4. We provide a destination- and client-aware path selection technique building circuits preemptively and achieving performance benefits.

While we are inspired by and build off of several of the aforementioned ideas in our architecture, we are able to achieve both performance and security improvements without introducing any required protocol changes and in a more flexible manner, which we argue could make our techniques easier to deploy in the future.

3 CenTor: A CDN for Onion Services

CenTor is the first Content Delivery Network (CDN) for Tor's onion services. It realizes a variety of traditional CDN features such as optional geographical-awareness to bring content closer to users, dynamic content, and protection from DoS attacks. We start by introducing our goals when designing our CenTor CDN. We then discuss our threat model and provide a motivating example before delving into the details of our design and realization.

Goals. While CDNs are common and often used on the non-anonymous Internet for increasing the performance (and security) of web services, as we discussed previously, they have not seen the same adoption for anonymous web services. Our main objective is to bring some of these same performance and security benefits to onion services, for the first time.

Our first goal is to increase the scalability of onion services, allowing them to dynamically adjust to client traffic and demand without operators needing to deploy their own hardware or manually duplicate their onion service. We will refer to these additional nodes as *onion service replicas*.

Our second goal is to improve the performance of onion services by reducing the latency that clients experience and moving content closer to them. In addition to dynamically scaling replicas based on traffic load, we also wish to host these replicas across multiple geographic regions that are independent of where the original onion service is located. These geographically-aware replicas allow us to more broadly disperse content, increasing the odds that a replica is close to a potential user.

Third, we wish to support modern web application features like dynamic content, in addition to static websites. Support for dynamic content requires us to ensure that there is a mechanism for an onion service to safely and securely stay connected to its replicas. Our design must be flexible enough to support such differing use cases.

Fourth, we wish to provide greater resilience to DoS attacks. Recall that non-anonymous CDNs like Akamai secure against this in a variety of ways, including restricting access to the primary webserver by only allowing Akamai to connect to their customers' machines [89]. We wish to achieve this same sort of protection for

onion services, as well as ensure the availability of content even during excessive client traffic.

Fifth, we aim to improve the resilience of onion services to de-anonymization attacks. Even if the location of replicas are revealed, we wish to provide stronger guarantees and protections for the location of the original onion service host.

Finally, we need to ensure confidentiality of content being provided by the replicas, as well as enforce integrity and honest execution to ensure that content is faithfully delivered. While traditional CDNs often rely on trusting the CDN operator to achieve this, given the potential for sensitive content, as well as the expectations for privacy, we wish to ensure strong enforcement of these properties.

3.1 Threat Model

Our threat model is largely inherited from prior work. It is primarily based on that of the existing Tor network [29]. We assume that a powerful adversary, for instance, a nation-state [81], controls a small number of Tor relays and can therefore observe a fraction of traffic on the Tor network. Further, the adversary has the potential to modify traffic using its controlled relays by participating in the network. This also enables the adversary to perform traffic correlations and traffic analysis attacks [13].

As we leverage the Bento framework to achieve many of our goals, we also inherit its threat model. The Bento design is based on loading and executing functions on a Bento server, which is essentially another user's machine, which may lead to supplementary threats. To mitigate these concerns, Bento (and hence our techniques) relies on recent advances in trusted execution environments (TEEs), namely enclaves ("containers of enclaves") [38]. With such a TEE we assume that an adversary who runs a malicious Bento server cannot introspect on or influence functions running inside an enclave. We, therefore, consider such environments to be safe to both execute code, as well as store data such as web service content. While this work (and [77]) was built using SGX, we note that the techniques will work with any TEE with similar properties [38, 77]. And though there have been attacks on SGX [92], we do not believe that any of these represent fundamental flaws in all TEEs (and typically these have all been mitigated and patched [21]), and we note that TEEs are being used in a number of real world applications [63]. Nonetheless, we provide a more in-depth discussion of the implications of SGX compromise in Section 6.

3.2 Overview

We now provide an overview of how CenTor operates as a Bento function. It is important to note that this is just one (immediately deployable) realization of our CenTor CDN design. To use CenTor, an onion service operator Alice performs the following steps:

Choosing Replicas. Alice first identifies Bento servers for every *shadow* (a geographical region) to employ them as *replicas* for her service. She does so by accessing a database of Bento servers showing their availability per shadow (see Section 3.3).

Deployment. Next, Alice deploys the CenTor function onto these middlebox nodes, thus enlisting them as replicas, and sends the content of her service to the selected node(s). After receiving the required contents, the node(s) deploys the onion service in a non-anonymous fashion (which reduces the number of hops and hence

latency) and returns an alias `.onion` address to Alice, who can then publicize this along with their respective shadow. If she so chooses, Alice can then go offline, allowing the replicas to automatically serve her static content⁴.

Accessing the Replica. Suppose Bob now wishes to access Alice's onion service (without knowing who or where it is). He first accesses a list of Alice's onion service's replicas, which allows him to choose a replica which is "local", or in other words geographically lies in the same shadow as him. Bob can then connect to this replica by running a shadow-specific client script. This script ensures that Bob browses Alice's onion service through a (client-side) Tor circuit which consists of relays that are in the same shadow that he has chosen as well, so that he benefits from the geographic clusters.

Composing Functionality. The framework we have chosen also allows for easy composition with other Bento functions. This allows Alice to further improve the performance of her onion service by composing CenTor with other techniques such as LoadBalance [77], which will allow her onion service to have multiple replicas (these share the same hostname and private key) within the same shadow that will automatically scale up and down based on client traffic. This can also further strengthen DoS resilience.

We now describe these steps and the concrete design and implementation decisions that allow us to achieve our performance and security goals in detail, plus briefly discuss how CenTor provides security even though it runs on other users' machines.

3.3 CenTor Design and Implementation

We sought to develop an infrastructure that will allow users to deploy scalable onion services by replicating their content across multiple Tor routers. At a high level, we accomplish this by developing Bento functions to support delivering web content and establishing circuits, as per the onion service protocol, and that allow for adding and removing these onion service replicas on demand. It is worth noting that we use the standard Tor protocol for building circuits. In addition to the Bento infrastructure, we utilize the Python Stem library [22] for our implementation and do not change the underlying Tor code in any way.

We start with a brief summary of our design: An onion service operator designates certain Bento servers to act as its replicas by sending the content of the onion services to these node(s), where it is then stored. The replicas communicate with the onion service operator over a Tor circuit to completely protect and isolate the operator. The onion service's respective clients can then access its contents by locating a geographically "close" replica (in what we refer to as a *shadow*, thus reducing latency. We now dive more deeply into these concepts and the exact design of CenTor.

Shadowing. We first introduce the concept of *shadowing* as that features heavily in our design to obtain performance benefits. To connect with the most beneficial replica, each client selects a region, called a *shadow*, corresponding to its geographical location. The client is then restricted to choosing relays within this shadow to construct its circuit. Our goal here is to maintain client anonymity, i.e., the shadow should not reveal the client's (precise) location, while still moving content closer to the user. These shadows can

⁴We provide a detailed discussion in Section 3.3 of how this process changes if Alice's service supports dynamic content or if she chooses to have a more sophisticated setup.

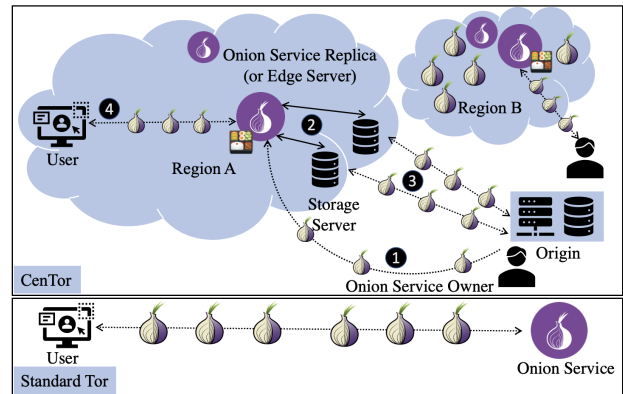


Figure 2: Architectural comparison of using the CenTor architecture versus standard Tor to access an onion service. The numbered steps correspond to the four phases presented.

be customized by each individual client at the level of location granularity information provided by Tor for its relays, allowing for a controllable trade-off between latency and anonymity.

Components. CenTor has a number of components that all work together to realize the CDN's functionality. The *origin* is owned and managed by the operator and includes the backend web servers, storage facilities, and databases that aid in hosting the onion service. The operator may also choose to use third-party storage (a *pseudo-origin*), for instance, a cloud service, which may host the operator's website content. This may further strengthen the operator's anonymity in case of a deanonymization attack, which we discuss in more detail later. *Replicas* are a copy of an onion service that support clients in a specific shadow. These replicas are all hosted on Bento nodes. Replicas may additionally have more sophisticated behavior and act as the *edge servers* to request content from the origin to support dynamic onion services. It is worth noting that this feature may not be required to support static services and some onion services, for instance, web pages that show dynamic data like live-weather. In such a scenario, the operator may choose to instead store scripts that fetch this data within the replica. They may also parse and prefetch content from the origin server, as appropriate, to ensure that it is already in memory when the user's browser requests it. One of the strengths of our design is that operators can customize their replicas (and hence their CDN experience) in this and many other ways based on their specific requirements.

Next, we present the usage of our CenTor CDN framework in four main phases, describing how the aforementioned components fit together. These four phases correspond to the labels in Figure 2.

Phase 1: Setup. In this phase, the onion service operator selects its replicas, belonging to different shadows⁵, from the provided database of Bento servers who are willing to host an onion service. The operator then sends the relevant onion service content (such as files or scripts) to the chosen replicas, which then host the onion service, in a non-anonymous fashion, on the operator's behalf. The operator also publishes the information required to access the replica services, along with their respective shadows.

⁵We note that it is not a requirement to place replicas in different regions but rather a suggestion to improve performance.

To advertise the replicas, the operator can choose to publicize the *.onion* address⁶ on portals, such as Hidden Wiki⁷, which serve as a directory of links to other *.onion* sites [97]. We also envision that a similar community portal (like Tor [73]) can exist for Bento nodes to advertise the services they offer, like CenTor support, along with provided storage space, as well as for onion service operators to publicize their respective *.onion* shadow-wise links. It is worth noting that Tor users already use such portals to access onion services [71]. If an onion service operator needs to shut down and migrate her service, she may be able to keep the same *.onion* address by retaining the service’s hostname and key-pair files.

Phase 2: Storage Servers. To further assist the replicas, we include the (optional) use of storage servers. These servers can store data on behalf of the operator, ensuring that the service remains active even when the operator is offline. This also helps to manage large or complex onion services, as replicas would then easily access and cache the required data. These storage servers can be owned by the operator or provided by third-party cloud servers.

Phase 3: Dynamic Content. To manage onion services that support dynamic content, we provide CenTor users with the flexibility to choose a suitable architecture based on the nature of their respective onion service. For instance, in the case of onion services with dynamic data that does not require maintaining state with the origin, operators can directly transfer update scripts to the replica [18], which will then automatically fetch state as needed. For stateful dynamic content, the onion service operator will generally need to maintain a connection with its replicas, allowing them to access and serve stateful content which is stored on the origin. However, if they wish to go offline, they could, for example, delegate this task to a number of storage servers and then go offline. For further enhancement, the operator could additionally use replicas like CDN edge servers to prefetch content from the (pseudo-)origin.

Phase 4: Execution. When a client wishes to access an onion service that uses CenTor, they first identify the shadow that serves the region where they are located or that they wish to use. Based on this, they then select the appropriate replica from the database that lists their availability along with the *.onion* address. To connect to and access the selected replica, the client runs our shadow-specific client script. Figure 2 showcases the client and onion service’s communication in the case of CenTor and standard Tor.

CenTor Implementation. We implement our CDN design as a function within Bento, the aforementioned NFV-like framework of [77]. The CenTor function is then loaded onto one of the Bento nodes by an onion service operator that wishes to deploy our CDN. When the operator first executes the function, it lets the operator send the service’s static contents, after which it launches the respective service and sends the operator an alias *.onion* address, which can be used as the domain name specific to that shadow. The function then continues to run and serve content until the operator terminates it⁸. It is worth noting that Bento servers can securely

host multiple services because of the strong security, privacy, and isolation guarantees provided by Bento.

Content protection. Recall that our last design goal was to ensure that CenTor provides strong confidentiality and integrity guarantees for any hosted content. As each CenTor replica executes inside its own enclave on the Bento server, all content inherits these protections (as well as strong isolation from any other replica). We discuss this further in Section 5.3. For additional data that must be stored outside protected memory, the Bento architecture enforces encryption before a write to disk with an ephemeral key which is only available to the specific enclave running that replica [38, 77]. We note that these features also provide a replica with the same level of protection and plausible deniability as Tor currently provides relays in situations where an onion service operator may try to host illicit content on them (see [77] for a further discussion).

Additional Features. While not something that we specifically provide, we note that the Bento architecture contains a notion of *middlebox node policies*, which allow a middlebox to specify what services it would like to provide⁹. CenTor users would then choose middlebox nodes that meet their requirements to host their content on. Additionally, to verify that CenTor is truly running inside an enclave and will thus provide all the discussed security and privacy guarantees, as well as ensure the Bento node is fully patched against any known vulnerabilities, the client can utilize SGX’s remote attestation feature [53]. CenTor also aims to deliver scalability by allowing onion service operators to leverage on-demand replicas by automatically scaling them up or down based on load. This is done in a manner similar to the techniques presented in [77].

μ Tor: Enhancing CenTor through multipath routing. Finally, we introduce μ Tor, a multipath routing design that establishes multiple communication paths between a client and server. Unlike previous proposals for multipath routing in Tor [3, 24, 102], which require changes to the underlying Tor protocol, μ Tor is uniquely built as a client-side Bento function, preemptively creating μ paths. It prioritizes low-bandwidth relays, enabling clients to reach a CenTor instance through multiple paths and gaining performance benefits from reduced hops. Our evaluation assesses μ Tor’s impact on performance improvements and anonymity. Due to space constraints, we provide a more detailed discussion of μ Tor implementation details and performance and anonymity evaluation in Appendix B, as well as a discussion of related work.

4 CenTor Anonymity Analysis

Now that we have presented our CDN design, we must discuss its potential impact on client anonymity. While the programmability of Bento eases the implementation and deployment of such functionality, it provides no guarantees that this functionality will not have unintended consequences. It simply facilitates deployment; it is then up to a function developer to assess the impact of doing so.

⁶.onion addresses enable easy integration of CenTor with Tor infrastructure, requiring no code or protocol adjustments. This simplifies deployment compared to alternative replica addressing methods, which introduce complexity, demand non-backwards-compatible changes, and necessitate additional mechanisms like anonymized payments for domains/clouds.

⁷Hidden Wiki allows users to create accounts anonymously thereby allowing them to propose *.onion* site links [8].

⁸Or the Bento operator shuts down.

⁹Bento operators publicly define their “middlebox node policy”, outlining the API calls they are either willing or unwilling to support and enabling them to specify the permitted system calls and allocation of resources to functions. These are rigorously enforced through the Bento architecture. An operator utilizes the Tor directory to discover Bento servers that align with their service’s criteria and selects one randomly.

4.1 Anonymity Analysis

Our CenTor design promises performance benefits to onion service clients in large part by operating on a shadow-based subset selection from the total available set of Tor relays. The chosen subset then acts as a pool for the guard, middle, and exit nodes (i.e., the circuit nodes) for the client. Therefore, there is a performance versus anonymity trade-off inherent in our proposal. We provide a detailed assessment and quantification of this potential anonymity loss in this section. We believe that this new anonymity analysis would also apply more broadly to future location-aware CDN proposals that utilize a similar high level design, even if they do not use NFV.

Anonymity Threats. As per the design of CenTor, clients that wish to gain performance benefits choose circuit relays which lie in the same shadow as their geographical location. The primary attack that we consider is one in which an adversary who partially observes multiple circuits is able to link them to a common user of a replica onion service, thereby detecting the user's location and reducing their anonymity [50].

Anonymity Function. We provide CenTor clients with the flexibility to strike a balance between anonymity and performance. To help with this, we offer them the ability to select a point on the anonymity-performance scale and provide minimum threshold values as guidelines, allowing them to make informed decisions that align with their privacy and performance needs. We consider the parameters described in Table 1. For clients, the anonymity set is defined as the amount of traffic in a specific region, as this determines the set a user can hide among [29]. We translate this into an assessment of the total client density in a specific shadow, as well as the entropy of user distribution across ASNs (Autonomous System Number) and countries, similar to [79]. Furthermore, we evaluate the relative number of Tor nodes (entry and exit nodes) involved in routing traffic in a chosen shadow. We draw inspiration for the parameters in Table 1 from prior work [79], which has seen acceptance by the Tor community, including the integration of the corresponding code. Here, S represents a shadow and x a set of locations (such as ASN l or country c). We compute the entropies $EL(S)$ and $EC(S)$ across all l and c within the S , respectively (there can be multiple l and c within S). The entropy-based criteria allow us to quantify anonymity with respect to an adversary who lacks prior knowledge of a client's ASN or country, ensuring sufficient anonymity over these sensitive attributes. To estimate ASN density, we utilize Tor's measured user-per-country statistics [73] and distribute users into ASes within their country proportional to the number of IPv4 addresses each AS originates, as explained in [79] and Section 4.2. Prior work has indicated that reducing the client anonymity set by $20x$ (i.e. having at least 5% of total Tor users, ASNs, and countries) still provides strong anonymity [79]. As such, for strong anonymity, CenTor users should aim to have CD , EL , and EC values of at least 0.05 for a chosen S . Additionally, previous research suggests that a selected cluster or subset of Tor nodes should contain 20% of total guard relays for good anonymity [1]. We extend this requirement to minimum relay density within a shadow. Therefore, RD and ED should have values of at least 0.2 for a chosen S . We understand that exit nodes serve as gateways for encrypted Tor traffic to reach the Internet, making them vulnerable

Table 1: CenTor Anonymity Function Parameters.

Symbol	Description
D_x	Client density (the fraction of all Tor client traffic) in a given location $x \in \{c, l\}$ where l is an AS and c is a country.
$CD_x(S)$	Total client density in set of location(s) x in shadow S , as $\sum_{x \in S} D_x$.
$EL(S)$	Entropy of client distribution across locations (ASNs) in a shadow, calculated as $\frac{-\sum_{l \in S} [D_l / CD_l(S)] \log_2 [D_l / CD_l(S)]}{\log_2(l_{max})}$, where l_{max} is the maximum number of ASes in shadow S .
$EC(S)$	Entropy of client distribution across countries c in a shadow, calculated as $\frac{-\sum_{c \in S} [D_c / CD_c(S)] \log_2 [D_c / CD_c(S)]}{\log_2(c_{max})}$, where c_{max} is the maximum number of countries in shadow S .
$RD(S)$	Total Tor relay density in shadow S .
$XD(S)$	Total Tor Exit relay density in shadow S .
$ED(S)$	Total Tor Guard relay density in shadow S .

to abuse and monitoring by adversaries. Therefore, it is recommended that CenTor clients aim for a diverse set of exit nodes by ensuring XD is at least 0.5 for a particular S , given that exit relays are approximately 2.5 times fewer in number than guard relays.

To maintain anonymity, it is important to meet certain minimum requirements for each parameter. To accommodate the variations in minimum acceptable values for the chosen parameters, we normalize each parameter¹⁰ described in Table 1 using the formula

$$\overline{parameter} = \frac{parameter_{oalue} - \min(parameter)}{\max(parameter) - \min(parameter)}$$

to ensure that their values range between 0 and 1, with 0 representing the minimum acceptable value (for example, 5% of total Tor users) and 1 representing "perfect" anonymity, such as that provided by the entire Tor network. We then calculate an anonymity score $\alpha_{CenTor}(S)$ for a specific shadow S by taking the average of these normalized values for the chosen parameters¹¹. Specifically, we use the formula¹²

$$\alpha_{CenTor}(S) = \frac{\overline{CD}(S) + \overline{EL}(S) + \overline{EC}(S) + \overline{RD}(S) + \overline{XD}(S) + \overline{ED}(S)}{6}$$

based on [79] but extended to fit our specific use case. This score helps clients make informed decisions about how much anonymity they are willing to sacrifice in exchange for performance gains.

This analysis assumes that Bento is pre-deployed on Tor relays. However, this assumption does not impact user anonymity, as all user interactions with Bento occur over a Tor circuit, and thus to any potential adversary the activities within Bento remain indistinguishable from typical Tor traffic [77]. Therefore, the quantity of Bento servers is not considered in the anonymity analysis.

We recognize that the anonymity score $\alpha_{CenTor}(S)$ provides only a generalization of the level of anonymity in a given shadow. For this reason, we recommend that extremely cautious clients take the time to also thoroughly examine the score of each parameter and determine which ones are most relevant to their specific needs. By doing so, they can gain a more granular understanding of the anonymity of their shadows and make more informed decisions about which parameters to prioritize and to what extent.

¹⁰We denote a normalized parameter as \overline{CD} , for example.

¹¹For more detailed explanations as to why we consider all parameters to be equally significant, as well as the calculations for each individual parameter, we refer the interested reader to Appendix A.

¹²To simplify notation, we use $\overline{CD}(S)$ as a generic term for $\overline{CD}_c(S)$.

4.2 Anonymity Evaluation

To evaluate the anonymity score α_{CenTor} for different shadows, we developed a database based on [45, 73]. We gather the number of IP addresses per ASN, and Tor users, relays, entry and exit nodes for every country. We then divide the IP addresses per ASN proportionally to the number of Tor users in that country as done in [79]. The resultant database is used to compute the score (based on a chosen S). These experiments were carried out in May 2021. We present the results in Table 2 and discuss their implications in more detail now. To provide further information on our chosen shadow specifications, we have included additional details in Appendix J. **Using α_{CenTor} .** To provide a more comprehensive understanding of how our architecture can be used safely and effectively, we now discuss how a CenTor user can carefully utilize it with the assistance of both the single-score metric α_{CenTor} as well as the component scores. To better visualize the potential anonymity levels provided by our architecture, Table 2 provides an approximate value of α_{CenTor} for different shadows. These values allow us to determine whether the anonymity levels meet our predetermined minimum requirements. In general, positive values of α_{CenTor} are desirable, indicating that the anonymity requirements we have set are generally met for each sub-component. For example, recall that we consider a minimum of 5% client density to be sufficiently safe for a shadow. A score of 0 for CD indicates exactly meeting this 5% minimum threshold, while a negative score means the shadow fails to meet this requirement, and a (large) positive indicates surpassing it. Since this holds for all components, a higher α_{CenTor} score generally indicates many non-zero (safe) components.

Based on our evaluation, we can conclude that a number of shadows, such as Western Europe, Europe, and Eurasia, provide strong anonymity that meets all of our minimum requirements for each parameter. Many of these regions are actually geographically quite small, meaning they are also ideal for our CDN purposes.

While α_{CenTor} provides a useful high-level overview of anonymity levels, we believe that it is also important for CenTor users to inspect each individual parameter for assurance. Our anonymity analysis easily allows clients to select which parameters are most important to them and to what extent¹³, allowing them to choose a shadow that aligns with their specific risk tolerance. For example, shadows such as APAC and Central & Eastern Europe have borderline α_{CenTor} values (close to 0), which may indicate that certain individual parameters have negative values. In such cases, we strongly recommend that CenTor users inspect each parameter carefully to ensure that the overall level of anonymity is sufficient for their needs. For shadows such as the Americas and Africa, which have many negative components (and thus an overall negative score), users face a much higher risk of being deanonymized. As a result, we do not recommend the use of these shadows for CenTor users who require a very strong level of anonymity.

Ultimately, choosing a shadow involves balancing performance and risk, and our analysis helps with this. Balancing these considerations can be a complex task though, so, to aid service operators and clients, we provide (safe) recommendations in Appendix G.

Discussion. Our anonymity analysis has also led to a few unexpected insights. For example, despite what an outside observer

might guess, the United States does not offer as strong anonymity properties for CenTor as one may expect. Out of the total 1398 exit nodes, only 324 are located in the region, resulting in a low $XD = 0.23$, well below our threshold of 0.5. Additionally, the United States only has 539 entry nodes out of a total of 3664, resulting in an $ED = 0.14$, which falls below our threshold of 0.2. Our analysis also reveals that including Europe in a shadow can significantly enhance the anonymity of CenTor users. Even Western Europe, despite being one of the smallest shadows in our study, offers robust anonymity guarantees to CenTor users (Germany alone hosts 978 entry nodes). While the large number of nodes in Germany might be well-known to the community, facts like this might be unknown to a new user, pointing to the usefulness of our scoring system.

We believe our analysis might be more broadly useful as well. Our findings can help inform where it might be the most beneficial to set up new instances of various node types. For example, we recommend strengthening the relay density in regions such as Africa, South America, and APAC, where it is currently comparatively low. We have quantitatively shown that incorporating a diverse set of relays could enhance the existing anonymity set (and performance) for similar classes of location-aware Tor systems. While pages such as Tor Metrics [73] do quantify raw numbers for some of this information already, we nonetheless found it surprising to see the real impact these node densities can have when quantifying anonymity.

Limitations. Opting for a (weak) shadow with a low α_{CenTor} score may result in both the guard and exit nodes of a circuit being within the same AS¹⁴. This proximity increases vulnerability to traffic analysis, a limitation of our current architecture. The $EL(S)$ parameter in α_{CenTor} is designed to help address this concern, promoting a broader distribution of users across multiple ASes, which challenges the singling out of user-specific traffic due to a higher volume of Tor traffic. It is worth noting that CenTor leverages Tor’s circuit selection algorithm as well, facilitating the establishment of circuits across multiple ASes. However, we acknowledge that this still might not be enough to guarantee the prevention of traffic analysis by AS-level adversaries, as demonstrated by [64, 87].

5 CenTor Evaluation

We now evaluate the performance and security benefits of CenTor.

5.1 Performance Evaluation

Overall, CenTor proved to considerably reduce latency and download times when compared to standard Tor, as demonstrated throughout this section. For our experiments, we deployed our client in the U.S, and our onion service in Brazil, which hosted a 10 MB file that our client downloads. We deployed our own Bento node (Intel NUC i7-10710U) in the United States with 12 CPUs, 32 GiB RAM, and Ubuntu 18.04 OS. Our experiments use multiple Amazon EC2 T2 instances (with 2 vCPUs, 4 GB RAM and Ubuntu 18.04 OS) to host the onion services and make client requests. Where relevant, error bars on our plots indicate the 95% confidence interval.

For the first set of experiments (presented in Figure 3), we chose the United States—a “weak” shadow (as that is where our client was located). Recall that while connecting to a legacy onion service, there are two 3-hop circuits (client-side and onion-service-side).

¹³The individual score could even be adjusted to weight components.

¹⁴ASes can observe traffic equivalent to the percent of Tor nodes owned in the shadow.

Table 2: α_{CenTor} scores for different shadows. A score of 0 indicates our minimum requirements to ensure client anonymity are exactly met, 1 corresponds to a “perfect” shadow that encompasses the entire Tor network, and a negative value is below the requirements. Any positive number is generally considered safe, with larger numbers indicating greater anonymity.

Shadow	$\overline{EL}(S)$	$\overline{EC}(S)$	$\overline{CD}(S)$	$\overline{ED}(S)$	$\overline{XD}(S)$	$\overline{RD}(S)$	$\alpha_{\text{CenTor}}(S)$
Africa	0.647	0.783	-0.038	-0.25	-1.0	-0.247	-0.017
Africa & Europe	0.677	0.610	0.475	0.671	0.364	0.591	0.564
Western Europe	0.485	0.586	0.124	0.396	0.151	0.330	0.345
Central & Eastern Europe	0.501	0.502	0.083	0.171	-0.538	0.147	0.144
Eurasia	0.830	0.694	0.580	0.677	0.389	0.641	0.635
Europe	0.630	0.581	0.673	0.671	0.364	0.587	0.584
APAC	0.626	0.524	0.225	-0.21	-0.948	-0.145	0.012
Americas	0.505	0.288	0.257	-0.181	-0.960	-0.176	-0.044
Americas & Western Europe	0.523	0.491	0.435	0.466	0.193	0.403	0.418

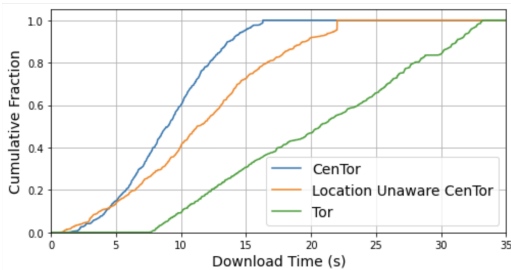


Figure 3: Download time comparison: CenTor, location unaware CenTor and standard Tor.

CenTor on the other hand, does not require an onion service-side circuit as the replica is deployed in a non-anonymous fashion, reducing the number of hops to three. We also compare CenTor’s performance with what we call *Location Unaware CenTor*—a “perfect” shadow with $\alpha_{\text{CenTor}} = 1$ —which does not employ any shadowing techniques and only makes use of CenTor’s ability to reduce the number of hops. In other words, Location Unaware CenTor simply reduces the number of hops by communicating with a (random) non-anonymous onion service replica. This provides clients with a performance and security middle ground between CenTor and standard Tor. We observe that CenTor provides about a 56.4% reduction in download time and location unaware CenTor a 34.5% reduction over standard Tor, also a perfect shadow. Additionally, clients have the flexibility to opt for shadow-awareness exclusively at the exit node, leaving the guard and middle nodes to be chosen normally and providing another middle-ground option. We discuss the details of this approach in Appendix F.

To further explore the anonymity-performance trade-off, we also evaluated a client’s Time to First Byte (TTFB) by varying the shadow size in four selected regions (see Figure 15 in Appendix C). We observed that as the α_{CenTor} score increases, i.e., anonymity improves, client-side performance decreases. *Location Unaware CenTor*, without region-based restrictions, exhibits higher TTFB, highlighting the impact of node distribution globally. Both CenTor and *Location Unaware CenTor* outperform standard Tor by reducing the number of hops and choosing nodes strategically.

Dynamic content. As we discussed previously, there are a number of ways that an onion service operator can set up a dynamic service that produces real-time data, with different usage-performance

trade-offs. One approach is to place scripts generating the information within the replica. This does not increase the delay for the client accessing the service, as the script will directly populate the service with any updated information. Another approach is to connect the replica with the origin server through a Tor circuit. In this scenario, the replica prefetches the data whenever there is an update on the origin server, so there is also no extra client-side delay. However, if the replica needs to contact the origin server to fetch data when the client accesses the service (for example, to retrieve information from a database), there will be an additional delay corresponding to the three extra hops. This will result in a download time similar to accessing an onion service using the standard Tor protocol (as the number of Tor hops will be equivalent).

We confirm this experimentally by having a client download a 20MB file from an onion service, which in turn dynamically fetches this file from a storage server, as shown in Appendix D.

Storage costs. We conclude with a brief discussion of storage costs. We note that the average size of Alexa 50 top websites are about 1.2 MB [100]. A more detailed `amazon.com` webpage is about 4.6 MB whereas a less complex `google.com` one is 179.7 KB [99]. While initial versions of SGX provide a limited amount of protected memory (128MB with 93MB of this usable by applications), the most recent versions support paging which greatly increases this capacity, allowing for the potential of up to 512GB per socket [32, 52, 77]. Even without paging, this still allows us to store a large amount of web content in a replica. Additionally, for very large services, a storage server could further be utilized if need be or content could be encrypted and securely stored on disk until it is needed. We note that prior work has already demonstrated the feasibility of running a CDN inside SGX, and we refer the interested reader to [38] for a more detailed set of benchmarks and discussion on SGX’s capacity to support and serve modern web services.

5.2 Shadow Simulations

For our second set of experiments, we seek to answer a number of large-scale deployment-related questions that may arise within the Tor network given our CenTor architecture: How would the simultaneous connection of multiple users to the same CenTor function impact client performance and the load on the relay? What would be the effect of running multiple CenTor instances on a Tor relay, particularly when it is also handling standard Tor traffic? In cases where a region experiences low relay density but high

client density, could this situation lead to localized traffic congestion as clients converge on the limited available Tor relays? We therefore conduct a comprehensive analysis of the potential impact on Tor relays and clients of deploying Bento nodes to support the CenTor architecture, through Shadow simulations [47] to answer these questions. Shadow enables us to model large scale Bento (and CenTor) deployment on the Tor network in various configurations. **System setup.** Our simulations were executed on a system with two Intel(R) Xeon(R) Platinum 8352Y CPUs (64 total cores/128 threads) and 1.5TB of RAM, involving an array of software components: TGen for generating realistic background traffic, OnionTrace for recording asynchronous Tor events, and the Bento [77] server, which optionally ran CenTor¹⁵. Our simulations incorporated data related to Tor consensus, server descriptors, Tor relay information, available bandwidth, current capacity, and performance measurements, all obtained from Tor metrics data as of April 2023 [73]. To facilitate our experiments, we utilized TorNetTools [48], a utility tailored for comprehensive Tor network experimentation¹⁶.

Experimental setup. Within our simulation, network parameters were set to 100% of the Tor network, consisting of 6666 Tor nodes with a total capacity of 837.68 Gbit/s, 10 authorities, 5 onion services and 752,338 active Tor users. To evaluate the potential impact different deployments of our infrastructure might have on a client, we tested a client initiating a connection to an onion service. The client began with a 60-second pause before actively engaging in streaming activities involving data of diverse sizes (50 KiB, 1 MiB and 5 MiB), with 60-second intervals between each transition. Note that any performance differences will become most evident as the network reaches full operation and that the graphs converge at the end of the simulation due to the experimental setup, not our system.

Baseline. Our initial experiment focused on the impact of variable client loads for Tor only, spanning from 400,000 to 752,000 clients, to establish a baseline measurement for our simulations (see Appendix E). Subsequently, we maintained a constant client count of 752,338 while assigning Tor relays the role of Bento nodes, configured to generate Bento traffic (see Figure 4). At the 2500-second mark, we observed that the configuration with 3500 and 6000 Bento servers exhibited performance degradation of only roughly 4.4% and 9.6% respectively, compared to the baseline without Bento servers.

Multiple Clients Connecting to a CenTor Instance. To understand the implications of multiple clients connecting to a single Tor relay that simultaneously relays Tor traffic, serves as a Bento server, and hosts a CenTor instance, we configured 3500 Tor nodes to function as Bento servers. In our initial experiment, we introduced 5000 clients connecting to the same Bento server, which operates in a non-anonymous mode, while simultaneously serving as a CenTor instance. Subsequently, we increased the client count to 10,000 and compared this scenario with a Tor relay that did not serve as a Bento server. The aim of this comparison, as shown in Figure 6, was to evaluate the performance impact on the Tor relay of CenTor clients connected while it also handles regular Tor traffic. At the 2400-second mark, it was evident that the performance of the

relay supporting 10,000 CenTor clients experienced a degradation of approximately 0.4%, compared to the “vanilla” Tor relay.

Figure 5 illustrates the performance impact on a client connecting to an onion service hosted by a Tor relay that concurrently served about 5000 or 10,000 other CenTor clients, contrasted with a client interacting with a standard 6-hop onion service. At the 2200-second mark, we can see that the CenTor client, that connects with about 5000 other clients to the same Bento server, demonstrated a performance advantage of approximately 2.9% (and about 1.2% better than the CenTor client that connects with about 10,000 clients to the same Bento server) over the vanilla Tor client.

Multiple CenTor Instances on a Tor Relay. Our next objective was to assess how the performance of the Tor relays was influenced by the operation of multiple CenTor instances in conjunction with their standard Tor traffic responsibilities. To achieve this, Tor relays configured to host CenTor instances were set up to concurrently run multiple instances of these services, with the number of instances varying from 0 to 20. As depicted in Figure 7, the simulation revealed that the relay’s performance exhibited approximately 1% performance degradation for 20 instances respectively when compared to the 0 instance baseline (representing a normal Tor relay).

Shadow-Aware Routing. Finally, we turn our attention to regions with a distinct characteristic: a low relay density but significant client density, such as the US. In such a scenario, we explore how this combination can give rise to localized traffic hotspots, where clients may direct their traffic through the limited available Tor relays, potentially impacting the performance of both clients and relays operating within this environment. With this in mind, we examine how both clients and Tor relays are influenced when a subset of clients adopts our architecture within a shadow region, focusing on the US as an example. Specifically, we directed 9k clients to utilize shadow-aware routing while connecting to a non-anonymous onion service. This setup resulted in 933 out of the 6666 relays residing within the shadow network. We compare the performance of a relay as well as a client within the shadow network to those operating outside the shadow environment under normal conditions. As shown in Figure 8, we observed that at the 2000-second mark, the performance of a shadow relay exhibited a degradation of approximately 0.7% when compared to a non-shadow relay.

Next, clients within the shadow network utilized a 3-hop connection to the onion service, comprising a location-aware and location-unaware client. This performance was then compared with that of a client accessing another onion service via a standard 6-hop connection, uninvolved in the shadow network. The results, shown in Figure 9, indicate that the CenTor client achieved approximately 3.6% performance improvement, while the shadow-unaware CenTor client displayed approximately 1.9% performance improvement over the vanilla Tor client, at the 2500-second mark.

From these simulations, we conclude that while running CenTor (and Bento) at a large scale on Tor relays might marginally consume resources and slightly impact the performance of these relays, it consistently delivers enhanced client-side performance.

5.3 Security Discussion

Having demonstrated the performance improvements that CenTor can provide to onion service clients (which speaks to our first three design goals), we now discuss how CenTor protects both onion

¹⁵<https://github.com/shadow/{oniontrace,tornettools,tgen,shadow}>

¹⁶OnionTrace v1.0.0, TGen v1.1.2, TorNetTools version 2.0.0, Shadow v3.1.0, and Tor version 0.4.9.0-alpha-dev.

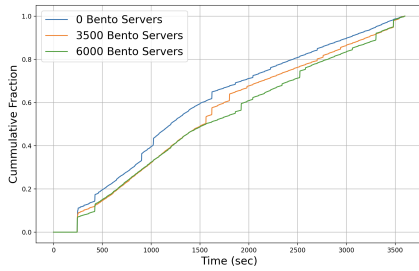


Figure 4: Client-side: Performance comparison of Tor network with the integration of variable Bento node(s).

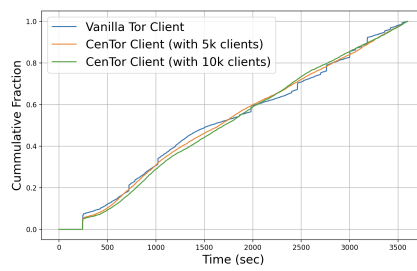


Figure 5: Client-side: Performance comparison of vanilla Tor client with CenTor client when multiple clients connect with the same CenTor instance.

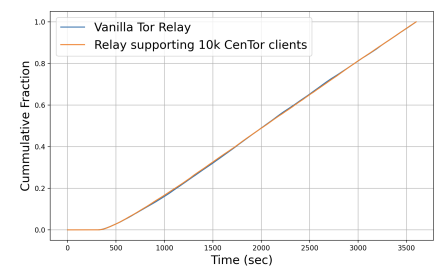


Figure 6: Relay-side: Performance comparison of vanilla Tor relay with Tor relays supporting multiple CenTor clients.

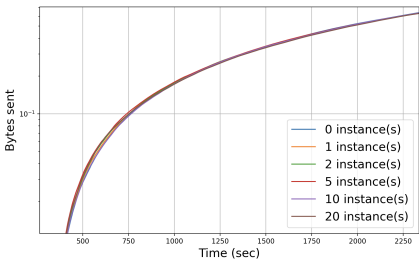


Figure 7: Relay-side: Performance comparison of vanilla Tor relay with relays supporting multiple CenTor instances.

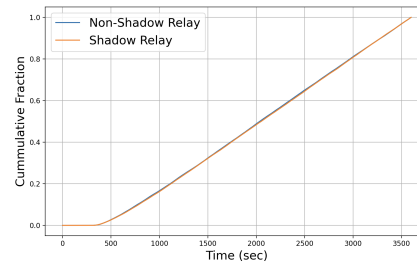


Figure 8: Relay-side: Performance comparison of vanilla Tor relay with Tor relay within a shadow.

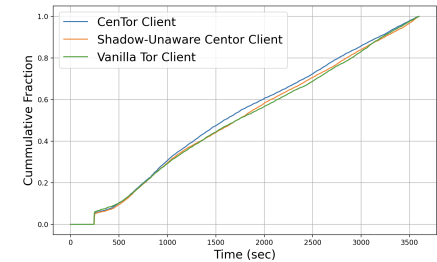


Figure 9: Client-side: Performance comparison of vanilla Tor client with CenTor clients in a shadow.

services themselves, as well as their content, against a variety of well-known attacks, thus achieving our last three design goals.

We note that as a result of our design (and use of Bento) we largely achieve our last three goals for free. Most notably, our replicas provide a level of protection and indirection for the primary origin/onion service operator, in much the same way a traditional CDN shields its clients, providing strong protections against both deanonymization and DoS. We are free to reveal the location of these replicas, as it is much less detrimental if an attacker finds them (and an operator can even quickly shut down a replica if they become concerned). In certain circumstances (such as a service with static content), our setup even allows for an onion service operator, after completing CenTor's setup phase, to go offline.

Protection Against Deanonymization Attacks. Although onion services promise their operators anonymity over the Tor network, there have been deanonymization attacks that aim to leak the operator's location [15, 41, 60]. A circuit fingerprinting attack involves identifying circuits, based on their properties, that serve onion services or their clients [57]. For instance, one such identifying property is based on the nature of HS-RP circuit (the circuit between the service and its client's RP) which we remove by eliminating this connection altogether. In general, the way operators and clients utilize our CDN alters the traditionally expected circuit behavior for onion services. Since there is no replica-side circuit, the probability of carrying out a successful circuit fingerprinting attack is reduced. Additionally, since the basic idea of CenTor is to decouple the operator and onion service from their replicas hosted on regional Bento servers, even if these already-public Bento servers

are identified, it does not leak the location of the original operator (particularly if they went offline after making the replicas available).

Our architecture for dynamic content does possess additional risks of deanonymization over others though, since the origin has to be online and connected to all replicas to maintain the state. This risk can be dodged if the onion service operator uses a pseudo-origin (say through a cloud service) which can host the service's content on origin's behalf. After this setup, the operator can then go offline, letting only the pseudo-origin maintain the state with replicas. In such a scenario, the location of the operator remains hidden in the case of a successful deanonymization attack which would then only reveal the location of the pseudo-origin.

Protection Against DoS Attacks. Having multiple regional replicas to host an operator's content and cater to a wide variety of shadows aids with the distribution of client requests and helps protect the origin. However, even if a malicious client attempts to perform a DoS attack on one of the operator's replicas, the operator can always spin them down. This detection and shutdown can even be done automatically with another function. This approach allows for a sophisticated response in the face of DoS attacks, where the provider can just choose to shut down the replica under attack, and spin up another one elsewhere. Even if the operator is not able to spin down the replica, they could just drop any connection that they maintain to it, thus rendering the node incapable of serving dynamic traffic or finding the onion service provider again. Additionally, the ability for an onion service operator to spin up and down their services based on the demand in a specific shadow ensures content availability as the onion service can be accessed

through other replicas deployed by the operator, even in the event of attack, eliminating the case of a single point of failure. Finally, an onion service operator can launch her service with various denial of service mitigation options already provided by Tor that can limit the number of connections and their periods [27]. We discuss the considered scope of DoS attacks in greater detail in Appendix H.

Protection Against Malicious CDN Nodes. Our final goal was to ensure that it was safe to deploy onion service content on another node not owned or controlled by the operator. As Bento servers run TEEs, we can assume that the CenTor function runs inside an enclave on the untrusted machine. An adversary who runs a malicious Bento server thus cannot introspect on or manipulate an onion service’s content either while it is being transferred or hosted. This means that we get these guarantees largely for free through the use of the Bento architecture. Additionally, all communication with the replicas is done over Tor circuits and hence secure channels.

6 Discussion

In addition to providing immediate security and performance improvements for onion services and quantifying their trade-offs, we believe this work also points to an interesting and unresolved set of challenges with bringing programmability to anonymity networks. While such systems allow users to easily build and deploy functions, they provide no guarantees that doing so will not have unintended or subtle consequences. Functions themselves must be carefully analyzed to quantify the impact they might have on potential users, and users need techniques and tools to understand these trade-offs.

Limitations. We are the first to introduce a deployable CDN architecture for Tor’s onion services, though we acknowledge that our innovations are currently subject to some limitations, though we believe none of these are insurmountable hurdles. Our designs require widespread adoption of Bento (or some form of programmable anonymity network), which is still a relatively new concept. However, as Bento does not require any changes to the Tor protocol, we believe there is no reason that such adoption is not possible. An additional constraint is our reliance on (Bento and Tor) relay operators and their willingness to volunteer as hosts for CenTor, as well as to permit access to resources that CenTor needs.

TEEs (and SGX). CenTor explicitly relies on a trusted execution environment, such as Intel SGX, to provide strong guarantees of confidentiality and integrity¹⁷. We acknowledge that TEE compromise has happened in the past [92, 93], though vendors typically provide mitigations [44, 93] and means of ensuring that such mitigations have been applied [53]. If a single TEE was to be compromised, content within it would become visible to the Bento server operator, and the operator would be able to tamper with or alter the execution of the CenTor function. We emphasize that this affects only the web service that is being hosted on the compromised device, as well as any clients that request content from it. The confidentiality and integrity guarantees of content on all other nodes is still preserved. We note that a compromise should have no affect on the anonymity of a client though, as all requests are made via Tor circuits, so we degrade only to standard guarantees provided by Tor, even in this case. However, as discussed previously, through the remote attestation process provided by many TEEs, all participants should

be able to ensure that functions and content are hosted on fully up-to-date nodes that are patched against all known vulnerabilities. Additionally, we argue that there are cases where, even without access to a TEE, our improvements are still useful. For instance, if an onion service is already deployed in a non-anonymous fashion with content that does not require protection, then we can still deploy CenTor. Alternatively, if the onion service operator owns or trusts the nodes hosting the replicas, then such compromises are not a concern. We still believe an interesting line of future work would be to achieve our stated CenTor goals without using a TEE or to provide strong guarantees even in the face of TEE compromise.

Currently, we have deployed special client-side functionality to allow usage of our presented improvements. We leave it to future work to integrate the ability to access CenTor replicas and utilize μ Tor’s multipath routing into existing Tor infrastructure, thus making our deployed architecture usable and seamless.

6.1 Operator Liability

While we acknowledge that our proposed architecture may present challenges for relay operators, it is important to emphasize that our policies are aligned with and derived from those of Tor and CDN service providers such as Cloudflare (which we discuss in more detail in Appendix K). In most cases, service providers, including CDN and cloud providers, are not seen as the publishers or owners of information from their users and are not liable for the data stored, as long as they lack knowledge of illegal activities or information. Similarly, regarding damage claims, providers like AWS can be held responsible if they are informed of unlawful content but do not promptly remove it or disable access, since users can report objectionable content to service providers for action¹⁸.

In addition, the operation of a traditional Tor relay already comes with inherent risks that vary depending on the role it serves. The extent of the risk is a personal choice that each operator must make based on their individual circumstances, which is also the case with CenTor. In some regions, even basic traffic relaying may be fraught with legal and security implications that would deter potential operators. Despite these risks, many are still willing to contribute resources to the Tor network.

Hosting sensitive content. While we cannot necessarily answer the question for relay providers of willingness to deploy, part of our goals were to limit the risk as much as possible, as well as carefully design CenTor to also allow service operators to be able to adhere to similar policies and structures as discussed previously. The Bento framework and TEEs introduce an additional layer of security and control, where data is decrypted and processed exclusively within secure enclaves, ensuring that Bento servers are unable to either see or control the actual content. This isolation mechanism serves as a safeguard for operators’ liability and maintains data confidentiality. The use of Bento also allows operators to fine-tune the level of resources they wish to allocate, like storage and bandwidth, and even specify policies for exit nodes, giving operators flexibility to manage their involvement. Relay operators can additionally choose at any time to terminate either a specific replica (for example, given a take-down request) or their willingness to host at all.

¹⁷However, we are not bound to SGX and could utilize any TEE with similar properties.

¹⁸Interpretation and application of these laws may vary by jurisdiction and case specifics.

Interesting future work might be to develop a compromise mechanism of some sort, for example allowing an OS requesting hosting to prove (anonymously) its reputability so that each operator can make a decision. Work on oblivious computation in TEEs could help alleviate user concerns too. And even though TEEs already address many of the concerns raised, there is the additional interesting possibility of future work to ensure greater uncertainty as to what is running in them or, conversely, to solve the seemingly challenging problem of allowing an Onion Service requesting hosting to prove things (anonymously) about content it wishes to host.

7 Ethical Considerations

To test the performance of our techniques, we conducted many of our experiments on the live Tor network. We, therefore, deployed our own Tor relays, including the exit nodes, so that we would not harm the performance of the Tor network or affect the security and privacy of its users in any way. While our relays were not deployed as private, they had very limited uptime during each experiment, making it highly unlikely for regular users or routing algorithms to discover them. Their uptime was low enough that propagation delay in achieving consensus prevented them from becoming operational (which we monitored the logs to ensure). Moreover, we only logged our traffic. We deployed our own Bento servers that could act as onion service replicas. Finally, while we discuss various attacks on hidden services and how we can protect against them, we do not aim to deanonymize any onion service or its users in practice.

8 Conclusion

We have sought to explore ways to provide onion service operators with easy-to-deploy means of enhancing their users' experiences through performance improvements, as well as protecting themselves against common attacks. At the core of this effort was the development of CenTor, the first CDN for Tor onion services, which is flexible enough to support various different configurations and types of web services. We provide detailed quantitative anonymity analyses, demonstrating the performance vs. anonymity trade-offs and allowing individual users to customize their experience within the anonymity trilemma. We have implemented CenTor and evaluated it on the live and simulated Tor network, demonstrating concretely the performance benefits that it can bring.

Acknowledgments

Arushi Arora and Christina Garman's work was partially supported by NSF grant CNS-1816422.

References

- [1] Masoud Akhondji, Curtis Yu, and Harsha V Madhyastha. 2012. LASTor: A low-latency AS-aware Tor client. In *2012 IEEE Symposium on Security and Privacy*.
- [2] Md Washik Al Azad, Hasniuj Zahan, Sifat Ut Taki, and Spyridon Mastorakis. 2023. DarkHorse: A UDP-based Framework to Improve the Latency of Tor Onion Services. In *2023 IEEE 48th Conference on Local Computer Networks (LCN)*. IEEE, 1–6.
- [3] Mashael AlSabah, Kevin Bauer, Tariq Elahi, and Ian Goldberg. 2013. The path less travelled: Overcoming Tor's bottlenecks with traffic splitting. In *Privacy Enhancing Technologies Symposium (PETS)*.
- [4] Mashael AlSabah, Kevin Bauer, and Ian Goldberg. 2012. Enhancing Tor's performance using real-time traffic classification. In *ACM Conference on Computer and Communications Security (CCS)*.
- [5] Mashael AlSabah, Kevin Bauer, Ian Goldberg, Dirk Grunwald, Damon McCoy, Stefan Savage, and Geoffrey M Voelker. 2011. DefenestraTor: Throwing out windows in Tor. In *Privacy Enhancing Technologies Symposium (PETS)*.
- [6] Mashael AlSabah and Ian Goldberg. 2016. Performance and security improvements for tor: A survey. *ACM Computing Surveys (CSUR)* (2016).
- [7] Amazon. n.d. Amazon Web Services Agreement. <https://aws.amazon.com/agreement/> Last accessed: November 8, 2023.
- [8] Anonymous. 2023. Hidden Wiki Link Proposals (.onion address). http://zqkthwuiavvvqqt4ybvvgvi7tyo4hj5xgfuvpdf60tjiycgwbqbym2qad.onion/wiki/Talk:Main_Page#LINK_PROPOSALS.
- [9] asn. 2013. Hidden Services need some love. <https://blog.torproject.org/hidden-services-need-some-love>.
- [10] Armon Barton and Matthew Wright. 2016. DeNASA: Destination-Naive AS-Awareness in Anonymous Communications. *Proc. Priv. Enhancing Technol.* (2016).
- [11] Lamiaa Basyoni, Aiman Erbad, Mashael AlSabah, Noora Fetais, Amr Mohamed, and Mohsen Guizani. 2021. QuicTor: Enhancing Tor for Real-Time Communication Using QUIC Transport Protocol. *IEEE Access* (2021).
- [12] Lamiaa Basyoni, Aiman Erbad, Amr Mohamed, Ahmed Refaey, and Mohsen Guizani. 2020. Proportionally Fair approach for Tor's Circuits Scheduling. In *2020 International Symposium on Networks, Computers and Communications (ISNCC)*.
- [13] Lamiaa Basyoni, Noora Fetais, Aiman Erbad, Amr Mohamed, and Mohsen Guizani. 2020. Traffic Analysis Attacks on Tor: A Survey. In *IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*.
- [14] Alex Biryukov, Ivan Pustogarov, Fabrice Thill, and Ralf-Philipp Weinmann. 2014. Content and popularity analysis of Tor hidden services. In *2014 IEEE 34th International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE, 188–193.
- [15] Alex Biryukov, Ivan Pustogarov, and Ralf-Philipp Weinmann. 2013. Trawling for tor hidden services: Detection, measurement, deanonymization. In *IEEE Symposium on Security and Privacy*.
- [16] Yazan Boshmaf, Isuranga Perera, Udesh Kumarasinghe, Sajitha Liyanage, and Husam Al Jawaheri. 2023. Dizzy: Large-Scale Crawling and Analysis of Onion Services. In *Proceedings of the 18th International Conference on Availability, Reliability and Security*, 1–11.
- [17] Yérom-David Bromberg, Quentin Dufour, Davide Frey, and Etienne Rivière. 2022. Donar: Anonymous {VoIP} over Tor. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 249–265.
- [18] Cloudflare. 2023. Caching static and dynamic content. <https://www.cloudflare.com/learning/cdn/caching-static-and-dynamic-content/>.
- [19] Cloudflare. 2023. Cloudflare Comprehensive DDoS Protection. <https://www.cloudflare.com/ddos/>.
- [20] Cloudflare. n.d. Cloudflare Terms of Use. <https://www.cloudflare.com/website-terms/> Last accessed: November 8, 2023.
- [21] Intel Corporation. 2018. L1 Terminal Fault. <https://software.intel.com/content/www/us/en/develop/articles/software-security-guidance/advisory-guidance/l1-terminal-fault.html>.
- [22] Damian and The Tor Project. 2023. Stem: Python Controlled Library for Tor. <https://stem.torproject.org>.
- [23] Debajyoti Das, Sebastian Meiser, Efsandiari Mohammadi, and Aniket Kate. 2018. Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency-choose two. In *IEEE Symposium on Security and Privacy*.
- [24] Wladimir De la Cadena, Daniel Kaiser, Asya Mitseva, Andriy Panchenko, and Thomas Engel. 2019. Analysis of Multi-path Onion Routing-Based Anonymization Networks. In *IFIP Annual Conference on Data and Applications Security and Privacy*.
- [25] Wladimir De la Cadena, Daniel Kaiser, Andriy Panchenko, and Thomas Engel. 2020. Out-of-the-box Multipath TCP as a Tor Transport Protocol: Performance and Privacy Implications. In *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*.
- [26] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. 2002. Towards measuring anonymity. In *International Workshop on Privacy Enhancing Technologies*.
- [27] Roger Dingledine and Nick Mathewson. 2019. Tor Manual. <https://2019.www.torproject.org/docs/tor-manual.html.en>.
- [28] Roger Dingledine and Nick Mathewson. 2021. Tor Path Specifications. <https://github.com/torproject/torspec/blob/main/path-spec.txt>.
- [29] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. *Tor: The second-generation onion router*. Technical Report.
- [30] Christoph Döpmann, Valentin Franck, and Florian Tschorsch. 2021. Onion Pass: Token-Based Denial-of-Service Protection for Tor Onion Services. In *2021 IFIP Networking Conference (IFIP Networking)*. IEEE, 1–9.
- [31] Matthew Edman and Paul Syverson. 2009. AS-awareness in Tor path selection. In *Proceedings of the 16th ACM conference on Computer and communications security*.

- [32] Muhammad El-Hindi, Tobias Ziegler, Matthias Heinrich, Adrian Lutsch, Zheguang Zhao, and Carsten Binnig. 2022. Benchmarking the Second Generation of Intel SGX Hardware. In *Data Management on New Hardware*.
- [33] Kevin Gallagher, Sameer Patil, Brendan Dolan-Gavitt, Damon McCoy, and Nasir Memon. 2018. Peeling the Onion's User Experience Layer: Examining Naturalistic Use of the Tor Browser. In *ACM Conference on Computer and Communications Security (CCS)*.
- [34] Christina Garman, Matthew Green, and Ian Miers. 2013. Decentralized anonymous credentials. *Cryptology ePrint Archive* (2013).
- [35] Milad Ghaznavi, Elaheh Jalalpour, Mohammad A Salahuddin, Raouf Boutaba, Daniel Migault, and Stere Preda. 2021. Content Delivery Network Security: A Survey. *IEEE Communications Surveys & Tutorials* (2021).
- [36] Yossi Gilad, Amir Herzberg, Michael Sudkovitch, and Michael Goberman. 2016. CDN-on-Demand: An affordable DDoS Defense via Untrusted Clouds.. In *Network and Distributed System Security Symposium (NDSS)*.
- [37] David Goldschlag, Michael Reed, and Paul Syverson. 1999. Onion routing. *Commun. ACM* (1999).
- [38] Stephen Herwig, Christina Garman, and Dave Levin. 2020. Achieving Keyless CDNs with Conclaves. In *USENIX Security Symposium*.
- [39] Kyle Hogan, Sacha Servan-Schreiber, Zachary Newman, Ben Weintraub, Cristina Nita-Rotaru, and Srinivas Devadas. 2022. ShorTor: Improving Tor Network Latency via Multi-hop Overlay Routing. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1933–1952.
- [40] Diana L Huete Trujillo and Antonio Ruiz-Martinez. 2021. Tor hidden services: A systematic literature review. *Journal of Cybersecurity and Privacy* 1, 3 (2021), 496–518.
- [41] Alfonso Iacovazzi, Daniel Frassinelli, and Yuval Elovici. 2019. The {DUSTER} attack: Tor onion service attribution based on flow watermarking with track hiding. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*.
- [42] Government Info. 1998. Digital Millennium Copyright Act (DMCA). <https://www.govinfo.gov/content/pkg/PLAW-105publ304/pdf/PLAW-105publ304.pdf>.
- [43] Government Info. 2023. Section 230 of the Communications Decency Act. <https://www.govinfo.gov/content/pkg/USCODE-2021-title47/pdf/USCODE-2021-title47-chap5-subchapII-partI-sec230.pdf>.
- [44] Intel. 2024. Your Source for Software and Hardware Security Information. <https://www.intel.com/content/www/us/en/developer/topic-technology/software-security-guidance/overview.html>
- [45] ipinfo.io. 2023. ipinfo.io. <https://ipinfo.io> Last accessed: November 8, 2023.
- [46] Isabela. 2023. Tor is slow right now. Here is what is happening. <https://blog.torproject.org/tor-network-ddos-attack/>.
- [47] Rob Jansen and Nicholas J Hopper. 2011. Shadow: Running Tor in a box for accurate and efficient experimentation. (2011).
- [48] Rob Jansen, Justin Tracey, and Ian Goldberg. 2021. Once is Never Enough: Foundations for Sound Statistical Inference in Tor Network Experimentation. In *30th USENIX Security Symposium (Sec)*. See also <https://neverenough-sec2021.github.io>.
- [49] Rob Jansen, Tavish Vaidya, and Micah Sherr. 2019. Point break: a study of bandwidth denial-of-service attacks against tor. In *USENIX Security Symposium*.
- [50] Aaron Johnson, Rob Jansen, Aaron D Jaggard, Joan Feigenbaum, and Paul Syverson. 2015. Avoiding the man on the wire: Improving tor's security with trust-aware path selection. *arXiv preprint arXiv:1511.05453* (2015).
- [51] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. 2013. Users get routed: Traffic correlation on Tor by realistic adversaries. In *Proceedings of the ACM SIGSAC conference on Computer & communications security*.
- [52] Simon Johnson, Raghunandan Makaram, Amy Santoni, and Vinnie Scarlata. 2021. Supporting intel sgx on multi-socket platforms. Intel Corporation.
- [53] Simon Johnson, Vinnie Scarlata, Carlos Rozas, Ernie Brickell, and Frank McKeen. 2016. Intel Software Guard Extensions: EPID Provisioning and Attestation Services.
- [54] Joshua Juen. 2012. Protecting anonymity in the presence of autonomous system and Internet exchange level adversaries. (2012).
- [55] Ishan Karunanayake, Nadeem Ahmed, Robert Malaney, Rafiqul Islam, and Sanjay Jha. 2020. Anonymity with Tor: A Survey on Tor Attacks. *arXiv preprint arXiv:2009.13018* (2020).
- [56] Seongmin Kim, Juhyeng Han, Jaehyeong Ha, Taesoo Kim, and Dongsu Han. 2018. Sgx-tor: A secure and practical tor anonymity network with sgx enclaves. *IEEE/ACM Transactions on Networking* (2018).
- [57] Albert Kwon, Masha'al AlSabah, David Lazar, Marc Dacier, and Srinivas Devadas. 2015. Circuit fingerprinting attacks: Passive deanonymization of tor hidden services. In *USENIX Security Symposium*.
- [58] Olaf Landsiedel, Alexis Pimenidis, Klaus Wehrle, Heiko Niedermayer, and Georg Carle. 2007. Dynamic multipath onion routing in anonymous peer-to-peer overlay networks. In *IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference*.
- [59] Alejandro Buitrago López, Javier Pastor Galindo, and Félix Gómez Mármol. 2023. Exploring the availability, protocols and advertising of Tor v3 domains. In *2023 JNIC Cybersecurity Conference (JNIC)*. IEEE, 1–8.
- [60] Srdjan Matic, Platon Kotzias, and Juan Caballero. 2015. Caronte: Detecting location leaks for deanonymizing tor hidden services. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*.
- [61] Damon McCoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. 2008. Shining light in dark places: Understanding the Tor network. In *Privacy Enhancing Technologies Symposium (PETS)*.
- [62] Satyajayant Misra, Reza Tourani, and Nahid Ebrahimi Majd. 2013. Secure content delivery in information-centric networks: Design, implementation, and analyses. In *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*.
- [63] moxie0. 2017. Technology preview: Private contact discovery for Signal.
- [64] Rishab Nithyanand, Oleksii Starov, Adva Zair, Phillipa Gill, and Michael Schapira. 2015. Measuring and mitigating AS-level adversaries against Tor. *arXiv preprint arXiv:1505.05173* (2015).
- [65] Lasse Overlier and Paul Syverson. 2006. Locating hidden servers. In *IEEE Symposium on Security and Privacy*.
- [66] Lasse Overlier and Paul Syverson. 2006. Valet services: Improving hidden servers with a personal touch. In *International Workshop on Privacy Enhancing Technologies*.
- [67] Lasse Overlier and Paul Syverson. 2007. Improving efficiency and simplicity of Tor circuit establishment and hidden services. In *International Workshop on Privacy Enhancing Technologies*.
- [68] Andriy Panchenko, Asya Mitseva, Martin Henze, Fabian Lanze, Klaus Wehrle, and Thomas Engel. 2017. Analysis of fingerprinting techniques for tor hidden services. In *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society*.
- [69] The European Parliament and the Council of the European Union. 2000. Electronic Commerce Directive. <http://www.columbia.edu/~mr2651/e-commerce/2nd/statutes/ElectronicCommerceDirective.pdf> Last accessed: November 8, 2023.
- [70] Gang Peng. 2004. CDN: Content distribution network. *arXiv preprint cs/0411069* (2004).
- [71] Gavin Phillips. 2022. How to Find Active .Onion Dark Web Sites. <https://www.makeuseof.com/tag/find-active-onion-sites/>.
- [72] The Tor Project. 2023. The Legal FAQ For Tor Relay Operators. <https://community.torproject.org/relay/community-resources/eff-tor-legal-faq/>.
- [73] The Tor Project. 2023. Tor Metrics. <https://metrics.torproject.org>.
- [74] The Tor Project. 2023. Tor Project Consensus Parameters. <https://consensus-health.torproject.org/#consensusparams>.
- [75] The Tor Project. 2023. Why is the first IP address in my relay circuit always the same? <https://support.torproject.org/tbb/tbb-2/>.
- [76] Joel Reardon and Ian Goldberg. 2009. Improving Tor using a TCP-over-DTLS Tunnel. In *USENIX Security Symposium*.
- [77] Michael Reininger, Arushi Arora, Stephen Herwig, Nicholas Francino, Jayson Hurst, Christina Garman, and Dave Levin. 2021. Bento: Safely Bringing Network Function Virtualization to Tor. In *ACM SIGCOMM*.
- [78] Florentin Rochet and Tariq Elahi. 2022. Towards Flexible Anonymous Networks. *arXiv preprint arXiv:2203.03764* (2022).
- [79] Florentin Rochet, Ryan Wails, Aaron Johnson, Prateek Mittal, and Olivier Pereira. 2020. CLAPS: Client-Location-Aware Path Selection in Tor. In *ACM Conference on Computer and Communications Security (CCS)*.
- [80] Michael Rosenberg, Jacob White, Christina Garman, and Ian Miers. 2023. zk-creds: Flexible Anonymous Credentials from zkSNARKs and Existing Identity Infrastructure. In *IEEE Symposium on Security and Privacy*.
- [81] Max Schuchard, John Geddes, Christopher Thompson, and Nicholas Hopper. 2012. Routing around decoys. In *Proceedings of the 2012 ACM conference on Computer and communications security*.
- [82] Piyush Kumar Sharma, Shashwat Chaudhary, Nikhil Hassija, Mukulika Maity, and Sambuddho Chakravarty. 2020. The road not taken: re-thinking the feasibility of voice calling over tor. *arXiv preprint arXiv:2007.04673* (2020).
- [83] Robin Snader and Nikita Borisov. 2008. A Tune-up for Tor: Improving Security and Performance in the Tor Network.. In *Network and Distributed System Security Symposium (NDSS)*.
- [84] Robin Snader and Nikita Borisov. 2010. Improving security and performance in the Tor network through tunable path selection. *IEEE Transactions on Dependable and Secure Computing* (2010).
- [85] Martin Steinebach, Marcel Schäfer, Alexander Karakuz, Katharina Brandl, and York Yannikos. 2019. Detection and analysis of tor onion services. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*. 1–10.
- [86] Yixin Sun, Anne Edmundson, Nick Feamster, Mung Chiang, and Prateek Mittal. 2017. Counter-RAPTOR: Safeguarding Tor against active routing attacks. In *2017 IEEE Symposium on Security and Privacy (SP)*.
- [87] Yixin Sun, Anne Edmundson, Laurent Vanbever, Oscar Li, Jennifer Rexford, Mung Chiang, and Prateek Mittal. 2015. {RAPTOR}: Routing attacks on privacy in tor. In *24th USENIX Security Symposium (USENIX Security 15)*.
- [88] Can Tang and Ian Goldberg. 2010. An improved algorithm for Tor circuit scheduling. In *ACM Conference on Computer and Communications Security*

- (CCS).
- [89] Akamai Technologies. 2023. Akamai DDoS Protection. <https://www.akamai.com/us/en/resources/ddos-protection.jsp>.
 - [90] Tor. n.d.. Tor Path Specifications. <https://github.com/torproject/torspec/blob/main/path-spec.txt> Last accessed: November 8, 2023.
 - [91] Florian Tschorsch and Björn Scheuermann. 2016. Mind the gap: Towards a backpressure-based transport protocol for the Tor network. In *Symposium on Networked Systems Design and Implementation (NSDI)*.
 - [92] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F Wenisch, Yuval Yarom, and Raoul Strackx. 2018. Foreshadow: Extracting the keys to the intel {SGX} kingdom with transient out-of-order execution. In *{USENIX} Security Symposium*.
 - [93] S. van Schaik, A. Seto, T. Yurek, A. Batori, B. AlBassam, D. Genkin, A. Miller, E. Ronen, Y. Yarom, and C. Garman. 2024. SoK: SGX.Fail: How Stuff Gets eXposed. In *2024 IEEE Symposium on Security and Privacy (SP)*.
 - [94] Camilo Viecco. 2008. UDP-OR: A fair onion transport design. *Proceedings of Hot Topics in Privacy Enhancing Technologies (HOTPETS'08)* (2008).
 - [95] Tao Wang, Kevin Bauer, Clara Forero, and Ian Goldberg. 2012. Congestion-aware path selection for Tor. In *Financial Cryptography and Data Security: 16th International Conference, FC 2012, Kralendijk, Bonaire, February 27-March 2, 2012, Revised Selected Papers 16*. Springer, 98–113.
 - [96] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. 2014. Effective attacks and provable defenses for website fingerprinting. In *USENIX Security Symposium*.
 - [97] Wikipedia. 2022. The Hidden Wiki. https://en.wikipedia.org/wiki/The_Hidden_Wiki.
 - [98] Philipp Winter, Anne Edmundson, Laura M Roberts, Agnieszka Dutkowska-Żuk, Marshini Chetty, and Nick Feamster. 2018. How do Tor users interact with onion services?. In *USENIX Security Symposium*.
 - [99] SolarWinds Worldwide. 2023. Pingdom Website Speed Test. <https://tools.pingdom.com>.
 - [100] SolarWinds Worldwide. 2023. Solarwinds pingdom. <https://www.pingdom.com/blog/webpages-are-getting-larger-every-year-and-heres-why-it-matters/>.
 - [101] Lei Yang and Fengjun Li. 2015. Enhancing traffic analysis resistance for tor hidden services with multipath routing. In *International Conference on Security and Privacy in Communication Systems*.
 - [102] Lei Yang and Fengjun Li. 2015. mtor: A multipath tor routing beyond bandwidth throttling. In *2015 IEEE Conference on Communications and Network Security (CNS)*.

Appendices

A Detailed Discussion of α_{CenTor}

We now provide an extended explanation of some of the computations in α_{CenTor} , as well as why we consider all parameters in the score to be equally significant to maintain sufficient anonymity while achieving performance gains.

Based on information theory, the concept of entropy delivers a measure of the information possessed in a system [26]. Since we desire that our adversary gains minimum information by observing the system (a chosen shadow S), the value of randomness in the system should be higher, implying that Tor users should not be easily distinguishable. We can obtain this by having high Tor users in a particular S . We, therefore, choose the parameters $EL(S)$ and $EC(S)$, which provide entropy distributions of clients across ASNs and countries, respectively. Based on this entropy, we can then calculate the degree of anonymity with respect to the number of countries and ASNs in S . If these values of user (or client) entropy in S are small, it means that the chosen S is not providing a high degree of anonymity. It immediately follows that the value of CD should be lower than as well.

We now motivate our rationale for relay density and why this is important to consider as part of α_{CenTor} . One of the primary means of compromising a client's anonymity is the case where an adversary can control both the guard and exit relays used [15]. If the adversary can control more than a given threshold of Tor relays,

it can simply observe more traffic on the network, and, therefore, the value of $RD(S)$ should be higher for a given S .

To help overcome the possibilities of traffic analysis attacks, Tor creates a small set of guard relays (three by default) per client for a long duration. Typically, a client uses the same guard relay for 60-90 days [75]. It is, therefore, equally important to have a substantial number of Tor relays present in S for a client to choose from and evade deanonymization attacks such as correlation and guard placement attacks. Based on the work of [102], this probability of compromise for a 3-hop circuit can be expressed as:

$$P_{\text{compromised}} = \sum_{a=0}^g P(a) \cdot P(\text{compromised}|a)$$

where g is the guard set, a represents the relays controlled by the adversary, $P(a)$ represents the probability of g containing a malicious relays, and $P(\text{compromised}|a)$ is the probability of a client being compromised if there are a malicious relays. $P(a)$ and $P(\text{compromised}|a)$ are defined as follows:

$$P(a) = \binom{g}{a} \cdot f_{gbw}^a \cdot (1 - f_{gbw})^{g-a}$$

where f_{gbw} and f_{xbw} represent the fraction of bandwidth of adversary controlled guards and exits (with respect to total bandwidth of guards and exits) respectively, and

$$P(\text{compromised}|a) = f_{xbw} \cdot \left[1 - \frac{g-a}{g}\right]$$

To understand this better, assume that there are e and x numbers of malicious entry and exit relays, respectively. Intuitively, if we choose an S (as listed in Table 2), the probability of correlation attacks [51] is higher since the number of Tor relays (including guard and exit relays) in S is inherently reduced. In a worst-case scenario, say, all e and x may lie in the chosen S . For instance, we pick $e = 732$ and $x = 278$, which is 20% of total entry (3664) and exit (1398) relays.

As an example, we now calculate $P_{\text{compromised}}$ in case of an adversary performing a correlation attack for two concrete examples. For $S = \text{Central and Eastern Europe}$ (a case where the set of guard and exit relays is relatively small), our chosen values e and x result in $f_{gbw} = 0.571$ and $f_{xbw} = 0.815$ resulting in $P_{\text{compromised}} = 0.465$. Similarly, for $S = \text{Eurasia}$ (a case where the set of guard and exit relays is relatively large), $f_{gbw} = 0.252$ and $f_{xbw} = 0.268$, resulting in $P_{\text{compromised}} = 0.067$. We can clearly witness that an S that contains higher $ED(S)$ and $XD(S)$ provides a lower value of $P_{\text{compromised}}$, and therefore better anonymity to an CenTor user.

This means that a diverse set of Tor relays is crucial when selecting a shadow, making the parameters $RD(S)$, $ED(S)$, and $XD(S)$ equally significant while computing α_{CenTor} .

B μ Tor: Enhancing CenTor through multipath routing

We now provide a detailed discussion of the μ Tor implementation and performance and anonymity analysis, as well as a discussion of related work. Our approach builds upon previous research [102] that supports distributing bulk traffic among multiple low-bandwidth relays which are often overlooked and underutilized (as per the Tor relay selection protocol [95]). This strategy effectively reduces the risk of congestion on Tor relays, also reducing the load on high-bandwidth relays.

Related Work. Multipath routing for Tor essentially channels multiple, parallel Tor circuits. Clients and servers split traffic across these multiple circuits to achieve greater throughput [102]

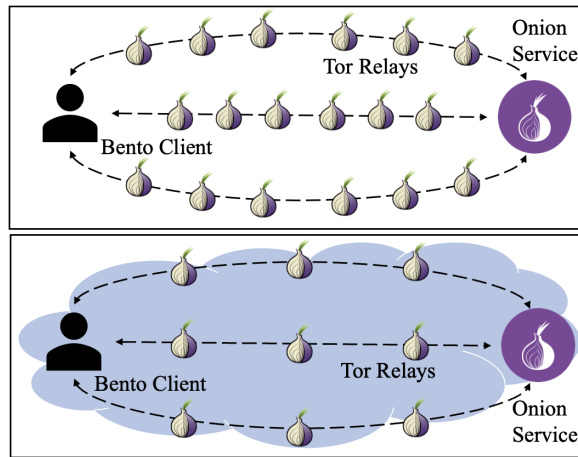


Figure 10: Architectural comparison of the standard 3Tor architecture versus the 3Tor architecture composed with location-aware CenTor.

and improved interactivity [3, 94], similar in spirit to MPTCP [25]. Of particular note, Yang et al. [102] introduced *mTor*, a self-adaptive multi-path Tor routing algorithm that avoids high-bandwidth relays and distributes the load on low-bandwidth ones, thus benefiting bandwidth-intensive as well as latency-sensitive applications. They later extended their architecture for onion services, with the primary goal of guarding against traffic analysis attacks [101]. Unfortunately, their architecture not only modifies Tor’s protocols and source code but adds auxiliary circuits, i.e., extra middle node(s), on the server-side of the 6-hop circuit between the client and the onion service. Landside et al. [58] proposed a dynamic Multipath Onion Router (MORE) for peer-to-peer overlay networks, requiring client participation as an Onion Router.

B.1 Design and Implementation

μ Tor aims to improve the performance of bandwidth-intensive applications that utilize Tor onion services by distributing the bandwidth load across multiple circuits.

The Tor relay selection process for constructing circuits prefers high-bandwidth relays, which implies that many low-bandwidth relays are under-utilized and mostly idle [102]. Therefore, we propose a framework that aims to use these relays for bandwidth-intensive applications without further clogging high-bandwidth relays and contending with latency-sensitive applications. In other words, we utilize low-bandwidth relays to construct multiple paths between the client and the onion service¹⁹. Moreover, by reducing the load on the high-bandwidth relays, μ Tor may also benefit latency-sensitive applications by freeing up overall capacity [3].

Like CenTor, μ Tor is realized as a function that can be deployed in the Bento architecture. The function preemptively creates μ paths²⁰, preferring low-bandwidth relays, using the Stem library [22]. The μ Tor function also takes an input β from the user,

¹⁹We note that our design is not exclusive to these relays though and any can be utilized in practice.

²⁰This parameter can be specified by each individual user.

which defines the maximum number of bytes that the function can fetch per path. The function generates a thread which creates a path and then sends a sub-request to the onion service to fetch β bytes. This thread creation happens sequentially. Every new thread selects a new path to fetch the next β bytes. These bytes are received in parallel on the client-side, along with a sequence number. Based on the sequence number, the received data is then assembled. Each circuit is tagged “dirty” after the data is received by the client or the TCP connection is closed. The circuit’s endpoints (the client and the onion service) are responsible for splitting the traffic at one endpoint and buffering, re-ordering, and delivering in-order cells to the application at the other end of the circuit. Because we do not change the overall interaction protocol between the client and onion service in any way, this approach could be deployed right away without any modifications to the Tor code, using the Bento architecture. Figure 10 showcases the structure of the client and onion service’s communication at a high level using what we refer to as 3Tor (which means traffic is split over three paths) in the case of a legacy onion service (6-hop) versus its composition with location-aware CenTor (3-hop). Utilizing μ Tor with the location-aware replicas in our CDN allows us to not only improve bandwidth but also reduce latency by cutting down on the number of hops.

Is this Secure? While we have discussed how it is relatively easy to deploy μ Tor as a function, we again must ask ourselves: is this secure? What impact might this have on an onion service or a user? One immediately obvious concern is that this might make it easier to perform a DoS attack on an onion service, as we have now made it easy for a client to open multiple connections to it. Tor does provide DoS mitigation techniques for Tor relays against an adversary who can attempt to make a huge number of concurrent connections. For instance, Tor allows a maximum of three concurrent connections to an onion service²¹ and the formulation of at least three circuits²², by default. Further, relay operators can manually customize these values in *torrc* options (*DoSCircuitCreationMinConnectionsNUM*, *NewCircuitPeriodNUM*) [27] to provide μ Tor users with more adaptability, or may choose to operate Tor’s default consensus parameters [74] to achieve better DoS resistance. Additionally, we could enforce thresholds for β so that the client does not potentially create a large number of circuits.

To answer the question about impact on the user, we provide an in-depth quantitative analysis of the anonymity impacts of utilizing μ Tor in different configurations in Appendix B.2. To summarize, as one might intuitively expect, the greater the number of paths used by μ Tor, the higher the possibility of a correlation attack. If users wish to strike a more equal balance between anonymity loss and bandwidth benefits they could, for example, choose to use a common guard node for their multipath setup. Our detailed analysis provides the interested user with a more in-depth understanding of these trade-offs though, so that they can make a decision themselves about their desired parameters.

²¹As mentioned in DoS mitigation options in Tor Manual[27].

²²On startup Tor tries to maintain one clean fast exit circuit that allows connections to port 80, and at least two fast clean stable internal circuits in case we get a resolve request or hidden service requests[90].

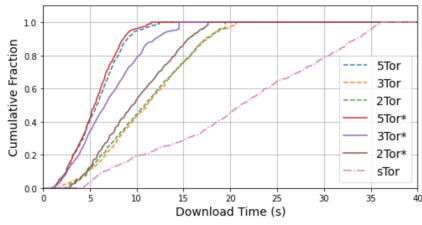


Figure 11: Performance comparison for single versus unique guard node(s) per μ circuits.

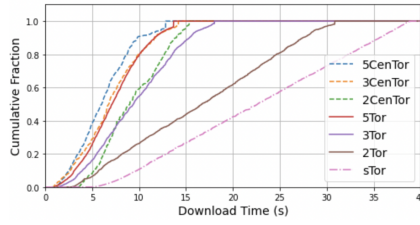


Figure 12: Performance comparison for standalone μ Tor and the composition of CenTor and μ Tor.

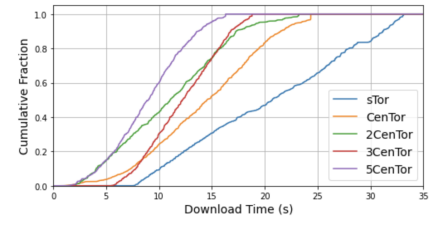


Figure 13: Performance comparison of standalone CenTor with the integration of CenTor and μ Tor.

B.2 Detailed μ Tor Anonymity Analysis

We discuss the in-depth potential anonymity risks that may be introduced due to the construction of multiple circuits from the client to the server, as well as analyze the anonymity provided to clients using μ Tor, in more detail.

Anonymity versus Performance Trade-off. As discussed previously, an adversary can perform correlation attacks by controlling both the guard and exit relays of a Tor circuit that the client uses, thereby compromising the client's anonymity. When a client interacts with an onion service through a Bento server by executing μ Tor, the adversary needs to be able to control three²³ entry relays to perform such attacks. Moreover, the probability of performing a successful correlation increases when a relay in the guard set is controlled by the adversary and even more so when the size of the guard set is increased (which happens in the case of μ Tor). When a client is interacting with an onion service, the adversary generally needs to be able to control both entry relays (of the client as well as the onion service) to effectively perform a correlation or traffic analysis attack [57]. In this case the $P(\text{compromised}|a)$ can be defined as

$$P(\text{compromised}|a) = f_{xbw} \cdot \left[1 - \frac{(g-\mu)}{\mu} \right]$$

where μ represents the number of paths used by μ Tor (other parameters are discussed in Appendix A). Also, $\mu \geq g$ implies that μ paths will use up all entry relays in the guard set thereby increasing the correlation possibility. To successfully carry out a 6-hop correlation attack in the case of an onion service, an adversary needs to control both the guard and exit node in S (the server-side circuit) and only the guard node in C (the client-side circuit).

Therefore, we can calculate the joint probability of compromise of an onion service as

$$P_{\text{compromiseOS}}(C, S) = P_{\text{compromiseOS}}(S) \cdot P_{\text{compromiseOS}}(C|S)$$

where $P_{\text{compromiseOS}}(S)$ is equivalent to $P_{\text{compromised}}$. Further, $P_{\text{compromiseOS}}(C|S)$ depicts the case wherein S is already compromised implying that it is also equivalent to $P_{\text{compromised}}$ with $f_{xbw} = 1$.

To analyze the correlation risk of μ Tor in practice, we consider that the adversary controls about 20% of the total Tor exit relays [102]. Figure 14 shows a comparison between the probability of compromise in the μ Tor architecture for 6-hop legacy onion services

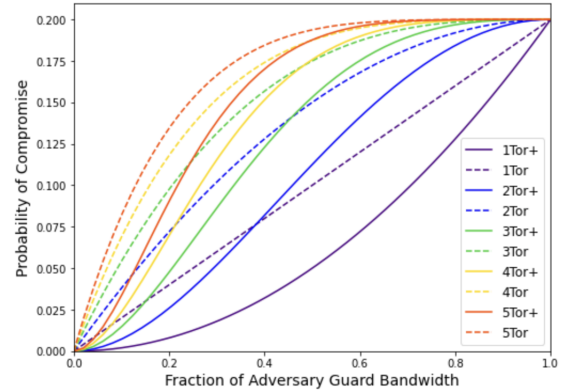


Figure 14: Probability of compromise in the μ Tor architecture. The + indicates a legacy onion service.

(denoted by μ Tor+) versus 3-hop μ Tor composed with location-aware CenTor (or for standard 3-hop non-anonymous websites). As one might intuitively expect, this figure shows that the greater the number of paths used by the μ Tor architecture, the higher the possibility of a correlation attack, thus demonstrating both the importance of the choice of μ to balance performance with anonymity, as well as why such a quantitative analysis is necessary to help users make informed parameter choices based on their own comfort level. Additionally, we can see that the probability of compromise in the case of a 6-hop circuit is lower than a 3-hop circuit, implying the (somewhat expected) trade-off between anonymity and latency. Based on our analysis, we agree with Tor recommendations and suggest, even with μ Tor, users utilize three relays in the set of their guard nodes. Additionally, μ Tor users can choose to use a common guard node for their multipath routing architecture to achieve better anonymity along with somewhat improved bandwidth benefits.

It is important to note as well that the probability of compromise when μ Tor is combined with CenTor is same as the probability of compromise in a 3-hop circuit, denoted as $P_{\text{compromised}}$, where the parameter g now pertains to the guard nodes in the respective shadow. This implies that as the value of g decreases within a shadow, $P_{\text{compromised}}$ would increase.

²³The entry for the connection to Bento, the connection from the Bento function, and the connection from onion service

B.3 Evaluation

To understand the performance benefits of μ Tor we analyze it over the Tor network, varying μ , i.e., the number of paths between the client and the onion service host, in different scenarios. Our experimental setup is identical to that used in Section 5. We start by independently testing the performance with a common guard node (represented by μ Tor) and unique guard nodes (represented by μ Tor*) for a variety of different values of μ and present the results in Figure 11. The results are generally what one would intuitively expect. We observe that higher values of μ , in general, provide lower download time. Also, μ Tor performs significantly better than standard Tor (*sTor*). Further, we see that μ Tor* performs better than its coequal μ Tor since a common guard node, while increasing security and decreasing the chance of deanonymization attacks, can prove to be a performance bottleneck for merging traffic. Next, we compare the performance benefits of standalone μ Tor with bringing location-awareness to it through CenTor (μ CenTor) and present the results in Figure 12. This composition provides the benefit of both multipath routing and reduced hops between the client and replica. As expected, we conclude that 3-hop μ CenTor performs better than its 6-hop peer μ Tor (and significantly better than standard Tor), for all values of μ . We observe that 3CenTor provides about 63.15% reduction in download time and 5CenTor 68.4% over standard Tor. Finally, we demonstrate how client-side use of μ Tor can further enhance the previously discussed performance gains of CenTor in Figure 13. For instance, by introducing two multi-paths and five multi-paths to the use of CenTor, a client can achieve about 4% and 33% additional reduction in download time, respectively.

Client Computational Cost. An additional concern that a μ Tor client might have is increased computational burden from maintaining multiple circuits just to connect to a single service. However, the Tor protocol already ensures that a client maintains a minimum number of (at least three and at most twelve by default) clean circuits [28] and also preemptively builds spare circuits based on client usage. And our design only utilizes the existing Tor protocol for circuit construction and communication. Therefore, we argue that a client who desires to run μ Tor is not subject to added computation overhead beyond what they might normally experience.

C CenTor Anonymity-Performance Trade-off: TTFB

In this experiment (result shown in Figure 15), we selected three weak shadows with $\alpha_{CenTor} < 0$: *United States*, *North America*, and *Americas*; a strong shadow with $\alpha_{CenTor} = 0.418$: *Western Europe + Americas*, and two perfect shadows with $\alpha_{CenTor} = 1$: *Location Unaware CenTor* and *Tor* (defined in Appendix J), which allows us to directly explore the anonymity-performance trade-off. The client, as well as the onion service host, are both located in the United States. For the evaluation, we simply increased the shadow size (therefore increasing the client’s anonymity) and measured the client-side TTFB. As expected, the *United States* (weak) shadow provides the best performance, as it restricts client-side circuit nodes to be within the United States, thereby decreasing the geographic distance between hops. Client-side performance degrades, and anonymity increases, as the shadow size expands (and the geographic distance between circuit nodes potentially increases). It is

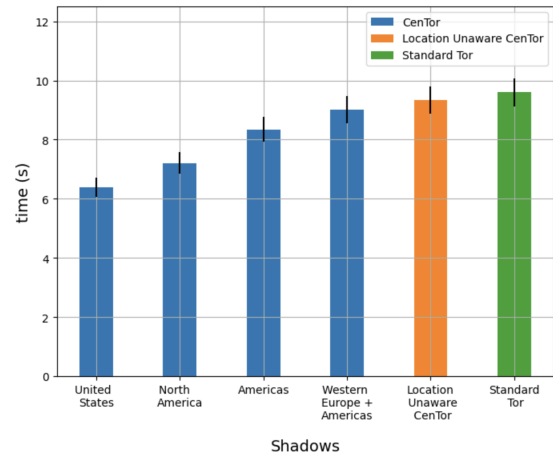


Figure 15: Anonymity vs Performance (time-to-first-byte) comparison: CenTor with different shadows, location unaware CenTor, and standard Tor.

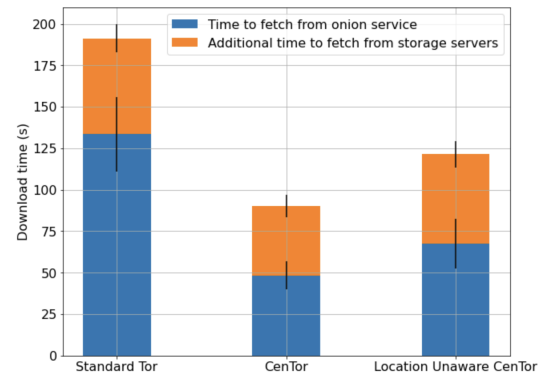


Figure 16: Download times (sec) for fetching dynamic content from a storage server.

worth noting that *Location Unaware CenTor* has a higher TTFB than regular CenTor. Intuitively, this is due to the fact that client-side circuit nodes in the case of *Location Unaware CenTor* do not have a region-based (shadow) restriction and are therefore distributed all over the globe. In other words, the performance for this case can be somewhat variable, depending on if all client-side Tor nodes are picked within a cluster or more sparse and distributed. CenTor and *Location Unaware CenTor* both perform better than standard Tor, simply because the latter case increases the number of hops from 3 to 6 and chooses nodes (somewhat) randomly from all over the globe.

D Experiment: CenTor Dynamic Content

In this experiment, we have a client download a 20MB file from an onion service, which in turn dynamically fetches this file from a storage server, in different scenarios. All entities (the client, Bento server and storage server) are located within the United States. We present the results in Figure 16.

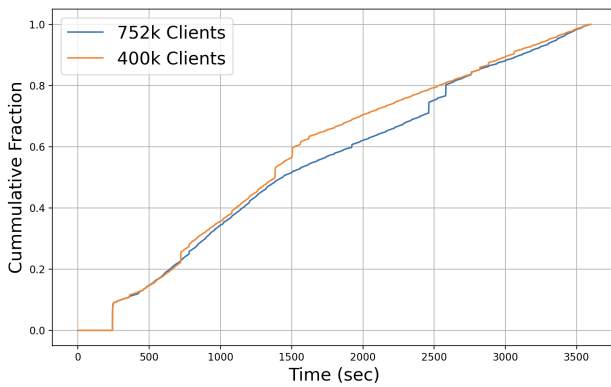


Figure 17: Performance comparison of Tor clients in different instances of client loads.

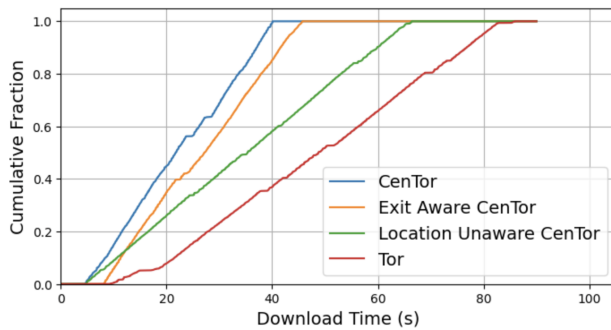


Figure 18: Performance comparison of Exit-Aware CenTor client.

We first note that the time to (dynamically) fetch data from the storage server is similar in all scenarios, as this always happens on a single Tor circuit. As expected, we can see that dynamically fetching the file using location-unaware CenTor has performance quite similar to just fetching it from the onion service using the standard Tor protocol (as the number of Tor hops will be similar and in varying locations). If we then use regular CenTor we see a significant decrease in download time, even with dynamic content, as it is much faster to access the replica and initiate the fetching of dynamic content. In either scenario, we again see how our CDN provides significant performance improvements over standard Tor.

E Shadow Baseline Simulation

This experiment focused on assessing the impact of variable client loads, spanning from 400,000 to 752,000 clients. As the client count increased incrementally, the network experienced escalating stress levels, which led to a decline in client performance. For instance, in a simulated network with 752,000 clients, we observed a performance degradation of approximately 8.1%, compared to the baseline with 400,000 clients at the 2000-second mark (as shown in Figure 17).

F Exit-Aware Routing

Another dimension worth exploring, which allows for enhanced anonymity compared to a typical CenTor client, while still delivering some performance benefits, involves introducing location-awareness exclusively to the exit node on the client-side circuit. In this scenario, the client is limited to using an exit node within the same shadow as the CenTor instance. The experimental configuration is the same as the one described in Section 5, where both the onion service and the client reside in the same shadow (U.S.). For the 3-hop CenTor case, the relays are also within the shadow. For exit-aware CenTor, only the exit node is in the shadow, while for location-unaware CenTor, the relays are globally distributed. In the case of Tor, there are the additional 3-hops (total 6-hops) to reach the onion service (anonymous-mode).

In this case, the client does not need to factor in the RD and ED parameters when computing α_{CenTor} . As depicted in Figure 18, the performance in this case falls between that of a location-unaware and a fully location-aware CenTor configuration.

G Specifics of the Anonymity Metric and Recommendations for Operators and Clients

Acknowledging the complexity faced by service operators and clients in selecting optimal replicas and shadows in practice, we offer basic (safe) starting recommendations to streamline this process.

We start by providing additional information to help better understanding the anonymity metric at a high level. The anonymity metric helps users compare various shadows and understand how much anonymity they are compromising. In the Table 4, we classify the scores and discuss what they mean at a high level in practice, to help guide client understanding.

Service operators. For service operators, diversifying relay choices is advisable to broaden client reach. Deploying replicas in countries associated with each shadow and incorporating multiple replicas with an option for a load balancer²⁴ in high client density ($CD(S)$) regions (strong shadows as shown in Table 5) is recommended. Table 3 outlines a selection of shadows where operators are advised to deploy a minimum of one replica to begin with.

CenTor Clients. As discussed in Section 4.2, clients should prioritize choosing shadows with a high α_{CenTor} score. We define *weak* shadows as those with an α_{CenTor} score below 0 and *borderline* shadows with an α_{CenTor} score $0 - 0.1$, meaning they meet at least the minimum anonymity requirements described in Section 4.1 (although this “weak” score may indicate potential negative values for specific parameters). In such instances, we strongly advise CenTor users to examine each parameter for adequate anonymity levels.

Optimal choices include **strong** shadows with α_{CenTor} scores above 0.1, indicating they surpass our minimum described requirements. We recommend clients use strong shadows as listed in Table 5 as a starting point. Clients have the flexibility to customize, build upon, or strengthen shadows by selecting other strong shadows to incorporate into their chosen potential weak shadows. For instance, clients can strengthen shadow *Americas*, a weak shadow with $\alpha_{CenTor} = -0.044$, by adding *Western Europe* to it (shadow

²⁴Such as the hidden service load balancer in [77].

Table 3: Shadow specifications

Shadow	List of Countries
Africa	South Africa, Nigeria, Tanzania, Kenya, Democratic Republic of the Congo, Egypt, Uganda, Algeria, Sudan, Morocco, Angola, Mozambique, Madagascar, Ghana, Cameroon, Niger, Mali, Malawi, Zambia, Senegal, Chad, Somalia, Zimbabwe, Guinea, Rwanda, Benin, Tunisia, Togo, Botswana, Seychelles
Africa & Europe	South Africa, Nigeria, Tanzania, Kenya, Democratic Republic of the Congo, Egypt, Uganda, Algeria, Sudan, Morocco, Angola, Mozambique, Madagascar, Ghana, Cameroon, Niger, Mali, Malawi, Zambia, Senegal, Chad, Somalia, Zimbabwe, Guinea, Rwanda, Benin, Tunisia, Togo, Botswana, Seychelles, Cyprus, Georgia, Kazakhstan, Turkey, Azerbaijan, Ukraine, Russia, Germany, United Kingdom, France, Italy, Spain, Ukraine, Poland, Romania, The Netherlands, Belgium, Czech Republic, Greece, Portugal, Sweden, Hungary, Belarus, Austria, Serbia, Switzerland, Bulgaria, Denmark, Finland, Slovakia, Norway, Ireland, Croatia, Moldova, Bosnia and Herzegovina, Albania, Lithuania, Macedonia, Slovenia, Latvia, Estonia, Montenegro, Luxembourg, Malta, Iceland, Andorra, Monaco, Liechtenstein
Western Europe	Germany, Austria, Belgium, France, Liechtenstein, Luxembourg, Monaco, The Netherlands, Switzerland
Central & Eastern Europe	Estonia, Germany, Latvia, Lithuania, Poland, Czech Republic, Slovakia, Hungary, Romania, Bulgaria, Slovenia, Croatia, Albania, Montenegro, Serbia, Macedonia, Bosnia and Herzegovina
Eurasia	United Arab Emirates, Hong Kong, Cyprus, Georgia, Kazakhstan, Turkey, Azerbaijan, Ukraine, Russia, Germany, United Kingdom, France, Italy, Spain, Poland, Romania, The Netherlands, Belgium, Czech Republic, Greece, Portugal, Sweden, Hungary, Belarus, Austria, Serbia, Switzerland, Bulgaria, Denmark, Finland, Slovakia, Norway, Ireland, Croatia, Moldova, Bosnia and Herzegovina, Albania, Lithuania, Macedonia, Slovenia, Latvia, Estonia, Montenegro, Luxembourg, Malta, Iceland, Andorra, Monaco, Liechtenstein, China, India, Indonesia, Pakistan, Bangladesh, Japan, Philippines, Vietnam, Thailand, Burma, South Korea, Malaysia, Nepal, Taiwan, Sri Lanka, Cambodia, Laos, Singapore, Mongolia, Bhutan, Maldives
Europe	Cyprus, Georgia, Kazakhstan, Turkey, Azerbaijan, Ukraine, Russia, Germany, France, Italy, Spain, Ukraine, Poland, Romania, The Netherlands, Belgium, Czech Republic, Greece, Portugal, Sweden, Hungary, Belarus, Austria, Serbia, Switzerland, Bulgaria, Denmark, Finland, Slovakia, Norway, Ireland, Croatia, Moldova, Bosnia and Herzegovina, Albania, Lithuania, Macedonia, Slovenia, Latvia, Estonia, Montenegro, Luxembourg, Malta, Iceland, Andorra, Monaco, Liechtenstein
APAC	United Arab Emirates, Hong Kong, China, India, Australia, New Zealand, Russia, Indonesia, Pakistan, Bangladesh, Japan, Philippines, Vietnam, Thailand, Burma, South Korea, Malaysia, Nepal, Taiwan, Sri Lanka, Cambodia, Laos, Singapore, Mongolia, Bhutan, Maldives
Americas	Belize, United States of America, Canada, US Virgin Islands, Costa Rica, Dominican Republic, El Salvador, Venezuela, Guatemala, Honduras, Mexico, Panama, Puerto Rico, Colombia, Brazil, Argentina, Bolivia, Chile, Ecuador, Paraguay, Peru, Uruguay
Americas & Western Europe	Belize, United States of America, Canada, US Virgin Islands, Costa Rica, Dominican Republic, El Salvador, Venezuela, Guatemala, Honduras, Mexico, Panama, Puerto Rico, Colombia, Brazil, Argentina, Bolivia, Chile, Ecuador, Paraguay, Peru, Uruguay, Germany, Austria, Belgium, France, Liechtenstein, Luxembourg, Monaco, The Netherlands, Switzerland
North America	United States, Canada, Belize, Costa Rica, Dominican Republic, El Salvador, Guatemala, Mexico
South America	Argentina, Brazil, Chile, Ecuador, Paraguay, Peru, Uruguay, Venezuela

Table 4: Understanding α_{CenTor} scores. High level discussion of various α_{CenTor} scores and what they mean in practice.

α_{CenTor} Score	Recommendation	Description	Example Shadows (described in Table 3)
< 0: weak shadow	Not recommended	The score does not match our minimum requirements and does not ensure that client anonymity is met.	Americas, Africa
0 – 0.1 (approx): borderline shadow	Use with caution	Values close to 0 indicate that certain individual parameters have negative values. Therefore, we strongly recommend that CenTor users inspect each parameter for their needs.	APAC, Americas
> 0.1: strong shadow	Recommended	The score is above our minimum requirements and ensures client anonymity is met, with larger numbers indicating greater anonymity.	Africa & Europe, Western Europe, Eurasia, Europe, Americas & Western Europe

Table 5: Strong shadows with positive α_{CenTor} .

Shadow	$\overline{EL}(S)$	$\overline{EC}(S)$	$\overline{CD}(S)$	$\overline{ED}(S)$	$\overline{XD}(S)$	$\overline{RD}(S)$	$\alpha_{CenTor}(S)$
Africa & Europe	0.677	0.610	0.475	0.671	0.364	0.591	0.564
Western Europe	0.485	0.586	0.124	0.396	0.151	0.330	0.345
Eurasia	0.830	0.694	0.580	0.677	0.389	0.641	0.635
Europe	0.630	0.581	0.673	0.671	0.364	0.587	0.584
Americas & Western Europe	0.523	0.491	0.435	0.466	0.193	0.403	0.418

Americas and Western Europe increases α_{CenTor} to 0.418) as shown in Table 2. Similarly, a weak shadow *Africa* can be strengthened by adding *Europe*, as shown in Table 2.

Table 5 provides a list of concrete “strong” shadows for clients. Clients are advised to choose the smallest possible shadow that is geographically close to them from this list, as a basic starting guideline, in order to maximize the performance vs. anonymity trade-off. For example, based on the α_{CenTor} score, users in Western Europe can simply use the *Western Europe* shadow to access a replica (in the same region), safely achieving high performance benefits (since the content is geographically closer to them) without compromising significantly on anonymity.

H DoS Against Onion Services

The design of Tor's onion services is vulnerable to DoS attacks, allowing attackers to deplete their resources with minimal effort [30]. DoS attacks on onion services exploit the protocol's requirements for significant computational and networking resources to establish a connection. Attackers can cause disproportionate resource usage on the service side with minimal effort, leading to potential service disruption and unavailability [46].

Upon receipt of an *INTRODUCE2* cell (Step 6 in Figure 1), the onion service undertakes two key actions: first, it constructs a circuit to the nominated Rendezvous Point (Step 7), consuming network and computational resources. Additionally, this cell initiates the initial phase of a cryptographic handshake, necessitating the onion service to perform resource-intensive asymmetric cryptographic operations. Attackers can exploit this by selecting any relay as the RP via an *INTRODUCE1* cell (Step 5), potentially embedding false information, and opting for a direct, shorter circuit to further minimize their own resource usage.

If an attacker initiates a sufficient number of circuits, it can lead to the onion service becoming inoperative due to the depletion of its resources.

Storage DoS attacks (on the Bento servers themselves) are currently out of scope for the Bento architecture, which does not currently have support for or prevent such attacks, or authenticate users. One potential solution could involve implementing (decentralized) anonymous credentials [34, 80] to privately and anonymously authenticate users, and thus limit the number of services. Additionally, each Bento operator can specify a maximum storage allocation for each function (though their middlebox node policy and other mechanisms). By rate-limiting the number of *CenTor* instances (along with their allowed storage), this approach can effectively cap the maximum storage each user, whether malicious or not, can consume.

I State of Tor Onion Services

Tor hidden services exhibit high volatility, with sites frequently becoming unavailable or shutting down, which complicates efforts to categorize and analyze their content consistently [40, 59]. As such, better understanding this space presents a significant challenge and an opportunity for future research. The content on onion services spans a broad spectrum, containing both static and interactive elements. Marketplaces, social media, cryptocurrency platforms, and news outlets typically offer dynamic content with real-time

updates. Conversely, repositories for books, educational materials, and personal blogs tend to be more static, although they may feature interactive components like comment sections [14, 16, 85].

J Shadow Specifications

This appendix provides a full description of the countries selected in a specific shadow in Table 3. Approximate α_{CenTor} scores for these shadows are provided in Table 2 in the main body.

K Legal Policies

We now summarize various legal and content policies for Tor, CDN, and other service/hosting providers in a bit more detail, based on their stated user agreements and other publicly available information.

Tor Policy [72]. Tor's core mission is to safeguard free expression, privacy, and human rights. Tor's developers provide technical support but cannot offer legal advice or prevent illicit use of Tor relays. Users should know that their communications with Tor's developers regarding relay-related matters are not legally privileged and may be accessible to authorities or litigants.

While Tor cannot guarantee users' legal immunity, the Electronic Frontier Foundation (EFF) advocates strongly for the protection of relay operators from liability related to traffic passing through their relays. EFF operates its own relay and can assist relay operators in evaluating their legal circumstances, connecting them with qualified legal counsel when necessary. Tor relay operators should avoid monitoring, logging, or disclosing Tor users' communications, as this can result in legal consequences. Always consult a lawyer before examining anyone's communications. The trustworthiness of relay operators listed in the directory cannot be guaranteed by Tor's developers or EFF.

Service Provider Policies. In most cases, service providers, including cloud providers, are not considered the publishers or speakers of information provided by other content providers. Essentially, a service provider is not held liable for information stored at the request of the service's users, as long as the provider has no actual knowledge of illegal activities or information.

For instance, AWS as per their user agreements, emphasize user responsibility for content compliance with policies and laws. AWS and its affiliates provide services and content “as is” without warranties, making no specific commitments regarding standards, reliability, content, or services [7].

Laws exist to safeguard service providers in such circumstances. Section 230 of the Communications Decency Act in the United States grants online platforms immunity from civil liabilities related to third-party content and content removal under certain conditions, emphasizing that service providers are not considered publishers or speakers of information from others [43]. The Digital Millennium Copyright Act (DMCA) provides limited liability to service providers for copyright infringement if they follow DMCA requirements, particularly in the “safe harbor” provisions [42]. In the European Union, the e-Commerce Directive shields service providers from liability for stored information as long as they lack knowledge of illegal activity or information and are unaware of circumstances indicating such illegality concerning damage claims

[69]. It is important to note that the interpretation and application of these laws can vary, depending on the jurisdiction and the specific case details.

Similarly, concerning claims for damages, the provider is not held responsible if they are not aware of circumstances that clearly indicate illegal activities or information. However, the liability of service providers, including AWS and others, can be triggered under these legal principles if they are informed about unlawful content but fail to promptly remove it or disable access to it.

If users come across objectionable content, they have the option to report it to the service providers, who will take appropriate action to address it. For instance, Cloudflare disclaims control over decentralized name registries and material accessible via the Distributed Web Gateway, stating that copyright holders must report infringement or abuse through Cloudflare's automated form for network-served material [20].