ParTEETor: A System for Partial Deployments of TEEs within Tor

Rachel King
University of Wisconsin-Madison
Madison, WI, USA
rachelking@cs.wisc.edu

Quinn Burke
University of Wisconsin-Madison
Madison, WI, USA
qkb@cs.wisc.edu

Yohan Beugin
University of Wisconsin-Madison
Madison, WI, USA
ybeugin@cs.wisc.edu

Blaine Hoak
University of Wisconsin-Madison
Madison, WI, USA
bhoak@cs.wisc.edu

Kunyang Li
University of Wisconsin-Madison
Madison, WI, USA
kli253@wisc.edu

Eric Pauley
University of Wisconsin-Madison
Madison, WI, USA
epauley@cs.wisc.edu

Ryan Sheatsley
University of Wisconsin-Madison
Madison, WI, USA
sheatsley@wisc.edu

Patrick McDaniel
University of Wisconsin-Madison
Madison, WI, USA
mcdaniel@cs.wisc.edu

Abstract

The Tor anonymity network allows users such as political activists and those under repressive governments to protect their privacy when communicating over the internet. At the same time, Tor has been demonstrated to be vulnerable to several classes of deanonymizing attacks that expose user behavior and identities. Prior work has shown that these threats can be mitigated by leveraging trusted execution environments (TEEs). However, previous proposals assume that all relays in the network will be TEE-based—which as a practical matter is unrealistic. In this work, we introduce `ParTEETor`, a Tor-variant system, which leverages partial deployments of TEEs to thwart known attacks. We study two modes of operation: non-policy and policy. Non-policy mode uses the existing Tor relay selection algorithm to provide users incident security. Policy mode extends the relay selection algorithm to address the classes of attacks by enforcing a specific TEE circuit configuration. We evaluate `ParTEETor` for security, performance, and privacy. Our evaluation demonstrates that at even a small TEE penetration (e.g., 10% of relays are TEE-based), users can reach performance of Tor today while enforcing a security policy to guarantee protection from at least two classes of attacks. Overall, we find that partial deployments of TEEs can substantially improve the security of Tor, without a significant impact on performance or privacy.

1 Introduction

Anonymity networks have existed for decades to provide users enhanced privacy when browsing the internet [8, 11, 36, 42]. The onion routing network Tor [9] is a prominent example that allows users to access/host services anonymously and bypass censorship. Entirely non-profit, Tor relies on people around the world volunteering their computing resources to host *relays*, which route users' traffic through the network via unique, random paths known as *circuits*. Currently, Tor has over 2 million users and over 6000 public relays in use [35].

While Tor enjoys widespread use, vulnerabilities in its infrastructure continue to emerge and threaten user privacy. Attempts at exploiting protocol-level information (e.g., circuit identifiers) have been successful in deanonymizing users; known classes of attacks on Tor include collusion [5, 34], fingerprinting [24], circuit identifier exploitation [6], and bandwidth inflation [4]. Ultimately, all of these attacks exploit the trust that is inherently placed in relays to safely route users' traffic. Developing a solution for effectively preventing such attacks is thus critical for the long-term success of Tor and the guarantees provided to users that rely on Tor.

Prior efforts like SGX-Tor [23] have used trusted execution environments (TEEs) to provide a means for trusting relays. However, the system assumes that every relay in the network is TEE-based. As the Tor network is entirely volunteer, and adopting TEEs necessarily requires software re-design and, for some relay operators, hardware changes, this greenfield assumption is at odds with practical considerations. Coordinating a transition to TEE-based relays among unassociated relay operators is infeasible, and thus this assumption is prohibitive. In practice, TEE-based solutions should tolerate partial rollout as relay operators begin adopting TEE-capable hardware and transitioning relay software. Thus, understanding the security guarantees of leveraging TEEs in partial (or incremental) settings will be pivotal to improving anonymity networks such as Tor.

In this paper, we introduce the `ParTEETor` system which provides security in partial deployments of TEE-based relays in the Tor network. Two modes of operation are provided: non-policy and policy-based. Non-policy mode allows users to benefit from TEE-based relays where they are available using the current Tor relay selection algorithm. Policy mode allows users to mandate specific TEE circuit configurations to guarantee protection from different classes of attacks.

As an initial means of evaluating partial deployments, we create a taxonomy of classes of attacks on Tor. We then map those classes to TEE circuit configurations that mitigate them. The identified configurations are designated as *security policies*. We introduce an *extended relay selection algorithm* that selects circuits that meet a client-specified security policy.

Next, we develop a simulation of `ParTEETor` using Python to evaluate its security, performance, and privacy over four realistic *deployment scenarios* of TEEs. We evaluate security as the general TEE protection circuits receive with no security policy enforced. To evaluate performance, we use measurements provided by the

Tor directory consensus for expected bandwidth of circuits when enforcing a security policy. Privacy is evaluated as the resulting reduction in circuit availability when enforcing a security policy.

We find that non-policy mode offers protection from at least two classes of attacks to more than 50% of circuits if at least 20% of relays are TEE-based. A possible consequence of policy mode is increased congestion at TEE-based relays if there is a limited penetration of TEEs. However, we still find that with only 10% of TEE-based relays, users reach an expected bandwidth of 8016.1 KB/s, meeting performance seen in Tor today while obtaining protection from two classes of attacks. While enforcing a security policy also reduces the space of available circuits, the reduction in this space still meets the privacy guarantees observed in former versions of Tor.

Our work shows that the use of TEEs in Tor is both practical and effective. We hope that our work encourages relay operators to begin taking concrete steps towards using TEEs to reinforce the long-term success of Tor and anonymity networks.

2 Background

Tor Network. Tor [9] is a volunteer-based anonymity network that uses onion routing [12] to conceal a user's internet activities. The main components of Tor are *clients*, *relays*, *directory authorities*, and *onion services*. Implemented as an overlay network, Tor encapsulates client TCP streams in multiple layers of encryption and routes them through several intermediate servers between a source (client) and destination (website). No component is aware of both the source and destination, thus allowing clients to remain anonymous to the wider internet (i.e., to relays, web servers, and other third-parties).

A user's Tor client establishes a *circuit* to send data by choosing three (or more) relays during *relay selection*. Ephemeral, pairwise encryption keys are negotiated between the client and each relay. To anonymize data for transmission, the client divides the data into fixed-size *cells* and successively encrypts every cell under each key, beginning with the last relay's key, followed by the middle, then the first. During transmission, each relay decrypts the cell to remove their layer of encapsulation before forwarding it to the next relay in the circuit. A relay is therefore only aware of the identities of components immediately prior and after itself in the circuit.

Directory authorities are a subset of ten relays in the network that are trusted to maintain the overall state of the network. Their central responsibility is upholding the *directory consensus document*, a document containing key information about each relay (e.g., public keys, bandwidth, flags, etc.) that clients query during relay selection. The directory authorities scan the network periodically to update the consensus document.

Onion services (formerly known as hidden services) are special sites and services only accessible through Tor. The key components of onion services are the *introduction points*, which are anonymous contact points of the site hosts, and *rendezvous points*, which connect anonymous circuits from the client and the site host. Clients first communicate with the introduction point of the onion service to identify a rendezvous point. Onion services then have their inbound traffic routed through a circuit to the rendezvous point. Clients also establish a circuit of their own to the rendezvous point.

Trusted Execution Environments. Trusted execution environments (TEEs) are hardware-based security primitives that protect the confidentiality and integrity of code and data running in untrusted environments (e.g., on outsourced servers) [22, 28, 29]. TEEs protect confidentiality primarily through *access-mediation* to protected memory regions containing sensitive code and data, and additional *CPU modes* that restrict the type and scope of operations that can be performed by code running either within the context of the TEE or outside of it. In effect, this ensures complete isolation from untrusted third-parties while processing sensitive data. *Attestation* capabilities are also provided to allow clients to verify the integrity of the code running inside the TEE. This grants clients assurance of the behavior of the code running on machines they do not physically manage. TEEs have begun to see wide adoption for securing cloud-based web services [7], but also for improving the security of Tor.

SGX-Tor. The SGX-Tor system demonstrated that TEEs provide an effective means to defend against several longstanding threats to Tor [23]. This was realized by porting the Tor software to run within a TEE. This ensures that sensitive data is kept secret and functions processing sensitive data cannot be tampered with. To enable communication with the outside world, the host that the TEE runs on acts as a proxy and forwards messages to endpoints specified by the TEE (e.g., to clients and other relays in Tor). Any data that requires persistence on the relay can then be sealed using a private key burned into the processor hardware and only accessible by the TEE code [15]. More broadly, the confidentiality, integrity, and attestation capabilities of the TEE thus ensure the trustworthiness of the relays in the network, preventing an otherwise untrustworthy relay from leaking sensitive information like circuit IDs, cell commands, router descriptors, or private encryption keys used for communication with clients. This effectively reduces the threat surface within Tor to only network-level adversaries.

3 System Overview

SGX-Tor [23] laid the foundation for integrating TEEs into Tor. However, it was limited by the assumption that every Tor client and relay has a TEE.

This greenfield assumption is impractical as it implies the entire network will be replaced, instead of allowing for compatibility of both TEE and non-TEE relays. While many systems already have the hardware to support TEEs, adoption still requires software re-design. For some relay operators, adoption could also mean hardware changes. Coordinating a transition to all TEE-based relays among unassociated relay operators is infeasible, and thus this assumption is prohibitive.

To stimulate adoption of TEEs by the broader Tor community, our central goal is therefore to construct a system that leverages partial deployments of TEE-based relays to enable users security, performance, and privacy.

ParTEETor. Based on the existing Tor design, the `ParTEETor` system (Figure 1) integrates TEEs in some relays—varying from very few to nearly all (See Section 6).

While one might expect to need hardware changes in order to use `ParTEETor`, this is not necessarily the case. Implementing

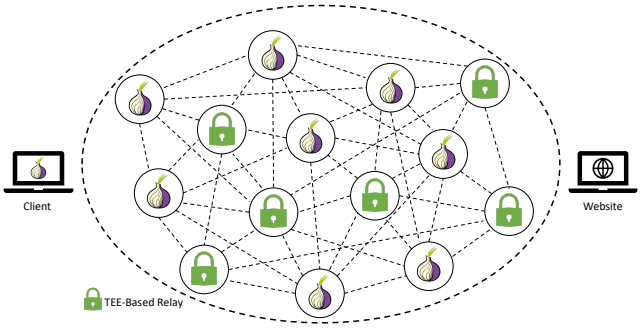


Figure 1: ParTEETor, a partial deployment of TEE-based relays in the Tor network.

TEEs is standard practice now for general purpose CPUS. Popular manufacturers like Intel, ARM, AMD, and Apple all utilize TEEs in their CPUs [2, 3, 16, 26, 27]. Therefore, many relay operators will only need to install a binary which utilizes their already existing TEE in order to use ParTEETor.

Like SGX-Tor, the relay code of TEE-based relays is moved into the TEE, thereby providing data and operation integrity and confidentiality guarantees. Moreover, attestation is used to ensure the integrity of the implementation (code). Thus, when a relay wishes to be available to clients as TEE-based, it must first be attested by the directory authorities, and re-attested at some periodicity to ensure the ongoing integrity of the relay.

ParTEETor operates in one of two modes: non-policy and policy (See Section 5). The non-policy mode is a straightforward application of Tor with the addition of TEEs being adopted by select existing relays. The relay selection algorithm in this mode does not change, allowing for circuits to incidentally benefit from the availability of TEE-based relays in the network. This mode represents the case where no changes to the client are required, as all changes to behavior exist solely in the TEE-based relays.

ParTEETor's policy mode extends non-policy mode by incorporating security policies into the relay selection algorithm. When a client connects to the network, it will select its circuit based on its desired security policy, which defines which relays in a circuit are TEE-enabled (See Section 4). The security policy enables users to choose their protection status from each class of attack based on the perceived threats. This mode requires updating the client to impose the selection of TEE-based relays in circuits.

Threat Model and Assumptions. We assume a similar threat model to other TEE-based systems [7, 23]. Adversaries are exploiting the required trust placed on relays to safely route traffic through the network. An adversary may be a compromised entry, middle, or exit relay in the network. They may attempt to corrupt or extract private information from the relay, such as circuit IDs or cell commands. However, we assume that the confidentiality and integrity of any code and data processed within the TEE is protected. We do not trust software running outside of the TEE (i.e., the operating system acting as a proxy for the TEE). We also assume that users trust their Tor client and therefore do not consider malicious client attacks [10, 19, 32]. We assume that all directory authorities are trustworthy, as these relays are operated by trusted parties in the Tor community. We note the limitations of TEEs in Section 7.

4 Mapping Attacks to Circuit Protection

In order to understand what security policies ParTEETor needs to support, we need to understand the relationship between TEE-based relays and their placement in circuits to protect against the known classes of attacks on Tor users (Table 1). To do this, we present a security analysis on the known classes of attacks in Tor. These attacks were identified by prior work as the universal examples for each known attack class [23].

As not all attacks require collusion of every relay in a circuit, ensuring a TEE-based relay is present in every position of a circuit would not be necessary for protection from each attack. Prior work provided a preliminary analysis on the minimum requirement of TEE-based relays [23]; we expand on their efforts by detailing how the specific TEE placement in a circuit provides protection against each attack. We breakdown each attack to identify the specific position (i.e., distinct relays in a circuit) where TEEs are required in a circuit to ensure mitigation.

4.1 Replay Attack

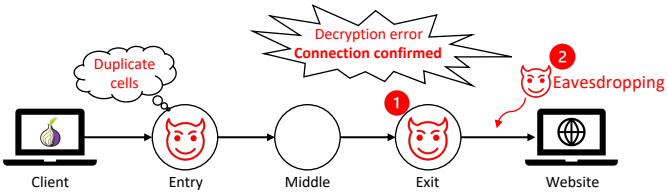


Figure 2: Replay Attack Scenario

Pries *et al.* propose the Replay Attack [34], where an adversary duplicates Tor cells, causing decryption errors. As Tor uses AES-CTR for cell encryption, each relay and client maintain a counter when encrypting and decrypting along a circuit. Duplicating a cell will result in the counter being additionally incremented at the relays decrypting it, which causes the client and the relays' counters to be out of sync. This will then cause a decryption error at either edge of the circuit.

This attack can take two approaches, as shown in Figure 2: both the entry and exit relays to be malicious, or the entry is malicious and an adversary is eavesdropping on the connection between the exit relay and the destination. The initial premise is a malicious entry relay duplicates a relay cell before forwarding it down the circuit. Once the duplicated cell reaches the exit relay, the integrity check will fail and cause the circuit to be torn down. A malicious exit relay working with the entry can confirm the user and the destination, as this error was forced by the entry for the exit to recognize. A malicious eavesdropper, on the other hand, will notice the TCP stream being cut off unexpectedly, confirming the connection between the entry and the eavesdropper.

Protection. The Replay Attack [34] relies on malicious relay behavior. Specifically, the entry relay must duplicate the target cell to cause the decryption error. The guarantee of source code integrity

Attack	Adversary Relays	Adversarial Goal	TEE Requirement
Replay	Entry and Exit	Deanonymize users	Entry
Onion Services	Entry and Exit	Deanonymize onion services	Exit
Fingerprinting	Entry	Deanonymize users and onion services	Entry
Bad Apple	Exit	Deanonymize users	Exit
Bandwidth Inflation	Entry, Middle, and Exit	Increase relay's usage in circuits	Entry, Middle, and Exit

Table 1: Required TEE placement to mitigate attacks against Tor.

with a TEE prevents this behavior. Protection from this attack requires a TEE-based entry relay. It is not necessary to require both a TEE-based entry relay and exit relay. If the entry relay is TEE-based, we can guarantee this relay will not duplicate any cells, meaning any adversary at the exit relay will have no decryption error.

Considering the variation of the attack where the entry relay is malicious with an eavesdropper on the exit-to-destination link, a TEE cannot prevent this eavesdropper. This is the reasoning behind requiring the **entry** relay to have a TEE and not the exit: preventing the duplication of the cell from ever taking place at the entry, thereby impeding the eavesdropper from noticing any unexpected dropped connection.

4.2 Onion Services Attack

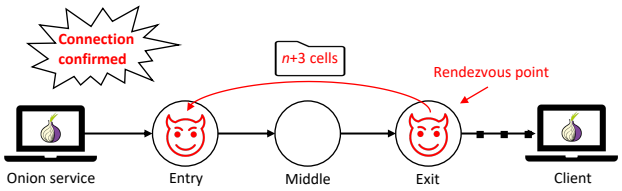


Figure 3: Onion Services Attack Scenario

Biryukov *et al.* propose an onion services attack [5] which relies on a malicious entry or middle relay, and rendezvous point to reveal the location of onion services, as shown in Figure 3. The adversary intends to confirm they are running the entry relay for an onion service by recognizing a pattern of cells sent by the rendezvous point. When the adversary requests to be introduced to the onion service as a standard Tor user, they provide their malicious rendezvous point to the onion service. The onion service then constructs a circuit to the rendezvous point, where the adversary sends a known quantity n (e.g. 50) of *padding* cells down the circuit, followed by a *destroy* cell.

If the adversary is the entry relay in this circuit to the rendezvous point, they will receive $n + 3$ cells in total, two *extended* cells from circuit establishment, n *padding* cells, then one *destroy* cell. This scenario confirms the adversary is the entry relay of the onion service, allowing them to reveal the identity of the onion service as the hop prior to them. If the adversary is only the middle relay, it will receive $n + 2$ cells total (one less *extended* cell), confirming the hop prior is the entry of the circuit (we will disregard this scenario from here on, as compromising the entry relay following this scenario would require additional attacks outside of this).

Protection. The Onion Services Attack [5] requires the rendezvous point to maliciously send padding cells down the circuit. However, the integrity guarantee of TEEs prevents this behavior. The attack also requires a malicious entry in order to recognize the padding cells and count the total cells sent. Thus, a TEE-based entry relay alone does not prevent this attack, as a network-level adversary can still observe the number of cells sent via analysis of the quantity and size of IP packets. To prevent an adversary from inferring this link, the act of transmitting extra padding cells must be prevented. This requires a TEE-based relay acting as the rendezvous point, which translates to the **exit** relay in the circuit.

4.3 Fingerprinting Attack

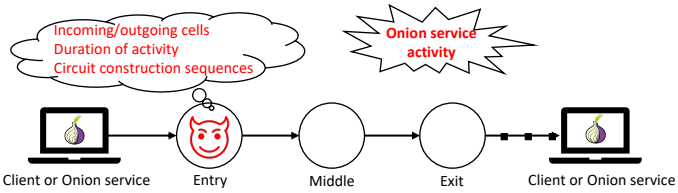


Figure 4: Fingerprinting Attack Scenario

Kwon *et al.* propose a fingerprinting attack [24] which is able to exploit circuit level identifying information to reveal if a given user is visiting an onion service, or the location of the onion service itself. This attack, shown in Figure 4, requires the adversary to be acting as the entry relay.

The adversary makes note of three different properties of the traffic to and from onion services: incoming and outgoing cells, duration of activity, circuit construction sequences. Based on these properties, introduction point circuits are first sought out, meaning circuits between a client and introduction point for an onion service. Once evidence of these circuits is found, the adversary monitors the users of these circuits further to determine rendezvous point circuits, either between a client or an onion service. The activity monitored can effectively determine if the adversary is an entry relay for an onion service or a user visiting an onion service. In the event the relay is acting as the entry for an onion service, the location of the onion service is now identified.

Protection. The Fingerprinting Attack [24] requires the entry relay to be malicious to recognize patterns of cells being sent across circuits. Additionally, the authors claim the attack can be implemented by someone eavesdropping on the connection between the

user and the entry relay. This attack is passive in that it only requires observing circuit identifying information, such as circuit IDs and number and sequences of cells, to distinguish different circuits. TEEs' guarantee of confidentiality prevents this information from being revealed to a malicious entity on the relay. A malicious host or entity eavesdropping on the connection will only see IP packets being sent.

Protection from this attack requires a TEE-based **entry** relay. This ensures the entry cannot recognize what traffic corresponds to specific circuits. An important note is that TEEs only reduce the adversary to a network level. This attack could still be possible via analysis of IP packets in an attempt to distinguish circuits.

4.4 Bad Apple Attack

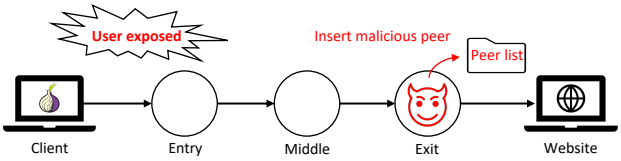


Figure 5: Bad Apple Attack Scenario

Tor multiplexes multiple TCP streams over one circuit. In the event a user is visiting a website, which may require multiple streams to fetch all the objects, these streams will be sent over the same circuit. Subsequently, if an exit relay is able to identify the source for one of the streams, it now knows the source of all the other streams along that circuit. The Bad Apple Attack [6], shown in Figure 5, exploits this design choice.

Targeting users of peer-to-peer (P2P) file sharing applications like BitTorrent, this attack requires the adversary to host a malicious exit relay, monitor users of P2P applications, and host a malicious peer for the applications. Blond *et al*'s attack exploits the fact that 70% of users accessing BitTorrent only use Tor to request peers, then connect directly to the peer via TCP.

When a user's circuit contains the malicious exit relay, and the user is requesting a list of peers to contact, the exit relay can modify the returned list to include their malicious peer. Then, when the user connects to the malicious peer outside of Tor, the user exposes their IP address (by design of P2P applications). The adversary operating the relay can then match the traffic to its clearnet peer to clients it sends data back to as an exit relay. Furthermore, the source of all other multiplexed streams of this particular exit relay's circuit are now exposed.

Protection. The Bad Apple Attack [6] protection is straightforward. This attack relies on the exit relay in a circuit to act maliciously. Moreover, no collusion is required for this attack, but the knowledge of circuit level information such as circuit and stream IDs is required to be able to distinguish between different circuits and streams. TEEs protect against the malicious relay behavior through their integrity guarantee. TEEs also hide circuit identifying information such as circuit and stream IDs. This prevents the adversary from recognizing different circuits through IDs. For these reasons, requiring the **exit** relay in a circuit to be TEE-based ensures protection under TEE capabilities. An important note is that the adversary is reduced to a network level, as they can still attempt to recognize different circuits and streams through IP packet inspection.

4.5 Bandwidth Inflation Attack

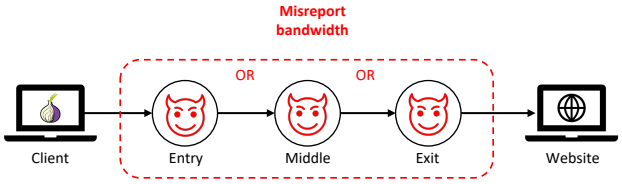


Figure 6: Bandwidth Inflation Scenario

Relays are selected for circuits weighted proportionally to their bandwidth. This creates an incentive for having higher bandwidth, as that relay is now more likely to be used in circuits. From an adversarial perspective, this is beneficial in their task of controlling entry and exit relays of a circuit [4]. Tor adopts the approach of bandwidth scanners to validate the reported bandwidth. A portion of the directory authorities are bandwidth authorities and will periodically scan the bandwidth via bidirectional probing. The published bandwidth of the relay is the median of at least three of the bandwidth authorities' measurements [21].

Bandwidth scanning is an improvement in preventing misreporting, shown in Figure 6, but doesn't defeat the attack entirely. As relays are able to recognize the directory authorities that scan for bandwidth, relays can provide more bandwidth to these streams by throttling the bandwidth they allow for the rest of its streams. This can effectively convince the directory authorities that the relay is capable of higher bandwidth which in turn increases the probability it will be chosen for circuits, as shown by Biryukov *et al.* [5].

Protection. Relays using TEEs self-report their bandwidth [23] in place of scanning. With this, any manipulation to the bandwidths being reported to the directory authorities would mismatch the bandwidth reported by the trusted TEE relay. As this attack is specific to each relay, protection requires **all (entry, middle, and exit)** relays in a circuit to be TEE-based.

Requiring the entry and exit relays in a circuit to be TEE-based would be an effective mitigation if considering bandwidth inflation as a means to leverage more attacks. Most attacks discussed require collusion, except for Fingerprinting [24] (malicious entry relay) and Bad Apple [6] (malicious exit relay). Requiring all relays in a circuit to be TEE-based would then not be necessary.

5 ParTEETor

ParTEETor is driven primarily by two components: a client's *security policy* and the *extended relay selection algorithm*. Additionally, how TEEs are deployed within the system (i.e., the *deployment scenario*) will have significant impact on its effectiveness. We discuss each aspect of the system and its use below.

5.1 Circuit Security Policy

A client's security policy specifies which attacks a circuit must provide protection against. Concretely, it describes what TEE-based relays must be present (and how they must be arranged)

```
Function RelaySelection (G = (V, E), R = (P, T)):
    circuit = [ ];
    for position, TEEreq ∈ R do
        relaylist = { };
        for v ∈ V do
            if position ∈ v.positions then
                if v.TEE or not TEEreq // Security Policy Extension then
                    add v to relaylist;
                end
            end
        end
        totalBW = ∑_{∀r∈relaylist} r.bandwidth;
        select relay r with probability r.bandwidth / totalBW;
        add relay to circuit;
    end
    return circuit
end
```

Algorithm 1: Bandwidth weighted relay selection for circuits. $G = (V, E)$ is the graph representing the Tor network and R represents the configuration for the circuit, *position*, which contains the relay types (default is Entry, Middle, Exit), and security policy of TEE requirements for the circuit, $TEEreq$. $v.positions$ is the individual relay's circuit position capabilities, and $v.TEE$ is its TEE status.

within a circuit to provide protection. Our mapping from Table 1 prescribes five policies: $\{\emptyset\}$, $\{Entry\}$, $\{Exit\}$, $\{Entry, Exit\}$, and $\{Entry, Middle, Exit\}$. For non-policy mode, a security policy is therefore empty. We evaluate the performance, security, and privacy consequences of enforcing the policies in realistic deployments in the following sections.

5.2 Extended Relay Selection Algorithm

Our extended relay selection algorithm adapts the relay selection algorithm currently used by Tor to enable clients to find policy-compliant circuits. In the base algorithm [41], relays are weighted solely by their available bandwidth when being selected for use in circuits. In our extended algorithm, client select circuits in one of two ways. In non-policy mode, the algorithm simply falls back to the base algorithm. However, in policy mode, it additionally considers both the client's security policy and the availability of TEE-based relays in the network (published by the directory authorities in the directory consensus document). An overview of the algorithm is shown in Algorithm 1. It works by iteratively searching the set of all relays to identify candidate TEE-based relays to use at each position in the circuit being constructed. Relays are similarly weighted by their available bandwidth during selection (line 14).

5.3 Deployment Scenarios

The availability of TEE-based relays in the Tor network (and their arrangement within a given circuit) directly impacts user security, performance, and privacy. To understand the efficacy of partial deployments, we introduce the notion of deployment scenarios. A deployment scenario describes a strategy for how relay operators could realistically begin upgrading their relays to run TEE-based hardware and software. We introduce four deployment scenarios that broadly capture such strategies: `Random`, `Bandwidth Weighted`, `Inverse Bandwidth Weighted`, and `Circuit-Position Weighted`. The first three allow a relay operator to decide which relays to upgrade by assigning a probability of being chosen for upgrade to each relay (either randomly or based on general relay characteristics like available bandwidth). The last allows a relay operator to more intelligently select relays for upgrade by assigning select probabilities to relays occupying distinct circuit positions.

Random Deployment. `Random` deployment weights w for each relay uniformly to represent a fully randomized selection. Explicitly, $w = 1/n$ where n is the number of relays in the network. No relay-specific characteristics are taken into account.

Bandwidth Weighted Deployment. `Bandwidth Weighted` deployment assigns weight w based on its bandwidth. This weight remains fixed. This approach is much like, and motivated by, the relay selection of Tor. Relays with higher bandwidth are more likely to be chosen for circuits because they can withstand more traffic. Considering this, these relay operators have more incentive to adopt TEEs. We consider this approach a best case scenario, in that requiring TEE-based relays will likely increase a user's performance.

Inverse Bandwidth Weighted Deployment. `Inverse Bandwidth Weighted` deployment assigns weight w based on the inverse of a relay's bandwidth. This weight remains fixed. This approach is the opposite of the `Bandwidth Weighted` deployment. Our motivation for this deployment is to understand the worst case scenario in terms of performance when requiring TEE-based relays, as now all of the relays with lesser-bandwidth will have a higher likelihood of being assigned to be TEE-based.

Circuit-Position Weighted Deployment. `Circuit-Position Weighted` deployment consists of four deployment distributions: Entry, Exit, Entry-Exit, and Entry-Middle-Exit. Each distribution selects relays specifically with respect to their circuit position capabilities. Therefore, each deployment is dependent on three weights: w_e = TEE-based entry relays, w_m = TEE-based middle relays, and w_x = TEE-based exit relays. Each of these weights represent the percentage of which are TEE-based, with respect to the total number of relays of that type.

In terms of the capabilities of relays with respect to their position in a circuit, every relay in the network is middle-capable. All entry relays can be in both the entry and middle positions of a circuit, and all exit relays can be in middle and exit positions. Additionally, some relays are both entry and exit-capable. Therefore, when considering these weights, under distributions Entry-Exit, and Entry-Middle-Exit, selection of relays may overlap. For example, selecting 10% of entry relays to be TEE-based, then selecting 10% of exit relays to be TEE-based could result in actually 15% of entry relays being

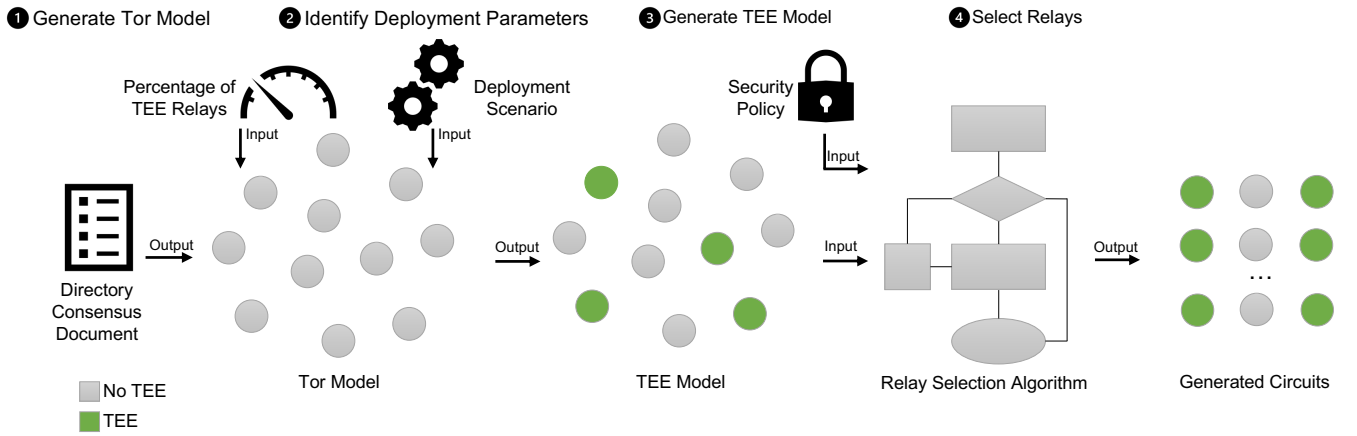


Figure 7: Pipeline of events for modeling partial deployments of TEEs in ParTEETor. A Tor network model is generated using relay data from the directory consensus. Relays are probabilistically assigned to be TEE-based for each deployment scenarios. Using Tor's relay selection algorithm with an additional parameter for TEE security policy, circuits are generated for analysis of their security and performance properties.

TEE-based because half of the exit relays chosen could also be entry-capable.

As each distribution assigns TEE-based relays according to their circuit position capabilities, weight assignments are dependent on the specific distribution:

- Entry: only vary the entry-capable relays, w_e
- Exit: only vary the exit-capable relays, w_x
- Entry-Exit: vary the entry-capable relays, w_e, and the exit-capable relays, w_x
- Entry-Middle-Exit: vary all relays, w_e, w_m, w_x

These deployment distributions provide insight into the impact each type of relay has on the network. They may be useful in assessing the trade-offs between relay cost and expected security/performance improvement (e.g., deciding to upgrade entry relays first, as they provide significant bandwidth to the network).

6 Evaluation

We evaluate ParTEETor via simulation of the real Tor network. The experiments here seek to answer the following questions: *(1) How many circuits benefit from the addition of TEEs under the existing relay selection algorithm? (2) How much congestion is present in circuits when enforcing TEE requirements under partial deployments? (3) What is the reduction in availability of circuits when enforcing TEE requirements under partial deployments?* We begin by briefly introducing the simulation framework and experimental evaluation.

Simulation Framework. The ParTEETor simulation models its behavior under different partial deployments of TEEs in the Tor network (Figure 7). We model the network as a graph [18], where each relay uses data from the advertised Tor network (i.e., IP address, expected bandwidth) as well as a TEE status representing whether the node has a TEE or not. Under each deployment scenario, we probabilistically assign relays to be TEE-based. For non-policy mode, we apply the existing Tor relay selection algorithm to simulate the behavior of selecting relays for circuits. We extend the algorithm by including an additional parameter for the security policy of the circuits for policy mode. We then generate circuits for two sets of experiments.

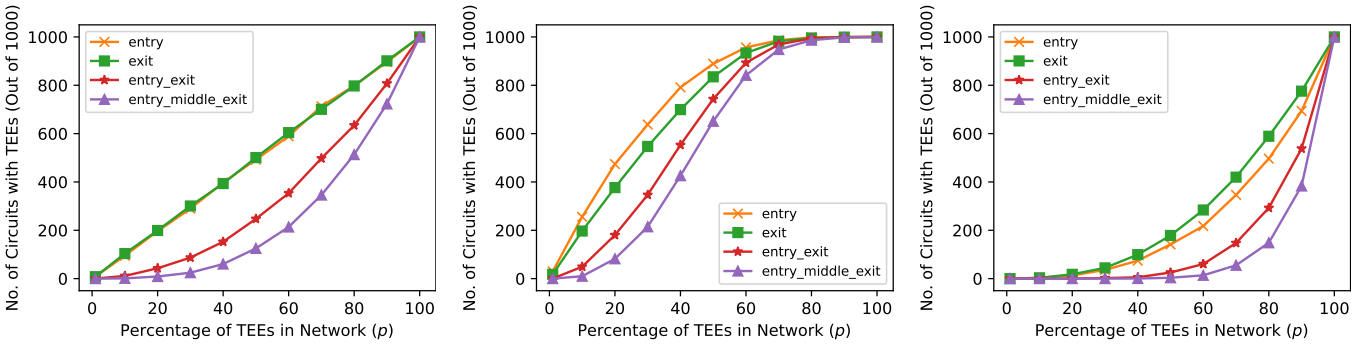
6.1 Experiment Setup

Our ParTEETor simulator is written in Python, using the networkx library to model relations between relay nodes. Simulations were performed on a Mac M1 CPU with 16 GB of ram and 3.2 GHz max clock speed. We use a directory consensus document published on February 26, 2023 [20] to generate our graph. In total, there are 6356 nodes, with 3179 having entry capability, and 1668 having exit capability. 849 nodes overlap in the categories, being both entry- and exit-capable. We run our extended relay selection algorithm to generate 1000 potential circuits per trial for each security policy. We run each trial 10 times and take the mean to normalize our results.

For Random, Bandwidth Weighted, and Inverse Bandwidth Weighted deployments, we denote p as the percentage of total TEE-based relays in the network. We evaluate p for the range from 1% to 100%. Depending on the specific deployment, the distribution of TEEs will vary. Recall that w is the probability a relay will be TEE-based and is specific to each deployment (see Section 5.3).

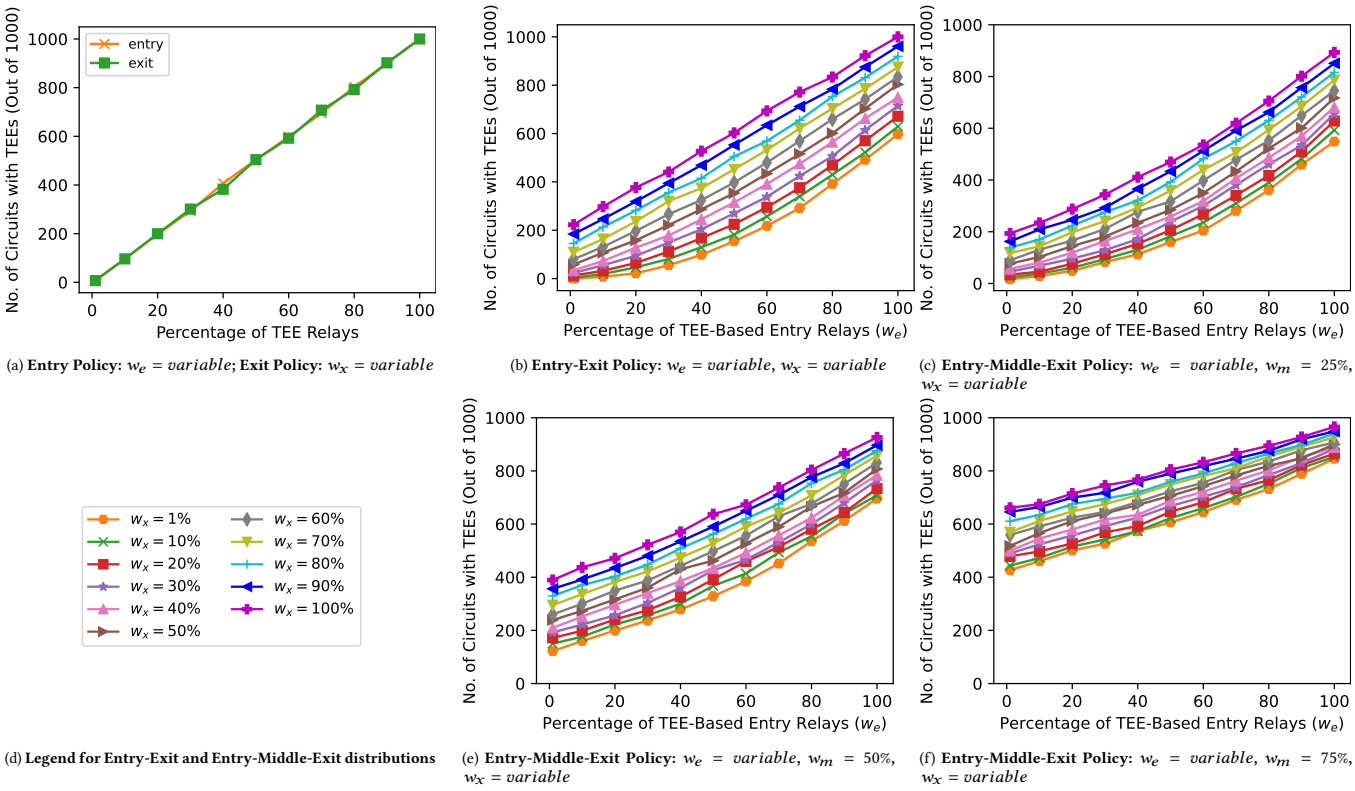
As Circuit-Position Weighted deployment specifically targets relays with respect to their circuit position capabilities, p is not an experimental parameter. Instead, weights w_e, w_m, and w_x are iterated through as experimental parameters for each TEE distribution of this deployment. When a weight is a variable in the deployment, we evaluate it over a range from 1% to 100%.

Evaluation Metrics. Our security metric is defined as the number of circuits generated that satisfy each security policy, when no policy is provided. This reflects the security implications of ParTEETor in non-policy mode, where TEE-based relays are present in the network and the existing relay selection algorithm is not extended.



(a) **Random Relay** (b) **Bandwidth Weighted** (c) **Inverse Bandwidth Weighted**

Figure 8: TEE presence in circuits when no TEE security policy is specified when incrementing the percentage of TEEs present in the network, p. Results are the mean over 10 trials of 1000 circuits each.



(a) **Entry Policy: w_e = *variable*; Exit Policy: w_x = *variable***

(b) **Entry-Exit Policy: w_e = *variable*, w_x = *variable***

(c) **Entry-Middle-Exit Policy: w_e = *variable*, w_m = 25%, w_x = *variable***

(d) **Legend for Entry-Exit and Entry-Middle-Exit distributions**

(e) **Entry-Middle-Exit Policy: w_e = *variable*, w_m = 50%, w_x = *variable***

(f) **Entry-Middle-Exit Policy: w_e = *variable*, w_m = 75%, w_x = *variable***

Figure 9: TEE presence in circuits when no TEE security policy is specified for the `Circuit-Position Weighted` deployment. Each policy increments weights w_e, w_m, and w_x as noted in the figures. Results are the mean over 10 trials of 1000 circuits each.

Our performance metric is defined as the expected median bandwidth of circuits generated with a security policy enforced. This reflects `ParTEETor` in policy mode, where TEE-based circuit requirements are enforced through the extended relay selection algorithm. We evaluate performance in terms of expected bandwidth, as this is a standard metric for performance in the Tor network [33, 40]. Additionally, the cost of requiring TEE-based relays in circuits is increased congestion, which we can study via the expected bandwidth. To compute the expected bandwidth of a circuit, we consult the directory consensus document for expected bandwidth measurements of all relays, as measured by the bandwidth directory authorities. Next, we divide each relay's expected bandwidth by the number of circuits it is a part of. The expected bandwidth of a circuit is then the minimum bandwidth of all relays in the circuit.

We do not consider latency in our performance evaluation, as it is not a factor in the relay selection algorithm. Due to the minimal delays (3.9%) incurred by TEEs in end-to-end performance of Tor [23], our simulation assumes zero overhead of TEEs.

Our privacy metric is defined as the number of unique circuits that are possible under security policies when there is a limited TEE percentage. This reflects `ParTEETor` in policy mode and presents the reduction in space of circuits, as specifying a policy reduces the size of the network to TEE-based relays.

6.2 Security of `ParTEETor` in Non-Policy Mode

In order to understand the security implications of `ParTEETor` in non-policy mode, we investigate what TEE coverage circuits can achieve without a security policy being enforced. Results for `Random`, `Bandwidth Weighted`, and `Inverse Bandwidth Weighted` deployments are found in Figure 8. We iterate over the total number of relays in the network, p, and quantify the number of circuits complying to each security policy. These results analyze circuits generated for all security policies. Results for the `Position-Weighted` deployment are found in Figure 9. We iterate over weights w_e, w_m, and w_x, depending on the TEE distribution, and quantify the number of circuits complying to the distribution. These results only analyze circuits generated with the security policy synonymous with the deployment distribution. For example, the Entry-Exit distribution's security results only present circuits that had both a TEE-based entry relay and TEE-based exit relay.

Random. Security results for `Random` deployment are in Figure 8a. These results show what security can be achieved with no strategic placement of TEEs in the network.

Statistically speaking, as w is uniform across all relays, the probability of a circuit containing a TEE-based relay in a given position is equivalent to p. This is seen in the fact the percentage of TEE-based entry relay circuits is almost identical to the percentage of TEE-based exit relay circuits, which are both almost identical to p.

As expected, circuits with multiple TEE-based relays are less prevalent. Statistically, the probability both a TEE-based entry and TEE-based exit relay are in a circuit is p^2. Then, the probability of each position in the circuit containing a TEE-based relay is p^3. Under this deployment, for circuits to contain a TEE-based relay in every position, at least an 80% TEE presence is required.

Bandwidth Weighted. Results for `Bandwidth Weighted` deployment are shown in Figure 8b. This deployment provides the highest number of circuits with TEE-based relays under all policies in comparison to all deployments. There is an increased likelihood of a TEE-based relay being present because of Tor's relay selection algorithm, as the relays with more bandwidth are selected to be TEE-based.

At only 23% TEE presence, 50% of circuits have a TEE-based entry relay, protecting users from two classes of attacks. At 28% TEE presence, 50% of circuits have a TEE-based exit relay, also protecting users from two classes of attacks. More significantly, though, at 44% TEE presence, more than 50% of circuits have a TEE-based relay in every position, providing protection from all classes of attacks.

Inverse Bandwidth Weighted. The `Inverse Bandwidth Weighted` deployment is shown in Figure 8c. This deployment has the lowest number of circuits with TEE-based relays under all policies in comparison to all the deployments. This is expected, once again, because of Tor's relay selection algorithm. The likelihood of receiving a circuit with a TEE-based relay is significantly reduced in this deployment because of the minimal bandwidth they offer the network.

With this deployment, in order for 50% of circuits to have at least one TEE-based relay, a TEE penetration of greater than 80% is required. For comparison, this is the same deployment percentage required for more than 50% of circuits to have all TEE-based relays (protection from every class of attack) in the `Random` deployment.

Circuit-Position Weighted. The security results of `Circuit-Position Weighted` deployment provide insight into which types of relays adopting TEEs can most improve security. Results for Entry distribution and Exit distribution are overlaid in Figure 9a. With this, the Entry distribution is iterating over w_e, while the Exit distribution is iterating over w_x. The results of these distributions are defined in the same manner as `Random` deployment. For Entry, the probability of a circuit containing a TEE-based entry relay is equivalent to w_e. For Exit, the probability of a circuit containing a TEE-based exit relay is equivalent to w_x. With this, both deployment distributions grow linearly as w_e and w_x increase.

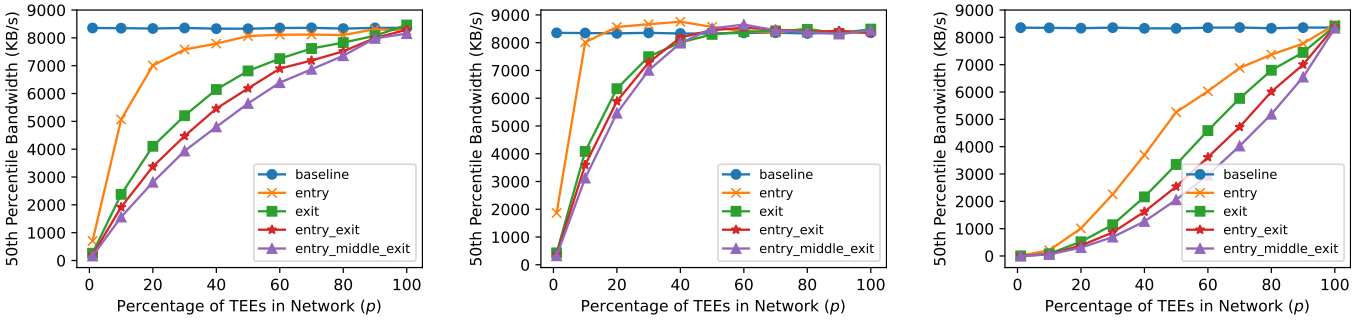
Entry-Exit (Figure 9b) and Entry-Middle-Exit (Figure 9c, Figure 9e, and Figure 9f) continue with steady growth. Interestingly, at low values of w_e but higher values of w_x, we see more circuits with TEE-based relays than one might expect. For example, with $w_e = 1\%$ and $w_x = 70\%$ in the Entry-Exit distribution, 10.9% of circuits still had both a TEE-based entry relay and TEE-based exit relay. With the Entry-Middle-Exit distribution, adding in $w_m = 50\%$, 29.5% of circuits are entirely TEE-based. These results are because of the multiple capabilities relays may have. As all relays are middle-capable, and many exits are entry-capable, increasing w_m and w_x inherently increases w_e in some manner.

These results also show the increase of TEE-based entry relays results in more circuits in comparison to the increase of TEE exit relays. For example, with $w_e = 60\%$ and $w_x = 30\%$ in the Entry-Exit distribution, 33.8% of circuits have a TEE-based entry relay and TEE-based exit relay. With these values reversed, only 26.6% of circuits have a TEE-based entry relay and TEE-based exit relay.

Takeaways. Overall, `ParTEETor` in non-policy mode provides users an immediate improvement to their security in terms of protection from known classes of attacks. While any TEE-based relay will benefit the network, higher bandwidth relays have the most influence because of how often they are used. Additionally, relays that are capable of being used in all positions of a circuit have a substantial impact, as they are not forced to adhere to one, or even two positions. Therefore, relay operators with high bandwidth, or that allow for multiple uses are capable of significantly improving the security of Tor by adopting TEEs.

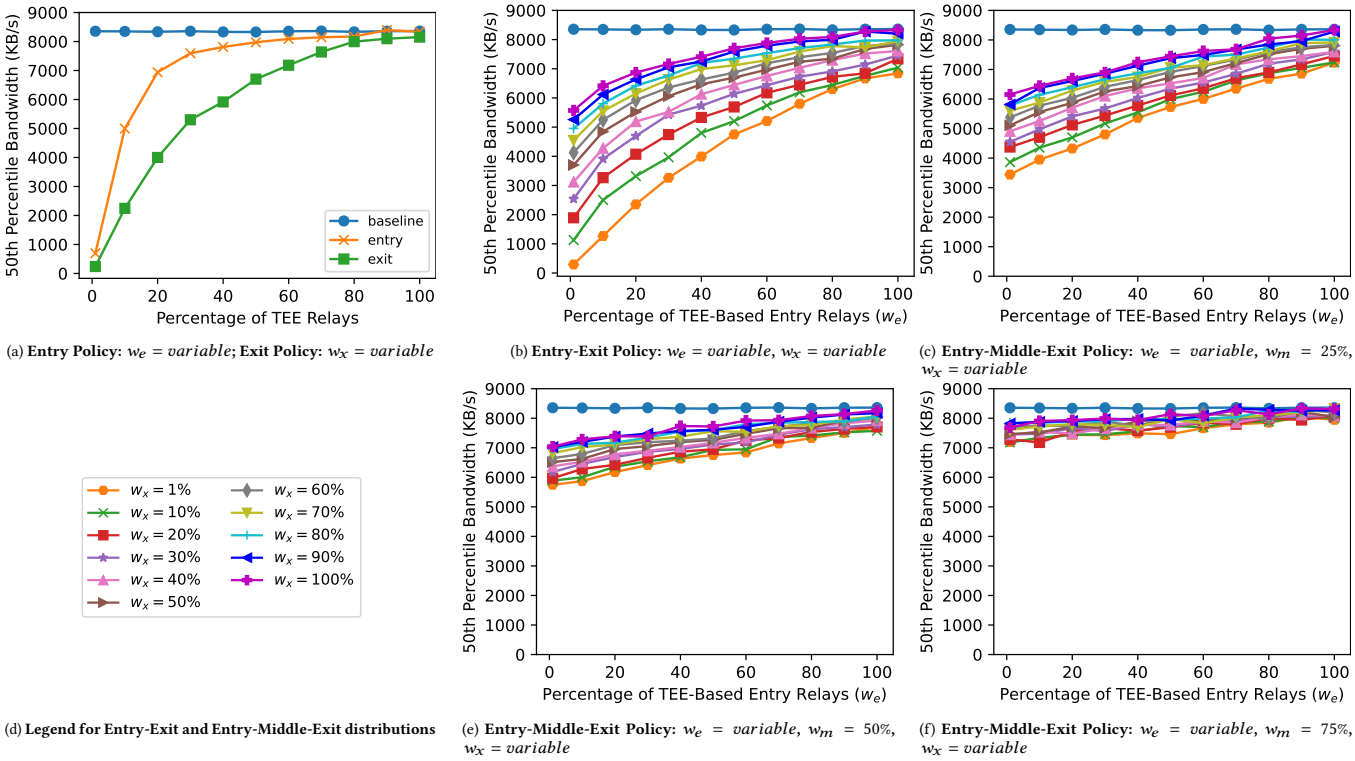
6.3 Performance of `ParTEETor` in Policy Mode

Recall that `ParTEETor` in policy mode enforces security policies by extending the relay selection algorithm. This mode guarantees a



(a) **Random Relay** (b) **Bandwidth Weighted** (c) **Inverse Bandwidth Weighted**

Figure 10: The median percentile bandwidth of circuits generated with each security policy when incrementing the percentage of TEEs present in the network, p. Results are the mean over 10 trials of 1000 circuits each.



(a) **Entry Policy:** $w_e = variable$; **Exit Policy:** $w_x = variable$

(b) **Entry-Exit Policy:** $w_e = variable$, $w_x = variable$

(c) **Entry-Middle-Exit Policy:** $w_e = variable$, $w_m = 25\%$, $w_x = variable$

(d) **Legend for Entry-Exit and Entry-Middle-Exit distributions**

(e) **Entry-Middle-Exit Policy:** $w_e = variable$, $w_m = 50\%$, $w_x = variable$

(f) **Entry-Middle-Exit Policy:** $w_e = variable$, $w_m = 75\%$, $w_x = variable$

Figure 11: The median percentile bandwidth of circuits generated for the `Circuit-Position Weighted` deployment, with each policy incrementing weights w_e, w_m, and w_x as noted in the figures. Results are the mean over 10 trials of 1000 circuits each.

user's security and is ideal for users concerned about protection from each class of attack. However, a potential consequence of this mode is increased congestion. Under limited deployments of TEEs, congestion may be more prevalent for users, as circuits will be forced to use the limited available TEE-based relays. To quantify this performance cost, we generate circuits with required TEE security policies. We then calculate median expected bandwidth of all circuits.

Results for `Random`, `Bandwidth Weighted`, and `Inverse Bandwidth Weighted` deployments are found in Figure 10. We iterate over the total number of relays in the network, p. These results analyze circuits generated for all security policies. Results for the `Position-Weighted` deployment are found in Figure 11. We iterate over weights w_e, w_m, and w_x, depending on the TEE distribution. Once again, these results only analyze circuits generated with the security policy synonymous with the deployment distribution. In each figure, 'baseline' represents the expected performance of users in Tor today, with no TEE requirements on circuits. To normalize

this value, we take the average across all the deployments, getting a final baseline expected bandwidth of 8347.3 KB/s.

Random. Performance results for this deployment can be found in Figure 10a. We see a steep increase immediately before tapering off with TEE-based entry security policy.

Notably, at only 20% TEEs in the network, circuits with a TEE-based entry relay requirement have a bandwidth of 7008.3 KB/s, a decrease of only 16% from the baseline. Considering more than half of the network is entry-capable, and entry relays have a minimum bandwidth they must meet, increasing p inherently increases the number of entry relays at a faster rate than exit relays. Congestion subsides quickly for TEE-based entry circuits, allowing performance of users to be only minimally reduced.

In comparison, the TEE-based exit security policy has a more steady increase in performance, though bandwidths are substantially lower. This is because there are fewer exit relays in the network in comparison to entry, resulting in more congestion. At 20% TEEs, circuits with a TEE-based exit relay have a 50.8% decrease in performance from the baseline. We see with the final two security policies performance continues to degrade as more TEE-based relays are required in the circuits, as to be expected. All circuits are using the same few TEE-based relays, resulting in more congestion.

Bandwidth Weighted. The results for `Bandwidth Weighted` deployment are shown in Figure 10b. This deployment shows the highest performance in comparison to other deployments. All security policies have a dramatic increase in performance from 1% TEE deployment until slowing around 25% TEE deployment. By requiring TEE-based relays in a circuit, this deployment provides the highest performance because users are biased towards the highest bandwidth relays, as these relays are TEE-based.

With only 10% TEEs in the network, TEE-based entry circuits nearly meet the baseline bandwidth with only a 3% decrease. At 44% TEEs, all security policies have a bandwidth meeting and/or exceeding the baseline. With these results, users do not need to sacrifice their performance for increased security by requiring TEE-based relays in circuits.

The most compelling result from this deployment, though, is after 10% TEEs in the network, TEE-based entry circuits surpass the baseline performance briefly, before slowing to baseline again around 70% TEEs. Due to all the TEEs being concentrated on the highest-bandwidth relays, the relay selection algorithm is even more biased towards high performance, which we see at these stages (we evaluate the privacy implications of this in the next section).

Inverse Bandwidth Weighted. We see a significant reduction in performance with `Inverse Bandwidth Weighted` deployment, seen in Figure 10c. Circuits with the TEE-based entry security policy perform the best, as with all other deployments. This is, once again, due to the substantial number of entry relays in the network. For TEE-based entry circuits to have at least half the bandwidth that is achieved by the baseline, 43% TEEs is required. For comparison, at this same TEE percentage, full TEE-based circuits in the `Random` deployment exceed this bandwidth. More than a 70% TEE presence is required for full TEE circuits to have this bandwidth.

Ultimately, this deployment scenario yields the most reduction in bandwidth compared to other deployments. This is to be expected, based on Tor's relay selection algorithm. As relays in this deployment have the least bandwidth in the network, requiring TEE-based relays, in turn, results in lower bandwidth.

Circuit-Position Weighted. The `Circuit-Position Weighted` deployment provides insight into which types of relays are most critical to performance.

The results for Entry distribution and Exit distribution are overlaid in Figure 11a. With this, the Entry distribution iterates over w_e, while the Exit distribution iterates over w_x. Once again, these distributions reflect the results of the `Random` deployment. With 20% TEE-based entry relays (w_e), circuits have a median bandwidth of 6941.7 KB/s, a 16.8% decrease from baseline. This reflects the fact that more than half of the relays in the network are entry-capable, meaning congestion begins to taper off around 20% TEEs. For the same decrease in performance, 54% of exit relays need to be TEE-based (w_x). A higher percentage of exit relays is required in comparison to entry relays because there are fewer exit relays in the network. Congestion is thus present at higher values of TEEs, as more circuits are vying for fewer existing TEE-based relays.

For the Entry-Exit distribution, shown in Figure 11b, one might expect to see for low w_e values similar results as in the previous two graphs. However, this is not the case, as we see when increasing the exit relays the bandwidth improves significantly, even with w_e at only 1%. This is because of the overlap in relays that are both entry and exit-capable. We see this with high w_x values, the network naturally load balances itself by using the excess TEE-based exit relays as entry relays. This same phenomenon is seen in the Entry-Middle-Exit distributions (Figure 11c, Figure 11e, Figure 11f). Despite the low values of w_e, as middle and exit relays gradually transition to adopting TEEs, the bandwidth of circuits increase because of the multi-capable relays in the network.

Takeaways. While one might expect to sacrifice performance to guarantee security with `ParTEETor` in policy mode, our performance evaluation shows this is not always the case. When requiring TEE-based relays in circuits, congestion persists only until TEE percentages reach a critical mass point. High bandwidth relays contribute most to reaching this point at sparse TEE deployments, as they can withstand the demand of circuits requiring TEE-based relays. Relays that can function in multiple positions of a circuit also impact performance positively. As they can be used in various positions, they can help meet the demand of circuits requiring TEE-based relays when a specific type of relay is limited. With relay operators that provide high bandwidth or that allow for multiple uses adopting TEEs, the network can meet demands without significantly reducing user performance.

6.4 Privacy of `ParTEETor` in Policy Mode

By specifying a security policy, the size of network is decreased, as circuits are limited to only containing TEE-based relays. Consequently, the space of available (policy-compliant) circuits is also reduced. When the space of possible circuits is reduced, the anonymity of users is impacted, as the predictability of circuits will be higher.

TEE Requirement	Number of Circuits	p = 1%	p = 5%	p = 10%	p = 25%	p = 50%	p = 75%
None	$x_c(e_c-1)(m_c-2)$	3.36×10^{10}					
Entry	$x_c(pe_c-1)(m_c-2)$	3.36×10^{8}	1.68×10^{9}	3.36×10^{9}	8.42×10^{9}	1.68×10^{10}	2.52×10^{10}
Exit	$px_c(e_c-1)(m_c-2)$	3.36×10^{8}	1.68×10^{9}	3.36×10^{9}	8.42×10^{9}	1.68×10^{10}	2.52×10^{10}
Entry, Exit	$px_c(pe_c-1)(m_c-2)$	3.36×10^{6}	8.42×10^{7}	3.36×10^{8}	2.10×10^{9}	8.42×10^{9}	1.89×10^{10}
Entry, Middle, Exit	$px_c(pe_c-1)(pm_c-2)$	3.36×10^{4}	4.21×10^{6}	3.36×10^{7}	5.26×10^{8}	4.21×10^{9}	1.42×10^{10}

Table 2: Number of unique circuits possible based on the ratio of TEE relays in the network and the security policy. p represents the percentage of TEE relays in the network and e_c, m_c, and x_c represent entry, middle, and exit capable relays, respectively.

Therefore, it is important to quantify the reduction in available circuits (and associated privacy) under specific TEE deployments and security policies. Through combinatorics, we derive the number of unique circuits that can be generated in `ParTEETor` in policy mode for each security policy, in partial TEE deployments. We juxtapose these results against historical Tor data to justify why even the reduced space of available (TEE-based) circuits still provides reasonable privacy guarantees.

Recall that circuit position capabilities of relays are not exclusive. All relays are middle-capable, regardless of their entry/exit capabilities. Some relays are both entry and exit-capable. We determine the breakdown of the number of relays with specific circuit position capabilities: Middle-capable relays: $m_c = 6356$, Entry-capable relays: $e_c = 3179$, Exit-capable relays: $x_c = 1668$. We denote p as the total percentage of TEE-based relays in the network, representing our deployment scenario. We consider deployments of 1%, 5%, 10%, 25%, 50%, and 75%. For each, we consider a uniform deployment amongst each relay capability. The formulas derived are found in Table 2, along with our results. In baseline Tor, more than 33 billion unique circuits are possible.

Our results show that requiring only one relay in a circuit to be TEE-based, whether it be entry or exit, provides the least reduction in availability, as expected–these TEE security policies place the least restrictions on possible circuits, while also mitigating two classes of attacks. With a 1% deployment, requiring a TEE-based entry relay results in only 1% of the possible circuits in comparison to no TEE requirement; however, this still results in 336 million possible circuits. For comparison, this space of circuits is equivalent to that of Tor in May of 2010 [35]. With a 25% deployment of TEEs, the space of circuits with both a TEE-based entry relay and TEE-based exit relay is two billion, which is equivalent to Tor in March of 2012. At a 50% deployment of TEEs, the space of full TEE circuits is equivalent to Tor in July of 2013, with four billion circuits possible.

Takeaways. Ultimately, our privacy results inform users on the trade-offs between security and privacy. By specifying a security policy, users are guaranteed protection from known classes of attacks, and receive reasonable privacy guarantees. While the space of circuits is significantly reduced, it can still meet or exceed the expected privacy of historical versions of Tor used by hundreds of thousands of people. Increased TEE penetration will gradually resolve this issue.

7 Discussion & Related Work

Limitations. While we demonstrate that TEEs are successful in defending against a broad class of attacks on Tor, we recognize their limitations: attacks against TEEs [13, 25, 30, 31] and defenses [1, 14, 38] have been published. Furthermore, TEEs can only reduce our adversary to a network level. The visibility that network administrators and ISPs have allows for more analysis on IP packets, which a local network adversary may not be able to analyze.

Relay Overloading. In this work, we do not consider relays reaching their bandwidth capacity. Consider the `Bandwidth-Weighted` deployment, for example. In this deployment, the highest performing relays are the most likely to be TEE-based. Due to Tor's relay selection algorithm, these relays are also most commonly used in circuits. This means that in this deployment, TEE-based relays will likely reach their bandwidth capacity because of their constant use. We leave to future work to model capacity limits of relays to better understand the impact on performance of users.

Implementing `ParTEETor` in Tor. We do not implement `ParTEETor` in the actual Tor network. We acknowledge SGX-Tor's [23] technical challenges of implementing the Tor protocol in a TEE, which have been identified and addressed. For this reason, implementing `ParTEETor` would largely reflect SGX-Tor, with only small modifications to the relay selection algorithm to support our security policies.

TEE-Augmented Tor. Other works have explored the use of TEEs for purposes largely unrelated to the attacks we evaluate here. For example, Panoply [39] considers in a case study implementing a directory authority within a TEE to to prevent adversaries from manipulating the consensus of the network. Jain *et al.* propose OpenSGX [17], a system that emulates SGX at an instruction level. In their evaluation, they are motivated in preventing attacks that exploit the private keys of directory authorities and exit nodes, in which they integrate the cryptographic functions of the Tor protocol into their system. ConsenSGX [37], considers the deployment of TEEs on directory cache relays for scalability problems. Clients are able to request only partial views of the network from TEE-enabled directory caches so they no longer need to store the entire consensus document. Note that none of these works consider the placement of the entire relay protocol within a TEE.

8 Conclusion

In this paper we have demonstrated that partial deployments of TEE-based relays (and with small changes to the Tor operation)

can substantially improve the resilience of the network to attacks. Moreover, our analysis shows that such security gains grow with the increasing penetration of TEEs. Thus, we argue that integration of TEEs is not only an obvious win, but necessary to Tor's future.

References

[1] Adil Ahmad, Byunggill Joe, Yuan Xiao, Yinqian Zhang, Insik Shin, and Byoungyoung Lee. 2019. OBFUSCURO: A Commodity Obfuscation Engine on Intel SGX. In *Proceedings 2019 Network and Distributed System Security Symposium*. Internet Society, San Diego, CA, 15 pages. https://doi.org/10.14722/ndss.2019.23513

[2] AMD. 2023. Using SEV with AMD EPYC™ Processors. https://www.amd.com/content/dam/amd/en/documents/developer/58207-using-sev-with-amd-epyc-processors.pdf

[3] Apple. 2024. Apple Platform Security: Secure Enclave. https://support.apple.com/guide/security/secure-enclave-sec59b0b31ff/web

[4] Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. 2007. Low-Resource Routing Attacks against Tor. In *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society* (Alexandria, Virginia, USA) *(WPES '07)*. Association for Computing Machinery, New York, NY, USA, 11–20. https://doi.org/10.1145/1314333.1314336

[5] Alex Biryukov, Ivan Pustogarov, and Ralf-Philipp Weinmann. 2013. Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization. In *2013 IEEE Symposium on Security and Privacy*. IEEE Computer Society, Oakland, California, 80–94. https://doi.org/10.1109/SP.2013.15

[6] Stevens Le Blond, Pere Manils, Abdelberi Chaabane, Mohamed Ali Kaafar, Claude Castelluccia, Arnaud Legout, and Walid Dabbous. 2011. One Bad Apple Spoils the Bunch: Exploiting P2P Applications to Trace and Profile Tor Users. In *4th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET 11)*. USENIX Association, Boston, MA.

[7] Chia che Tsai, Donald E. Porter, and Mona Vij. 2017. Graphene-SGX: A Practical Library OS for Unmodified Applications on SGX. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*. USENIX Association, Santa Clara, CA, 14 pages.

[8] George Danezis, Roger Dingledine, and Nick Mathewson. 2003. Mixminion: Design of a type III anonymous remailer protocol. In *2003 Symposium on Security and Privacy*. IEEE, IEEE, Berkeley, CA, 2–15.

[9] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The Second-Generation Onion Router. In *13th USENIX Security Symposium (USENIX Security 04)*. USENIX Association, San Diego, CA.

[10] Nathan S. Evans, Roger Dingledine, and Christian Grothoff. 2009. A Practical Congestion Attack on Tor Using Long Paths. In *Proceedings of the 18th Conference on USENIX Security Symposium* (Montreal, Canada) *(SSYM'09)*. USENIX Association, USA, 33–50.

[11] Michael J Freedman and Robert Morris. 2002. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*. ACM, Washington, DC, 193–206.

[12] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. 1996. Hiding Routing Information. In *Proceedings of the First International Workshop on Information Hiding*. Springer-Verlag, Berlin, Heidelberg, 137–150.

[13] Johannes Götzfried, Moritz Eckert, Sebastian Schinzel, and Tilo Müller. 2017. Cache Attacks on Intel SGX. In *Proceedings of the 10th European Workshop on Systems Security*. ACM, Belgrade Serbia, 1–6. https://doi.org/10.1145/3065913.3065915

[14] Gernot Heiser, Toby Murray, and Gerwin Klein. 2020. Towards provable timing-channel prevention. *ACM SIGOPS Operating Systems Review* 54, 1 (2020), 1–7.

[15] Intel. 2015. Intel® Software Guard Extensions. https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/overview.html.

[16] Intel. 2023. Intel® Processors Supporting Intel® SGX. https://www.intel.com/content/www/us/en/architecture-and-technology/software-guard-extensions-processors.html

[17] Prerit Jain, Soham Desai, Seongmin Kim, Ming-Wei Shih, JaeHyuk Lee, Changho Choi, Youjung Shin, Taesoo Kim, Brent Byunghoon Kang, and Dongsu Han. 2016. OpenSGX: An Open Platform for SGX Research. In *Proceedings 2016 Network and Distributed System Security Symposium*. Internet Society, San Diego, CA. https://doi.org/10.14722/ndss.2016.23011

[18] Rob Jansen, Kevin S Bauer, Nicholas Hopper, and Roger Dingledine. 2012. Methodically Modeling the Tor Network. In *5th Workshop on Cyber Security Experimentation and Test (CSET 12)*. USENIX Association, Bellevue, WA.

[19] Rob Jansen, Florian Tschorsch, Aaron Johnson, and Björn Scheuermann. 2014. The Sniper Attack: Anonymously Deanonymizing and Disabling the Tor Network. In *Proceedings 2014 Network and Distributed System Security Symposium*. Internet Society, San Diego, CA. https://doi.org/10.14722/ndss.2014.23288

[20] Damian Johnson. 2022. Directory — Stem 1.8.0 documentation. https://stem.torproject.org/api/directory.html.

[21] Tor Blog (juga). 2019. How Bandwidth Scanners Monitor The Tor Network. https://blog.torproject.org/how-bandwidth-scanners-monitor-tor-network/.

[22] David Kaplan, Jeremy Powell, and Tom Woller. 2016. AMD memory encryption. *White paper* 13 (2016), 12 pages.

[23] 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* 26, 5 (2018), 2174–2187. https://doi.org/10.1109/TNET.2018.2868054

[24] Albert Kwon, Mashael AlSabah, David Lazar, Marc Dacier, and Srinivas Devadas. 2015. Circuit Fingerprinting Attacks: Passive Deanonymization of Tor Hidden Services. In *24th USENIX Security Symposium (USENIX Security 15)*. USENIX Association, Washington, D.C., 287–302.

[25] Sangho Lee, Ming-Wei Shih, Prasun Gera, Taesoo Kim, Hyesoon Kim, and Marcus Peinado. 2017. Inferring Fine-grained Control Flow Inside SGX Enclaves with Branch Shadowing. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 557–574. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/lee-sangho

[26] Arm Limited. 2009. ARM Security Technology Building a Secure System using TrustZone Technology. https://developer.arm.com/documentation/PRD29-GENC-009492/latest/

[27] Arm Limited. 2017. TrustZone technology for ARMv8-M Architecture Version 2.1. https://developer.arm.com/documentation/100690/latest/

[28] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R Savagaonkar. 2013. Innovative instructions and software model for isolated execution. *Hasp@ isca* 10, 1 (2013), 8 pages.

[29] Bernard Ngabonziza, Daniel Martin, Anna Bailey, Haehyun Cho, and Sarah Martin. 2016. TrustZone Explained: Architectural Features and Use Cases. In *2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC)*. IEEE, Pittsburgh, PA, USA, 445–451. https://doi.org/10.1109/CIC.2016.065

[30] Alexander Nilsson, Pegah Nikbakht Bideh, and Joakim Brorsson. 2020. A Survey of Published Attacks on Intel SGX. *CoRR* (June 2020), 11 pages. http://arxiv.org/abs/2006.13598 arXiv:2006.13598 [cs].

[31] Oleksii Oleksenko, Bohdan Trach, Robert Krahn, Mark Silberstein, and Christof Fetzer. 2018. Varys: Protecting SGX Enclaves from Practical Side-Channel Attacks. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. USENIX Association, Boston, MA, 227–240. https://www.usenix.org/conference/atc18/presentation/oleksenko

[32] L. Overlier and P. Syverson. 2006. Locating hidden servers. In *2006 IEEE Symposium on Security and Privacy (S P'06)*. IEEE Computer Society, Oakland, California, 15 pp.–114. https://doi.org/10.1109/SP.2006.24

[33] Andriy Panchenko, Fabian Lanze, and Thomas Engel. 2012. Improving performance and anonymity in the Tor network. In *2012 IEEE 31st International Performance Computing and Communications Conference (IPCCC)*. IEEE, Austin, TX, USA, 1–10. https://doi.org/10.1109/PCCC.2012.6407715

[34] R. Pries, W. Yu, X. Fu, and W. Zhao. 2008. A New Replay Attack Against Anonymous Communication Networks. In *2008 IEEE International Conference on Communications*. IEEE, Beijing, China, 1578–1582. https://doi.org/10.1109/ICC.2008.305

[35] Tor Project. 2022. Tor Metrics. https://metrics.torproject.org/.

[36] Michael K Reiter and Aviel D Rubin. 1998. Crowds: Anonymity for web transactions. *ACM transactions on information and system security (TISSEC)* 1, 1 (1998), 66–92.

[37] Sajin Sasy and Ian Goldberg. 2019. ConsenSGX: Scaling Anonymous Communications Networks with Trusted Execution Environments. In *Proceedings on Privacy Enhancing Technologies*. De Gruyter Open, Stockholm, Sweden, 331–349. https://doi.org/10.2478/popets-2019-0050

[38] Sajin Sasy, Sergey Gorbunov, and Christopher W. Fletcher. 2018. ZeroTrace : Oblivious Memory Primitives from Intel SGX. In *Proceedings 2018 Network and Distributed System Security Symposium*. Internet Society, San Diego, CA, 15 pages. https://doi.org/10.14722/ndss.2018.23239

[39] Shweta Shinde, Dat Le, Shruti Tople, and Prateek Saxena. 2017. Panoply: Low-TCB Linux Applications with SGX Enclaves. In *24th Annual Network and Distributed System Security Symposium*. Internet Society, San Diego, California. https://doi.org/10.14722/ndss.2017.23500

[40] Robin Snader and Nikita Borisov. 2008. A Tune-up for Tor: Improving Security and Performance in the Tor Network. In *Proceedings 2008 Network and Distributed System Security Symposium*, Vol. 8. Internet Society, San Diego, California, 127.

[41] torproject. 2022. Tor Specifications. https://github.com/torproject/torspec.

[42] Bassam Zantout, Ramzi Haraty, et al. 2011. I2P data communication system. In *The Tenth International Conference on Networks (ICN 2011)*. ICN, Springer Singapore, The Netherlands, 401–409.