Sajin Sasy* and Ian Goldberg*

ConsenSGX: Scaling Anonymous Communications Networks with Trusted Execution Environments

Abstract: Anonymous communications networks enable individuals to maintain their privacy online. The most popular such network is Tor, with about two million daily users; however, Tor is reaching limits of its scalability. One of the main scalability bottlenecks of Tor and similar network designs originates from the requirement of distributing a global view of the servers in the network to all network clients. This requirement is in place to avoid *epistemic attacks*, in which adversaries who know which parts of the network certain clients do and do not know about can rule in or out those clients from being responsible for particular network traffic.

In this work, we introduce a novel solution to this scalability problem by leveraging oblivious RAM constructions and trusted execution environments in order to enable clients to fetch only the parts of the network view they require, without the directory servers learning which parts are being fetched. We compare the performance of our design with the current Tor mechanism and other related works to show one to two orders of magnitude better performance from an end-to-end perspective. We analyse the requirements to actually deploy such a scheme today and conclude that it would only require a small fraction (<2.5%) of the relays to have the required hardware support; moreover, these relays can perform their roles with minimal network bandwidth requirements.

Keywords: Anonymous Communications Network, Tor, Secure Hardware

DOI 10.2478/popets-2019-0050

Received 2018-11-30; revised 2019-03-15; accepted 2019-03-16.

***Corresponding Author: Sajin Sasy:** Cheriton School of Computer Science, University of Waterloo, ssasy@uwaterloo.ca

***Corresponding Author: Ian Goldberg:** Cheriton School of Computer Science, University of Waterloo, iang@uwaterloo.ca

1 Introduction

Privacy is an integral right of every individual in society [72]. With almost every day-to-day interaction shifting towards using the internet as a medium, it becomes essential to ensure that we can maintain the privacy of our actions online. Furthermore, in light of nation-state surveillance and censorship, it is all the more important that we enable individuals and organizations to communicate online without revealing their identities. There are a number of tools aiming to provide such private communication, the most popular of which is the Tor network [21].

Tor is used by millions of people every day to protect their privacy online [70]. Tor is a low-latency anonymous communication network that enables people to communicate online without revealing what they are communicating about and with whom. It is designed to enable its users to beat an adversary that monitors a part of the internet, by making connections through a series of virtual tunnels instead of direct connections. Tor is used today by whistleblowers, journalists, law-enforcement, government organizations, and many others [71].

However, Tor is currently reaching the limits of its scalability. In order to make a secure and private connection through Tor, a client constructs a three-hop *circuit* through a sequence of Tor *relays*. A client must have complete freedom in the selection of these three hops lest the client be susceptible to *epistemic attacks* [18, 19] that could deanonymize them. In an epistemic attack, an adversary learns, through some means, that particular clients do not know about all of the relays in the network, and importantly, the adversary knows which relays particular clients do know. The adversary then uses this knowledge to rule in or out specific clients from possibly being responsible for specific circuits. In order to enable clients to freely select the nodes for its connection, currently Tor requires each client to maintain a global view of the Tor network. The distribution of this global view of the network requires significant network bandwidth: if the number of relays

in network (and thus the capacity) scales linearly with the number of active users, then the bandwidth required to send information about all relays to all clients scales *quadratically* with the number of users. In previous work, McLachlan et al. [43] showed that at the then-current rate, Tor would spend more network bandwidth distributing this global view rather than for anonymous communication itself.[1] In addition, simply *storing* the complete current consensus could be problematic for mobile or embedded Tor clients.

In this work, we propose a novel solution for enabling clients of anonymous communications networks to pick relays for building secure circuits, without requiring the clients to maintain a global view of the network, and while avoiding epistemic attacks. Our solution takes advantage of the performance benefits of pairing Oblivious RAMs (ORAMs) [26] with trusted execution environments (such as Intel SGX [3]) as shown in recent works [2, 45, 60]. Our contributions are threefold:

- We design ConsenSGX, to enable clients to build secure circuits in anonymous communications networks without holding a global view of the state of the network.
- We implemented and evaluated our design on real hardware, and provide the results from our experiments in Section 5.
- We compare our work in terms of end-to-end latency against 1) previous designs towards the same goal to show one to four orders of magnitude improvement and 2) Tor's current mechanism to show one to two orders of magnitude improvement.

In Section 2, we first describe Tor's current network consensus distribution mechanism in detail, followed by other essential background information required for our proposal. In Section 3, we discuss other related proposals that aim to achieve better scalability for anonymous communication networks, followed by Section 4, where we discuss our proposal in depth. Section 5 describes the details of our implementation and evaluation and Section 6 addresses deployment details and considerations. Finally, Section 7 concludes this work.

2 Background

2.1 Tor

The Tor network (as of November 30, 2018) consists of about 6300 volunteer-run servers called *onion routers* or *relays*. In order to make a secure and private connection through Tor, a user makes a three-hop connection or *circuit* through these relays. Currently, there are nine dedicated servers called the *directory authorities*, which host and periodically update an authenticated copy of the global view of this network called the *network consensus*. The network consensus lists the relays currently available for making circuits, their IP addresses and ports, their cryptographic keys, their bandwidths for load-balancing purposes, and other such pertinent details.

In order to generate this network consensus, every relay in the Tor network, at the start of every *epoch* (one hour window) uploads a self-signed *relay descriptor* to each of the nine directory authorities. This descriptor describes the relay's keys, capabilities, and other additional information. The directory authorities then hourly generate their individual views of the current network status and descriptors of routers in the network, and then amongst themselves perform a consensus protocol, the output of which is the network consensus which contains all of the relay descriptors of active relays in an epoch, signed by the directory authorities. Every client in the Tor network then downloads this document once per epoch from a Tor *directory cache*, which are mirrors that host the signed network consensus for global distribution.

Tor relays in a circuit can have three different roles. The *guard relays* serve as the first hop of a circuit. Clients use a fixed set of relays as its guard relays, and hence the guard relay selection process is largely a one-time operation [23, 52, 80], but each client refreshes its guard set every 2 months [69]. Relays must be stable and have sufficiently high bandwidth (2 Mbps) [69] in order to be eligible to be a guard relay. *Exit relays* serve as the final hop of a circuit, and are the relays that communicate directly with the internet services that the Tor client is accessing. As the exit relay operators

1 However, we note that Tor switched to a more efficient mechanism for distributing this global view since McLachlan et al.'s work; specifically the microdescriptor model, which we elaborate on in Section 2.1. However even with the microdescriptor model, a straightforward calculation shows that the fraction of network bandwidth taken up by distributing microdescriptor consensuses is still about $N * 10^{-5}$, where N is the number of relays in the network. At the current size of 6300 relays this is a little more than 6%, with the percentage itself linearly increasing as the network grows.

sometimes have to handle complaints about Tor users, only a small fraction (about 800 out of 6300) of relays allow themselves to be used as exit relays. Those that do can set *exit policies* that describe IP addresses and/or ports to which they will or will not make connections. Relays that do not serve as either guard or exit relays can still serve as *middle relays*, acting as the middle hop of a circuit.

In order to build a circuit, a client has to pick a node from each of the above sets of relays, and build a telescoping three-hop connection through them to its destination. As mentioned before, currently Tor requires each client to download the network consensus, to ensure clients have complete autonomy in the selection of its relays. However this solution has poor scalablility; as more routers join this network, this network consensus grows larger. As more clients join Tor, more network consensuses have to be distributed via this network, thus leading to a large portion of the Tor network bandwidth being used up in simply distributing network consensuses.

There are several different ways the existing Tor network serves network consensuses to Tor clients. Currently Tor supports 28 different flavours of these network consensus documents [69]. Most of these variants are still maintained to continue supporting older Tor clients. Broadly, these methods can be classified into two main categories: the traditional server descriptor consensus and the more recent microdescriptor consensus.

Server Descriptor Consensus: This is the legacy form of network consensus documents. In this model, the network consensus contains the entire router descriptor of every valid router in the network for the given epoch. The clients then at every epoch downloads this network consensus in its entirety for obtaining the global view of the Tor network.

Microdescriptor consensus: The microdescriptor consensus mechanism aimed to reduce the bandwidth consumed in transmitting network consensus documents. The key observation here was that most relay descriptors remained unchanged across epochs. Hence the network consensus could become more lightweight by not repeating these descriptors in its entirety. The network consensus in this model has a reduced descriptor for each active router and a hash of the microdescriptor of the router. The microdescriptor of the router contains the entire body of the router descriptor, but clients only need to download them when their local hash mismatches with the one published for a router in an epoch.

In both of these broad methods, there are "diff" variants, which enable the clients to download just the differences (the diff) between their outdated local network consensus and the current network consensus. Diffs of consensuses are only served for clients that have a local state less than 5 hours apart from the current epoch. If the required diff is too outdated and hence not served by the directory caches, the client defaults to downloading the entire network consensus.

While the microdescriptor consensus mechanism slows the growth in bandwidth consumed by the consensus distribution as the network grows, it is not a long-term solution to this problem. There are two vital flaws in this current design, namely:

– Every client is required to know every relay.
– Every directory cache is required to know every relay.

Both of these are scalability throttles for the Tor network as noted in Tor's directory protocol v3 documentation [69]. Many privacy-preserving communications networks should aim to defend against epistemic attacks, including Tor, I2P [82], and modern mixnets like Loopix [55]. When any of these networks grow substantially, they must be redesigned in a fashion that does not mandate clients to possess the entire global view of the network. This work focuses on solving the former part of the problem, which is currently more pressing, considering the significantly larger number of clients than relays today in the Tor network.

2.2 Trusted Execution Environments (TEEs)

Over the last decade, hardware-aided trusted execution environments have been refined by industry and academic research. The initial designs like Intel's Trusted Execution Technology (TXT) [29] and AMD's Platform Security Processor (PSP) used Trusted Platform Modules (TPM) to provide measurements of software and platform components. However, such designs implied poor hardware utilization due to two main factors; first, the TPM had to vet the entire platform before it could instantiate a TEE. Second, once a TEE was instantiated, other applications could not be interleaved with trusted executions. Moreover, TXT was also plagued with security vulnerabilities [78, 79] due to memory leaks and its trusted System Management Mode assumptions.

More recent advances such as ARM TrustZone [5], AMD Secure Encrypted Virtualization (SEV) [32], and Intel SGX [3] embed cryptographic keys at manufacture time into the processor, which enable these processors to instantiate a TEE through modifications to their instruction set. Intel 7th generation processors (Skylake and onwards) have support for SGX (Software Guard eXecution) [3], which are extensions to the x86 instruction set to enable execution of secure code remotely via secure containers called *enclaves*. At a high level, the cryptographic keys fused in the processor at the time of manufacture enable the processors to set aside a portion of their memory as PRM (Processor Reserved Memory). Secure code to be executed within enclaves are loaded page by page into the PRM, preserving confidentiality and integrity using enclave-specific derived keys from the keys fused in the processor. After loading a program, SGX produces an *enclave measurement*, which is the hash of the program that was loaded into the enclave, thus allowing the remote user to verify the integrity of the program that was loaded into the enclave.

All memory contents that reside in the PRM are encrypted and integrity-verified (in hardware) using the aforementioned cryptographic keys. While different enclaves share the PRM, they store their pages using enclave-specific keys derived from the fused keys. The instruction set ensures that only enclaves that have the expected measurement can access these pages. In order to establish trust in a remote SGX-enabled machine, a user must perform a *remote attestation* protocol with it [30]. As of August 2018, this protocol involves verifying an ECDSA signature produced by one of Intel's special *architectural enclaves*, the Provisioning Certification Enclave (PCE). The PCE generates the private key for this signature from the fused key in the processor. The corresponding public key for verification is distributed by Intel in the form of an X.509 certificate.

While these TEEs have promising and desirable properties, we note that it is no panacea to generic secure remote computations. Since its inception, research has shown SGX to be susceptible to several side channel attacks [12, 36, 62, 63, 81]. However, previous works have also shown that applying known side-channel proofing techniques [51, 56] can help preserve security in specific application scenarios.

2.3 ORAM

ORAM or Oblivious RAM [26] is a cryptographic primitive introduced by Goldreich and Ostrovsky in 1993 as a countermeasure to information leakage to an adversary able to observe memory access patterns. Over the last two decades, research has led to several refined and improved constructions of ORAMs [20, 24, 58, 65, 66, 77]. However, ORAMs still remained confined to theoretical realms; primarily, this can be attributed to the large overheads introduced by this primitive. In order to provide access to memory "obliviously", Goldreich and Ostrovsky [26] prove a lower-bound logarithmic overhead with respect to the underlying memory.

ORAM schemes can be broadly classified into tree-based and hierarchical constructions. In this work we limit ourselves to tree-based constructions since they are simpler and more efficient. At a high level, in a tree-based ORAM scheme, one stores the underlying data in the form of a binary tree on the server side, where each node in the tree has a mix of real and dummy blocks encrypted under a probabilistic encryption scheme, and clients store a mapping from block identifiers to leaves of the tree. These constructions maintain the invariant that a block in the system will be present somewhere on the path leading from the root to the leaf the block is mapped to. In order to query data obliviously, ORAM schemes require clients to expend both storage and computation resources; the client-side storage maintains the block-to-leaf mapping and also stores 'overflow' blocks. Computation at the client side involves decrypting, reordering, and encrypting paths of the tree. In order to perform an oblivious access, the client queries for the path that the required block maps to from the server, decrypts the path to extract the required block from the path, and then reorders, re-encrypts, and returns this freshly encrypted path back to the server to replace the old one.

In practice this results in two major overheads: multiple network roundtrips[2] and significant client-side computation and/or memory. However, with the advent of trusted execution environments like Intel SGX [3], recent works [2, 45, 60] have shown that one can leverage such trusted execution environments to efficiently and securely instantiate ORAMs with overheads that are practically viable. With such a system, ORAM protocols get reduced to a single round trip request-response protocol, with only trivial client-side computation and

2 In order to maintain a small client state, ORAMs require multiple network roundtrips. This can be reduced to a single round trip at the expense of a much larger client state, which is typically unusable in real-world deployments.

memory overheads, at the expense of trust in a secure hardware module.

In our work, we make use of the Circuit ORAM [77] implementation in ZeroTrace [60]. We do not elaborate in detail on ZeroTrace here, but refer the interested reader to the original paper. In short, ZeroTrace is a doubly oblivious ORAM; that is, it fits the ORAM client logic (called the *ORAM controller*) into a secure hardware module, but given that Intel SGX is known to be susceptible to side-channel attacks [12, 36, 62, 63, 81], Sasy et al. modify the ORAM controller logic itself to be oblivious via linear scans and oblivious assembly-level functions through use of the x86 CMOV instruction. In our work, we make use of ZeroTrace as a black-box tool which enables secure and practical ORAM deployments via secure hardware modules.

We note that although our implementation uses Intel SGX as the underlying secure hardware module, our techniques are not tied to any unique property of SGX itself and in fact are generic enough to be realized by any trusted execution environment.

3 Related Work

In this section we discuss previous research towards scaling anonymous communication networks, which can be broadly classified into two directions: predominantly through peer-to-peer techniques, all of which were shown to be susceptible to a variety of attacks, and PIR-Tor, which is the only other proposal that can be classified as a client-server model. Here we do not consider Tor's own optimizations (that we discussed earlier in Section 2.1) that improve scalability through constant factors, but instead we focus on other works that try to improve the underlying asymptotic complexity. We use the same taxonomy described by PIR-Tor [48] to classify and discuss relevant related work in this area.

3.1 Peer-to-peer Models

Peer-to-peer models aim to build anonymous connections via circuits or tunnels constructed with limited knowledge of the global network. Each node in this model knows only about a small number of other nodes, called *neighbours*. In its simplest form, a client would initiate circuits through one of its neighbouring nodes, and then extend these circuits from the current last hop to one of the last hop's neighbours. However, this introduces an avenue for malicious nodes to perform route capture attacks, which are attacks that subvert these circuits by extending circuits only through colluding malicious nodes. Hence most of these proposals focus on reducing the impact of such colluding adversaries.

Additionally, peer-to-peer networks also enable Sybil attacks [22], hence almost all of these proposals rely on the assumption that an adversary would not be able to control nodes that are spread across global IP subnets; i.e., they use diversity of IP subnets to reduce the impact of Sybil attacks.

3.1.1 Structured Peer-to-peer Networks

Structured peer-to-peer topologies are predominantly distributed hash table (DHT) based topologies, in which every node is initially assigned a list of neighbours using a mathematical function based on a node identifier (typically its IP address), so that initial neighbour assignments are easily verifiable. However, routing and lookup mechanisms in DHTs are extremely vulnerable to attacks from malicious nodes in the system [39, 74].

In order to secure the lookup mechanism, Halo [31] and Salsa [50] proposed using redundant routing lookups. The idea is to perform multiple redundant lookup queries through probabilistically different paths in the network so as to reduce the impact of malicious nodes subverting a lookup. Mittal and Borisov [47] showed that these techniques of secure lookups for node selection are susceptible to information leak attacks, which are typically amplified by the redundancy employed.

Similarly, Panchenko et al. proposed NISAN [53], which is based on Chord [67], and used redundant lookups as well. NISAN attempted to mitigate information leak attacks by not revealing the lookup destination to intermediate nodes but instead downloading and processing the entire routing tables of intermediary nodes locally. Torsk [43] proposed constructing circuits by starting with the peer-to-peer models of Kademlia [42] and Myrmic [75], and imbuing it with an additional 'secret buddy' mechanism to break correspondence attacks while extending circuits. However, Wang et al. [76] showed that NISAN was still susceptible to information leak attacks which can be fine tuned by adversaries through a range estimation process to deanonymise traffic. Furthermore, they also demonstrated attacks against discovering these secret buddy nodes in Torsk, which eventually lead to user deanonymisation.

3.1.2 Random Walk Based Architectures

In random walk based peer-to-peer models, there is no well-defined initial assignment of neighbours for nodes, but instead nodes get to know about other nodes in the system through either some form of "information servers" that know of recently active nodes or by trying nodes from their local cache. Anonymous circuits are then constructed by making a random walk through these nodes and their individual neighbour lists.

In MorphMix [59] circuits are extended from the current last hop with the help of a randomly chosen 'witness' by the user. In order to prevent malicious colluding entities from breaking the anonymity of the system, the extender has to provide a list of its neighbouring nodes amongst which the witness randomly chooses the node for the next hop in the circuit. In addition, nodes use a tailored collusion detection mechanism (CDM) to avoid using nodes that they deem malicious and colluding. However, Tabriz and Borisov [68] showed weaknesses in the collusion detection mechanism and how an adversary can in fact model the user's internal CDM tables to avoid being detected and continue subverting a majority of its circuits.

Mittal and Borisov proposed ShadowWalker [46], which introduced the notion of 'shadows' for a node. A shadow of a node is responsible for verifying the node's neighbour information, and it does so by providing a digital signature of the routing table of that node. In contrast to the witness in a MorphMix system, these shadows need not be directly contacted by the initiator of a circuit, instead the shadows provide the nodes it monitors with the digital signature which can be passed on to the initiator of a circuit extension step without requiring separate lookups on the initiator's part, thus reducing the information leak attack surfaces. However, Schuchard et al. [61] showed that this design opens to another set of attacks, namely 'eclipse' attacks, where if an adversary can acquire a neighbourhood (node and all its shadows) composed entirely of adversary nodes then he can corrupt honest nodes' routing tables. The natural defence would be to increase the neighbourhood making it harder for adversaries to acquire such clusters, but this defence enables adversarial nodes to launch DoS attacks on honest nodes by refusing to issue valid digital signatures, demonstrating an innate tension between resistances to these two attacks. Their work also provides a defence by optimizing this trade-off but at the same time incurring significant overheads.

While peer-to-peer models have innate scalability benefits, as we described above they open up several other attack vectors.

3.2 Client-Server Model

Private Information Retrieval (PIR) allows a client to retrieve information from an online database without the database learning what was being requested. Mittal et al. proposed PIR-Tor [48], using PIR techniques [25, 44] for securely distributing relay descriptors from a network consensus. Instead of serving the entire network consensus to clients, the directory caches act as PIR servers, and clients then look up individual relay descriptors from these caches. Importantly, this technique avoids epistemic attacks, because no one other than the client itself knows *which* relay descriptors were fetched. In order to avoid malicious directory caches serving false data, the network consensus is modified to have *individually signed* relay descriptors which are used as PIR data blocks by the directory caches. By having each block contain a signature from the directory authorities over the contents of the descriptor, the timestamp of the consensus, and the block number within the consensus, modification, substitution, and replay attacks by the (untrusted) directory caches are prevented. Moreover, Mittal et al. note that these PIR queries are only required for the middle and exit relays, since clients use a fixed set of guard relays to enter the network [23, 80]. The paper considers two flavours of PIR—information-theoretic PIR (IT-PIR) and computational PIR (CPIR)—but discards the ITPIR option despite having better performance due to its requirement of non-colluding PIR servers which are practically hard to ensure without selecting from a small number of centralized trusted servers such as the directory authorities. Our proposal borrows several building-block ideas from PIR-Tor which translate well even to the ORAM setting, such as BLS signatures [10] to minimize the size overhead induced by individually signing relay descriptors and limiting directory cache queries to just middle and exit relays. We describe more details of PIR-Tor in Section 5, when we evaluate the performance of our proposal, and compare our system to it.

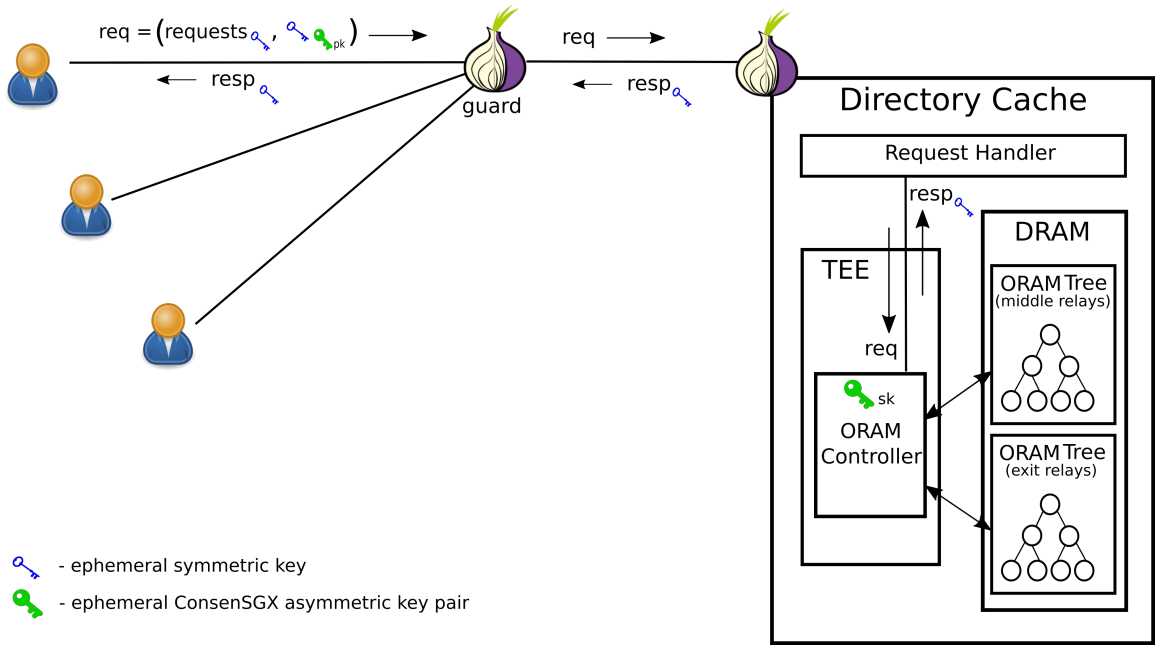

Fig. 1. Overview of the ConsenSGX architecture. When a key is used as a subscript of an element, we imply it is an encryption of that element with the corresponding key. Clients make directory requests through a one-hop circuit through their guard relay. Requests are hybrid encryptions of a sequence of relay indices they wish to fetch. Responses are the corresponding signed relay descriptors encrypted with the ephemeral key used for the request. Note that the secret key of the ephemeral ConsenSGX asymmetric key pair is generated within the TEE and never leaves the TEE unencrypted. The client learns this key (authenticated by the long-term ConsenSGX public key) when it creates the one-hop circuit; see Section 4.2.2.

4 ConsenSGX

In order to enable clients to efficiently construct anonymous circuits without possessing the entire network view, we propose that clients use an ORAM protocol to fetch only small portions of the network consensus, but to do so *obliviously*; i.e., without revealing which parts of the network they learned about to passive network adversaries or a malicious directory cache (similar to what PIR-Tor did with PIR instead of ORAM). The challenge we face is that while PIR protocols are inherently *multi-client*, ORAM protocols are typically *single-client*, since the client needs to store the decryption keys for the ORAM tree contents as well as the block-to-leaf mapping and overflow blocks. We address this challenge using ZeroTrace [60], which implements the ORAM controller *inside a trusted execution environment that runs on the server itself*. The actual clients of the protocol (the Tor clients) then make encrypted connections into the TEE so that the server cannot observe the contents of the requests or the replies. ZeroTrace is also itself oblivious, as mentioned above, so that the server cannot learn the internal state of ZeroTrace by observing its behaviour.

In Figure 1, we illustrate a high-level overview of our system architecture. Just like Tor today, clients make directory queries from a directory cache using a one-hop circuit through its guard node. For every epoch, TEE-supported directory caches instantiate ORAM controllers with the network consensus of that epoch as the underlying data. Clients can then perform batched queries with these ORAM controllers to obliviously retrieve portions of the network consensus. Importantly, the multiple round trips required by ORAM protocols in this setup become round trips between the ORAM controller (within the TEE) and the DRAM (dynamic RAM) on the same machine, and not network round trips.

We list the design goals of our system below, and then elaborate on the finer details of the architecture.

4.1 Design Goals

– *Scalability*: The current Tor design for distributing network consensus information has a bandwidth complexity that is linear in both the number of relays in the network, and also the number of clients. Thus assuming that the number of relays

scales linearly with the user base (in order to provide enough capacity to support those users), this results in a bandwidth complexity that scales quadratically with number of users. In our system this bandwidth overhead is constant (and *small*) per client, and so linear over the whole network, with a poly-logarithmic computational overhead at the directory caches with respect to number of relays.

– *Efficiency*: Our system makes efficient fetches for relay descriptors, with minimal computational overheads of a few milliseconds, which is significantly overshadowed by Tor network throughputs. This efficiency enables particularly lightweight deployments in scenarios where only a few circuits are needed. For instance, private browsing modes for Mozilla and Brave which plan to use Tor circuits [13, 49]; in this context we envision these browsers need to construct just a handful of circuits corresponding to a user's private tabs, hence downloading an entire network consensus is intuitively wasteful.[3]
– *Minimal Changes*: Our proposal can be weaved into Tor as another network consensus mechanism, without engineering efforts that require redesigning Tor's core components. This allows our proposal to be incrementally deployed on the real Tor network.

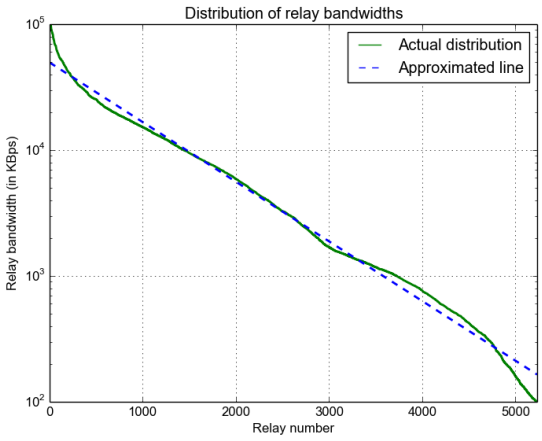

Fig. 2. Bandwidth distribution graph of relays in Tor as of September 2018. We truncate the tail of the actual distribution, which constitutes several relays that have less than 100 KBps bandwidth, which are never selected by clients to construct circuits, as they lack the "Fast" flag in the consensus.

4.2 System Architecture

4.2.1 Server Side

The consensus protocol of Tor remains the same for our proposal. Once consensus is achieved, the directory authorities release multiple variants of the network consensus as mentioned before. For the directory authorities we require two new changes; first, they have to publish two new variants of the network consensus document, a network_consensgx, which contains a record for each relay in the network, sorted by decreasing bandwidth, and params_consensgx, which contains a concise summary of the number and bandwidth distribution of the relays in the network. Each record in the network_consensgx also contains a signature by the directory authorities over the relay descriptor, its index in the network_consensgx, and the timestamp of the epoch. The combination of timestamp and relay index ensures that a directory cache cannot deviate from serving the correct network_consensgx without being detected by clients.

Once the network_consensgx is published, the directory caches that serve the consensus instantiate two ORAM trees populated with the signed relay descriptors for the middle and exit relays respectively. Recall that the guard relay selection process is performed only rarely (on the order of months), so simply downloading the full network consensus only at those times (or even just the guards) is reasonable.

For relay selection, Tor uses a bandwidth-weighted sampling mechanism biased towards higher-bandwidth relays to select a random relay from the pool of middles and exit relays for constructing a circuit. The params_consensgx document published by the directory authorities contains all of the parameters of the network that are not specific to a particular relay, including various protocol versions, flags to enable network features, etc.; this data is less than 4 KB in size. The params_consensgx also lists the number of relays in the middle and exit pool, the distributions of bandwidths for the relays in these pools, and the exit policies (or a hash thereof; see Section 4.3 below) currently associated to active exit relays in the network. The objective of this document is to enable clients to pick their relays obliviously and correctly with respect to the clients' desired properties for the relay just like how a client would do using the current Tor consensus mechanism. In order to do so, we note that the bandwidth distribution of relays is (as of September 2018) captured by an

3 At the time of writing Mozilla's project Fusion (Firefox USIng ONions) is still ongoing work, while Brave has already rolled out Tor circuits for private browsing tabs.

exponential curve, which can be closely approximated by a linear line on a linear-log plot of relay number to relay bandwidth as shown in Figure 2. The directory authorities would then have to compute and publish the slope and intercept of this line in params_consensgx to enable clients to sample relays correctly.

In order to deal with potential changes in this distribution we propose that the directory authorities select from a few common distributions (such as exponential, Pareto, etc.) to best parameterize the empirical bandwidth distribution. There are two things to note here with respect to mismatches in the empirical and best-fit bandwidth distribution: first, this process has no implications on security, but it is simply a mechanism that enables better load distribution of clients across relays; second, the bandwidth measurements that Tor uses are themselves noisy. In a recent proposal [41], Mathewson outlines a related approach that works for arbitrary distributions: our proposal above has the individually signed descriptors implicitly containing the index of the descriptor (and the client chooses the index from a parameterized distribution). Mathewson proposes that the individually signed descriptors *explicitly* contain a range of values that a *uniformly* sampled 32-bit integer would fall into in order to select that descriptor. The client then just chooses that uniform integer, regardless of the distribution. This proposal effectively increases descriptor sizes by eight bytes, but has the benefit of being agnostic to the shape of the empirical distribution.

While the params_consensgx needs to be distributed to every client at the start of every epoch, it is extremely lightweight and can be maintained at a constant size irrespective of the total number of relays in the system.

The second main change for the directory authorities is to verify the authenticity of caches that serve ConsenSGX. In order to provide the list of directory caches that serve ConsenSGX, the directory authorities have to verify the remote attestations produced by these caches which will include the caches' ConsenSGX public keys, but we note that this is only required when a new directory cache with ConsenSGX support joins the network.[4] These public keys are long-lived *signature verification* keys; we make this choice in order to hedge against possible future compromises of the TEE architecture that might expose the TEE's secrets. We do not want such an exposure to reveal past client lookups. The ConsenSGX enclave then periodically creates a fresh ephemeral asymmetric encryption keypair, signing the encryption key with its long-lived signature key.

Once the directory authorities validate the attestation, they can then provide the relay descriptors (including the CongenSGX public keys created by these enclaves) in the network_consensgx, thus allowing clients to query them.

In order to bootstrap this system, clients need to know the set of directory caches that support ConsenSGX queries; observe that just like the guard relay selection process this is a rare operation. Once a client has a set of such directory caches locally available from a one-time download of the entire network consensus, the clients can update them at a later point, for example, when among the local set of directory caches, the amount of unreachable ones reaches a predefined threshold. This update can then itself be done through ConsenSGX queries.

Unlike the current consensus methods of Tor, in ConsenSGX each of the relay descriptors must be individually signed by all of the directory authorities rather than bulk signed, since clients now retrieve only a single descriptor at a time and have to validate the authenticity of this descriptor in the current epoch. As mentioned before, PIR-Tor also faced this same issue and proposed BLS signatures [10] as the signature scheme for this purpose, due to its small signature size as a signature is just a single group element in this scheme. Further, although the PIR-Tor system did not use this fact, BLS signatures can also be *aggregated* [9], so that a single group element can be used to capture *all* of the directory authorities' signatures. Hence we use BLS aggregated signatures for each block in the network_consensgx.

4.2.2 Client Side

Once the directory authorities converge on the network_consensgx and have instantiated the aforementioned ORAM trees, clients can make oblivious relay descriptor fetches from those directory caches that support ConsenSGX. As in the current Tor protocol, clients contact directory caches over a single-hop Tor circuit consisting only of the client's guard. In order to save a network round trip, we could slightly extend the circuit-creation protocol to have the directory

4 This verification of TEE-aided directory caches is the only component that will change for adapting our proposal to other TEE architectures such as ARM TrustZone or AMD SEV. Supporting a variety of TEE architectures can avoid reliance on a monoculture and enhance security through platform diversity.

cache proactively send its current ConsenSGX-enclave asymmetric encryption key, signed by its long-term ConsenSGX-enclave public verification key, if requested in the circuit creation message by the client. Clients, already possessing the params_consensgx, locally select the *indices* of which relay descriptors to fetch, and simply send those indices, *encrypted to the directory cache's ConsenSGX enclave public key*, over that short circuit. We note that the params_consensgx also enables selection from the available relay descriptors on the basis of either the current Tor router selection algorithm or the Snader-Borisov criterion [64]; this choice is orthogonal to our goal of reducing the bandwidth consumed in constructing Tor circuits. Other router selection algorithms have been proposed [73]; to use ConsenSGX with such proposed algorithms, one could put the summary information about the consensus necessary for the algorithms into the params_consensgx. If those algorithms require effectively the entire consensus, then they themselves are a scalability bottleneck that ConsenSGX cannot alleviate.

4.3 Server Descriptors Analysis

In order to select the appropriate block size for our scheme, we analysed the relay descriptor sizes from the network consensus of the first hour of September 25, 2018. Figure 3 shows that there is a large variance in relay descriptor sizes. Looking closer at the cause of this variance, we note that the *exit policy* and *family* parameters are the main contributors. An exit policy is a list of whitelisted/blacklisted IP addresses and ports for exit relays, while the family is a list of other relays that are owned by the same entity. Exit policies are required for a client to build a circuit that can connect to the desired server host and port, and the objective of the family field is to build circuits with relays that are owned by different entities to reduce the chance of possible deanonymization.

We observe that these two fields change more slowly than other attributes of the descriptor; moreover, in the long run, the existence of very specific exit policies can itself become a deanonymizing attribute, and hence it would be ideal to limit exit policies (at least at the port level) to selected exit policy sets. In Figure 4, we plot the same histogram without these two fields and note that all relay descriptors can be upper-bounded by blocks of 2800 bytes. In order to account for the overhead from the BLS signature[5] from the directory authorities, and some extra headroom, we chose a block size of 3000 bytes for all our experiments. In order to pick a uniform block size for these relay descriptors, we propose moving the exit policies and families into a separate document that can be served if the policy/family details were to change, by merely appending a hash of this document with every params_consensgx. We note that there are other possible optimizations for compressing this policy list itself to further decrease this additional bandwidth, but we leave this as future work.

4.4 Security Trade-off

In addition to the standard Tor threat model, ConsenSGX introduces the security assumption of trusting the underlying TEE that is used for deploying ConsenSGX in exchange for efficiency. Specifically what this implies is that we trust that processors with TEEs have their hardware-fused cryptographic keys generated in a secure fashion, and that the APIs exposed for these TEEs perform exactly what their specifications claim and are bug-free, and finally also that they do not have any malicious backdoors inserted into them by the vendor. A violation of this trust would allow a ConsenSGX-enabled directory cache to potentially see that some client (but not *which* client) was looking up a particular set of server descriptors, opening the door for an epistemic attack.

As explained in Section 4.2.1, by separating the *long-term signature verification keys* and the *ephemeral asymmetric encryption key pair*, we provide forward secrecy that prevents a malicious processor vendor from inserting a "retroactive" backdoor, but nonetheless by design choices of our scheme we cannot undo an already-existing backdoor in the processor.

In the context of Intel's SGX in particular, this perhaps not as bad as it might seem, but only because a significant portion of processors in the wild are Intel manufactured, and systems using these processors are *already* implicitly reliant on Intel's processors, Intel Management Engine (ME), Intel's hardware random number generator (RDRAND), etc. not being backdoored. We discuss in more detail the ramifications of trusting trusted hardware in Section 6.1.

5 A BLS signature at the 128-bit security level is 461 bits [6, Table 14], or 58 bytes, taking into account the recent

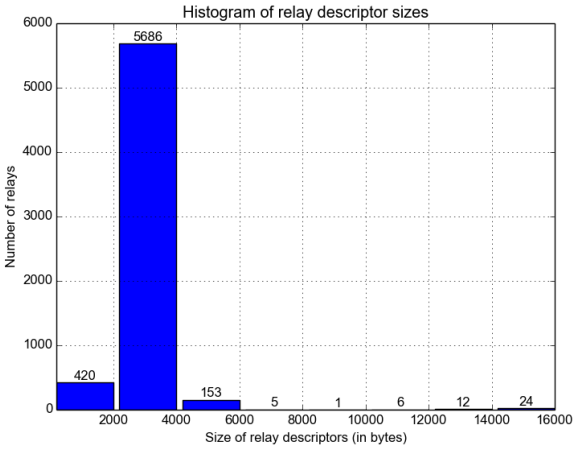

Fig. 3. Histogram of relay descriptor sizes.

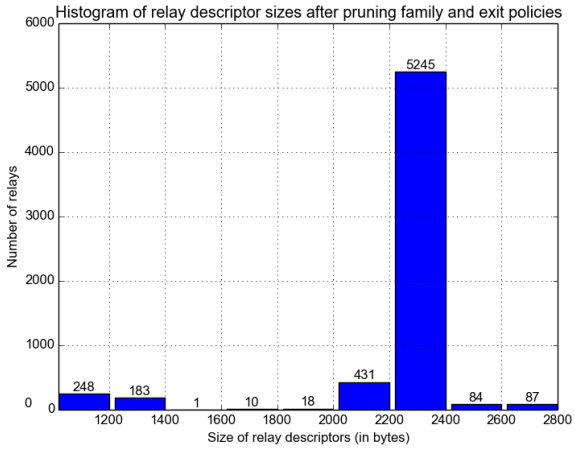

Fig. 4. Histogram of relay descriptor sizes after pruning *exit policy* and *family*.

5 Evaluation

As mentioned before, since P2P proposals have all been shown to susceptible to attacks by the security community [39, 47, 61, 68, 74, 76], we compare our proposal primarily against PIR-Tor, which to our knowledge is the only other scheme for scaling anonymous communications that has its security tied to a well-understood cryptographic protocol (PIR). The particular computational PIR scheme [44] used in PIR-Tor was later shown to be susceptible to attacks [37], in part due to the vulnerabilities of using non-standard underlying security assumptions [7, 8, 16]. Therefore, we compare ConsenSGX with PIR-Tor, but substituting XPIR [1] as the underlying PIR protocol instead. XPIR is the state-of-the art computational PIR scheme, using the BV homomorphic encryption scheme [11], which is based on standard Ring-LWE [57] assumptions and hence believed to be secure.

For completeness we also compare ConsenSGX against PIR-Tor's information-theoretic PIR (ITPIR) model with the simplest (and most efficient) XOR-based ITPIR scheme by Chor et al. [15] as the underlying PIR scheme. For this best-case performance setup for the scheme we compare our results to, we made use of the Percy++ v1.0.0 library. However, just like PIR-Tor, we rule out ITPIR in practice due to the three non-colluding server assumption which is hard to realize. In our experiments we consider a single server that hosts the corresponding PIR server (in the case of PIR-Tor) or ORAM tree (in the case of ZeroTrace). We vary the number of Tor relays in the network; these experiments do not differentiate between middle and exit relays, but we introduce this aspect later in this section.

All of our experiments are run on a server-grade Intel Xeon E3-1270, with four physical cores, 64 GB of DDR4 RAM, and support for Intel SGX. Our server machine runs Ubuntu 16.04, and our experimental results are all for a single core without any parallelism for all the systems we measure. Therefore, our measurements are per-core values, and multiple cores can be used trivially to serve independent clients in parallel, for example. Our experimental source code is available on our website.[6]

We evaluate the performance of these proposals by empirically evaluating how they would perform as we scale the Tor network to much larger sizes than they are today. To evaluate fairly, we analyse all facets of the proposals; i.e., client computation overhead, server computation overhead, bandwidth requirements, and an evaluation of end-to-end performance with practically reasonable bandwidth choices. We envision that in deployments of these proposals, the client would batch multiple requests in an epoch instead of querying for each circuit individually, thus we evaluate performance with varying choices of B, which denotes the number of relays requested in a query. We perform our experiments with two values of B: $\mathsf{B} = 10$ and $\mathsf{B} = 50$. We assume 10 circuits an hour to be representative of a low-usage

improvements in finite field discrete logarithm computations by Kim and Barbulescu [34].

6 https://crysp.uwaterloo.ca/software/consensgx/

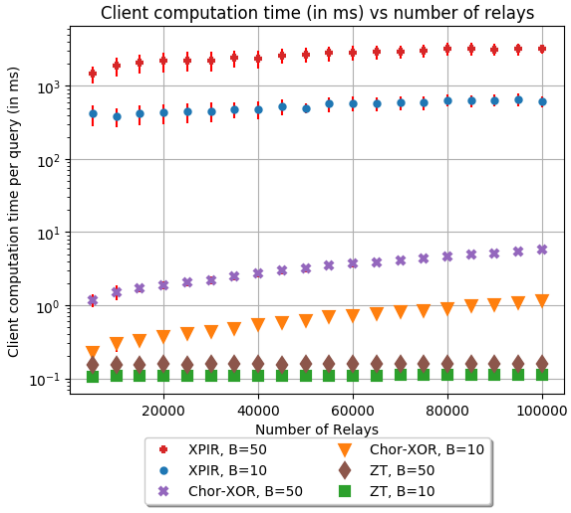

Fig. 5. Client computation time (in ms) for performing a query for B relay descriptors; i.e., cost of encrypting a query as well as decrypting the response received. This graph includes error bars for all points, but they may be too small to be seen.

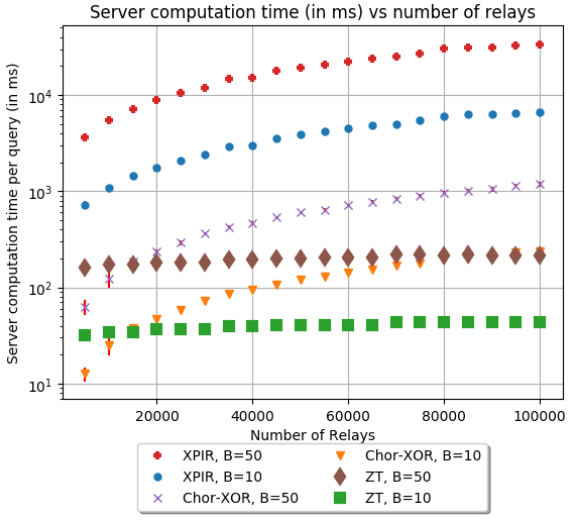

Fig. 6. Server computation time (in ms) for serving a received query for B relay descriptors. Note that these performance numbers are from a single core execution and without any parallelism. This graph includes error bars for all points, but they may be too small to be seen.

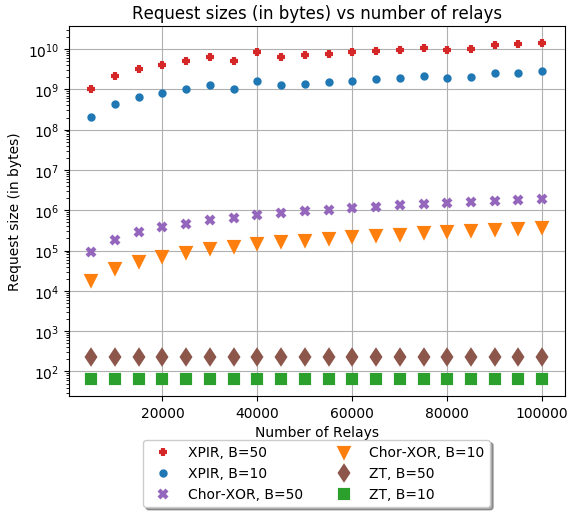

Fig. 7. Request size (in bytes) for a query to fetch B relay descriptors. There are no error bars because these values are all constant.

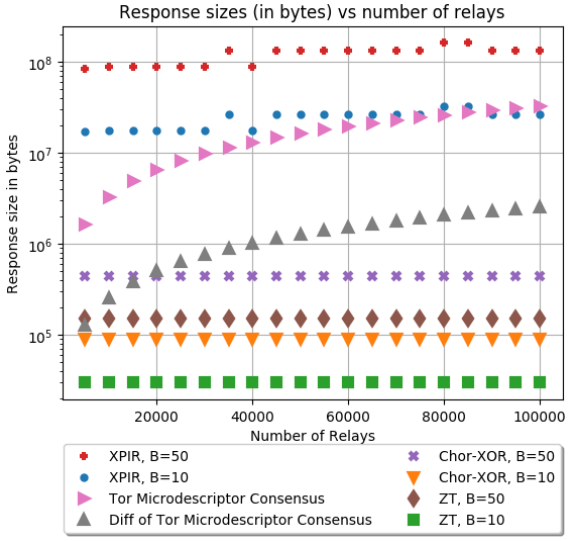

Fig. 8. Response size (in bytes) for a query to fetch B relay descriptors. This graph also shows the sizes for Tor's microdescriptor consensus and its diff variant; these values are extrapolated from the current consensus size. There are no error bars because these values are all constant.

client, and 50 circuits an hour to represent a heavy-usage client.

In Figure 5, we compare the client's computation overhead from both PIR-Tor and ConsenSGX. ConsenSGX outperforms by several orders of magnitudes in terms of client computation requirements, since when using ZeroTrace the client query is a single hybrid encrypted request; i.e. a single AES-GCM encryption of all of the ORAM indices it wishes to retrieve from the server, followed by one elliptic curve public key

encryption of the AES-GCM key.[7] We note that the client computation for our proposal is dominated by the single public key operation (which takes close to 0.1 ms). Since AES instructions are heavily optimized by the AES-NI [27] instructions today, we can perform an AES encryption in a few processor cycles. For computational PIR-Tor, XPIR has to perform a lattice encryption of an array of size proportional to the total number of relays, and this induces significant overheads as seen in Figure 5. Information-theoretic PIR-Tor has overheads that are significantly lower than its computational counterpart since it does not have to perform lattice encryptions, but this too scales poorly since the request sizes are still very large. Therefore, ConsenSGX is innately more suited for lightweight personal devices such as mobile phones, tablets, etc.

In terms of the server computation required, we see in Figure 6 that ConsenSGX is at least an order of magnitude faster than the PIR-Tor counterparts. This arises from the difference in asymptotic complexity of the two schemes. For a network of N relays to service a single query, a PIR server has to perform $O(N)$ work, while a ZeroTrace ORAM controller has to perform $(\log^3(N))$ computation. Amongst the IT-PIR and CPIR variants of PIR-Tor, we notice that the IT-PIR variant performs much better; this is expected since it has significantly lesser computation overheads being XOR-based as opposed to lattice computations (at the cost of requiring a non-collusion assumption amongst the PIR servers used). We allow XPIR's optimizer module choose the best parameters for d (recursion levels) and α (aggregation factor) in all of our experiments, given the block size and bandwidth constraints.

Figures 7 and 8 compare the request and response sizes involved in querying these schemes. The request size is orders of magnitude smaller for ConsenSGX since, as mentioned above, the query is simply an AES-GCM encryption of B indices, whereas XPIR queries contain B lattice encryptions of an N-length bit array. Similarly the response sizes are also much smaller for ConsenSGX, since it is an AES-GCM encrypted blob of B relay descriptors, whereas the BV encryptions of relay descriptors are much larger as seen in the graph. We note that the request sizes and server computation time for PIR-Tor with IT-PIR could be optimized a bit further by using query batching techniques as shown by Lueks and Goldberg [40].

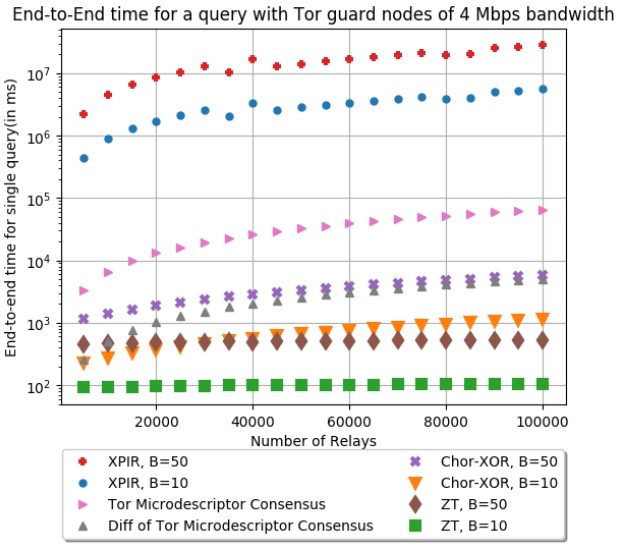

Fig. 9. End-to-end time taken for a query of B relay descriptors, assuming the guard relays can provide a client with a link of 4 Mbps bandwidth. This graph includes error bars for all but the Tor lines, but they may be too small to be seen.

Figure 8 also shows the bandwidth requirements of Tor's microdescriptor scheme and the diff variant of the microdescriptor scheme. In order to fairly analyze these two schemes, we ran experiments to analyse the amount of relay descriptors that change in each epoch, over the first six months of consensus data from Tor metrics [70] for 2018. The microdescriptors over this period have an average size of 660 bytes and on average 1.49% of them change every epoch. The lines for these two schemes take into account this additional bandwidth overhead in maintaining the global view assuming that this fraction stays the same with increase in the number of relays. This graph clearly illustrates the bandwidth savings our proposal introduces in comparison with PIR-Tor and Tor's current mechanism.

Finally, Figure 9 shows the time taken in an end-to-end setting for these schemes. Since consensus downloads and descriptor fetches in the Tor network currently use a one-hop path through the guard relay, the bandwidth we consider for the end-to-end setting should ideally be the average bandwidth available at a guard relay in the network. We instead approximate this bandwidth with double of Tor's current average network bandwidth, which is 2 Mbps [70]. We intentionally choose a much higher bandwidth than that available today at Tor relays, since the schemes we compare ConsenSGX against do benefit from higher bandwidths, although ConsenSGX itself uses the least bandwidth as shown by Figures 7 and 8.

7 We use the NIST P-256 (also known as secp256r1 or prime256v1) curve for this purpose.

We note that this graph does not distinguish between middle and exit relays; instead it shows the end-to-end time in downloading B relay descriptors from a set of N relays. Hence, in practice the total cost would be the sum of time taken for a query each to the pool of middle relays and exit relays respectively. For instance, today the Tor network has approximately 800 exits and the remaining 5500 serve as guards and middles; scaling that up to a total of 100,000 relays while preserving the ratio of exit to middle relays, implies a query to a pool with 13,000 exit relays, which takes 472.2 ms ± 0.4 ms end-to-end time assuming a query with $B = 50$, and another to a pool with 87,000 middle relays, which takes 517.8 ms ± 0.5 ms, resulting in a total time of 990 ms ± 0.6 ms, or barely one second. For the same setting, PIR-Tor's CPIR variant takes 4430.0 s ± 0.8 s and 20295 s ± 2 s for querying the middle and exit pool respectively, resulting in a total time of 24725 s ± 2 s. In contrast, PIR-Tor's IT-PIR model for the same queries takes 1.40 s ± 0.03 s and 5.1 s ± 0.05 s respectively, resulting in a much more reasonable total of 6.5 s ± 0.06 s. Tor's current microdescriptor mechanism, when scaled to 100,000 relays, takes 65.46 s and the diff variant takes 5.1 s.

In our experiments we use XPIR as the benchmark for CPIR, but we note that there have been recent advancements in CPIR constructions since XPIR, namely SealPIR [4] and PSIR [54]. Although we do not perform empirical comparisons against these two schemes in this work, here we provide intuitions as to why these schemes cannot enable scalability to the extent that ConsenSGX does. In comparison to XPIR, SealPIR reduces the client computational costs and query size due to their query compression technique, and further it even speeds up server-side computation time for batched query processing by their use of probabilistic batch codes. However, they still have large response sizes similar to that of XPIR, which are larger than the current trivial download mechanism that Tor uses (as illustrated by Figure 8). Similarly, while PSIR may seem asymptotically interesting for query processing,[8] the construction practically requires streaming the entire underlying database whenever the database changes (or when a threshold of queries have been made), which in our setting negates any of the asymptotic benefits

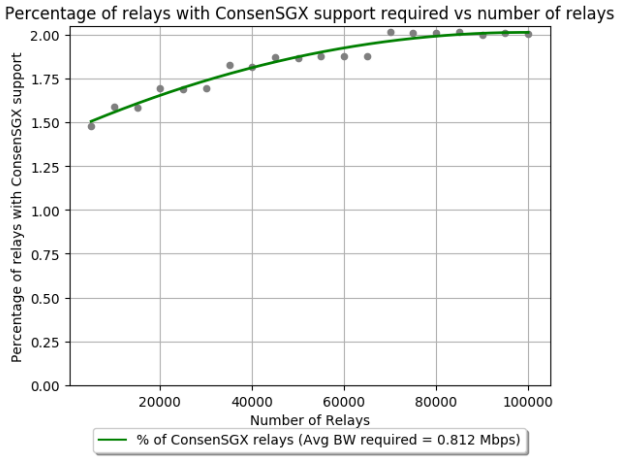

Fig. 10. Percentage of relays that need to support ConsenSGX to distribute $B = 50$ relay descriptors every epoch to all clients; we assume the number of clients scales linearly with the number of relays, maintaining the current client-to-relay ratio (2000:6). When we say the relay has ConsenSGX support we only assume it contributes one of its cores towards serving ConsenSGX requests.

gained from the lower query processing times and is strictly worse than just the current trivial download.

While PIR-Tor proposed a promising solution for scaling anonymity networks like Tor, it incurs significant overheads when used in practice as illustrated by Figure 9. On the other hand, our comparisons with all the other schemes towards the same goal shows that ConsenSGX can be deployed today to gain significant bandwidth savings along with the lowest end-to-end latencies, as demonstrated by Figures 7 to 9. From Figure 9, the only competitive system for ConsenSGX at today's network sizes is Tor's microdescriptor diff model, but ConsenSGX outperforms it quickly as the network size grows; moreover, while the diff variant of the microdescriptor scheme seems practically viable and desirable, it has a few subtle drawbacks; note that its advantages only apply to a user who is continuously online. These performance benefits disappear once a user goes offline for a few (currently five) hours and Tor defaults back to the regular microdescriptor model. Moreover, this also introduces a timing side channel for an adversary, and although we are yet to see any strong deanonymization attacks from this channel, this can reduce the anonymity set of a user.

8 Unlike the standard $\mathcal{O}(N)$ query processing time in PIR systems, PSIR has an $\mathcal{O}(\sqrt{N})$ asymptotic processing time for queries.

6 Deployment Considerations

From a deployment perspective, in order to avoid the potential hazards of introducing additional code for pairing-based cryptography within Tor, we also propose an alternative Merkle Tree based signature mechanism which can simply use Tor's current Ed25519 signature scheme. Specifically, instead of signing each relay descriptor individually, the directory authorities would construct a Merkle tree with the hash of each relay descriptor in the epoch as leaf nodes; the root of this Merkle tree is then published in the params_consensgx document, which is signed by all the directory authorities. The directory caches that serve ConsenSGX should then first reconstruct this Merkle tree from the network_consensgx and verify that the root matches the one published in the params_consensgx. These directory caches can then append each relay descriptor from the middle and exit relays in the network_consensgx with the Merkle validation path from that relay's leaf node to the root of the Merkle tree. Any client that then retrieves a relay descriptor from a ConsenSGX-serving directory cache, can verify the integrity of this relay descriptor by verifying its path on the Merkle tree.

While TEEs such as Intel SGX and TrustZone are now available on commodity hardware today, it would be ambitious to expect all existing relays to immediately support it. Hence we evaluate the number of directory caches with support for TEE required to serve all the clients in the system. For this, we assume that the ratio of clients to relays in the system remains the same as the network size grows; i.e., the current ratio of 2,000,000 clients to an approximate 6000 relays is maintained. We use the server computation overheads from Figure 6 for serving one client with $\mathsf{B} = 50$ on a single core of a relay acting as a directory cache, to estimate how many such relays we would need to serve all clients in an epoch. Figure 10 plots the percentage of relays with secure hardware support required against the total number of relays in the system. We observe that this percentage is relatively small, and consistently less than 2.5% of the total relays in the network. Note that all our choices of parameters are intentionally conservative; for instance, we only consider a single core per relay that is used towards serving ConsenSGX on these directory caches, and a value of $\mathsf{B} = 50$ which is generous as well, since as mentioned earlier this value of B accounts for heavy-usage clients.

A deployment question to address is that a number of Tor relays are actually deployed as virtual machines running in hosting facilities. While in this work we do not empirically evaluate ConsenSGX for virtual hosts, we note that SGX does currently have support for virtualization for KVM and Xen hypervisors [28]. On the other hand AMD's TEE, AMD Secure Encrypted Virtualization (SEV) [32] is designed exactly for this use case of initializing VMs on a remote untrusted host, while maintaining confidentiality and integrity guarantees.

Another interesting facet of our proposal is the low bandwidth overhead it has. We observe that even relays with relatively low network bandwidths can be used as directory caches for serving ConsenSGX, since as seen in Figure 10, the average bandwidth required is 0.812 Mbps. This provides a new avenue for low-bandwidth relays that are barely used for serving anonymous communications[9] to contribute more productively to the network. For instance, from Figure 2, we notice that out of Tor's 6300 relays today, only about 5200 of them have the "Fast" flag and are used towards building anonymous circuits; the other 1000 relays (about 17.4% of the network) have less than 100 KBps (= 0.8 Mbps), which is approximately the average bandwidth requirement[10] for ConsenSGX, and also accounts for a significantly larger fraction than what is needed to serve the entire client set today through ConsenSGX as shown by Figure 10.

6.1 Trusting Trusted Hardware

As mentioned before, secure hardware modules are known to have their own set of security vulnerabilities [12, 36, 62, 63, 81]. More recently, CacheQuote [17] showed side-channel leakages in Intel's *architectural enclaves*, which are special enclaves that perform the attestations, and Foreshadow [14] adapted the out-of-order execution flaws demonstrated in Spectre [35] and Meltdown [38] towards attacking SGX enclaves. Both these works succeed in extracting the private attestation keys from Intel's architectural enclaves, thereby dismantling its security guarantees.

9 Since relays for constructing a circuit are sampled using a bandwidth-weighted sampling mechanism, these are used rarely in practice.

10 Note that relays that have bandwidths lower than this can also serve as ConsenSGX directory caches, but the network would require more of such directory caches then to compensate.

While these attacks are currently pressing, fortunately the defences for these have been shown by the respective works and Intel is working towards patches for the same. Fortunately, mitigating these vulnerabilities does not require a complete redesign of the hardware architecture. Hence this should be deemed as part of the natural life cycle for such a hardware component. Furthermore, we are also beginning to see more promising open-source secure hardware modules like Keystone [33] surfacing, which will significantly further research in this space.

In our particular context, we note that ConsenSGX is not innately tied to Intel SGX, but can be instantiated through any hardware-aided TEE. We simply use SGX to implement our proof-of-concept for the design. We also note that compromising the secure hardware module that serves ConsenSGX does not directly compromise users' anonymity online, but it makes users susceptible to epistemic attacks or route fingerprinting attacks. These attacks as we alluded to before, aim to deanonymize users by having information about which relays these users know of or do not know of. In the context of a malicious adversary running compromised hardware that serves users via ConsenSGX, this adversary can then learn which B relays were fetched. But there are two caveats to note here; first, since users' make this fetch through a one hop circuit through their guard nodes, this implies that this adversary can only know the set of relay descriptors this guard relay fetched, and not end users directly. Second a client or user can lessen the impact of this attack by increasing this parameter B, as this would make it harder for this adversary to observe all traffic on all possible B relays that this user could use for their circuits. While this is intuitively sound, we have not yet seen any research in this direction of reducing impact of epistemic attacks, but there are promising practical implications from such a line of research.

7 Conclusion

In this work we described ConsenSGX, a novel technique for scaling anonymous communication networks. This technique can be used as a generic form of defence against epistemic attacks in any anonymity network, without enforcing clients to maintain a global view of the network. We evaluated our work by computing the performance benefits of using it with the most popular anonymity network used today, Tor, and also against other relevant proposals towards the same goal of scaling anonymous networks. Our experiment results demonstrated the performance benefits of our proposal; even when deployed today it can improve bandwidth consumption by an order of magnitude.

While our implementation currently uses Intel SGX as the underlying TEE, the ConsenSGX scheme is agnostic to this choice, and can easily be adapted for use with any other TEE. By enabling better scalability for Tor and similar anonymity networks, this work acts an incremental step towards enabling user's privacy online. There remains several future steps in this direction of research to enable anonymous communication networks to support every individual.

Ackowledgements

We thank our shepherd Nick Hopper for guiding this paper towards final acceptance. We thank the anonymous reviewers for their helpful suggestions, and particularly for the Merkle Tree variant discussed in Section 6. The authors thank NSERC for grants RGPIN-03858 and STPGP-463324, and the Royal Bank of Canada for funding this research.

References

[1] C. Aguilar-Melchor, J. Barrier, L. Fousse, and M.-O. Killijian. XPIR: Private Information Retrieval for Everyone. *Proceedings on Privacy Enhancing Technologies*, 2016.

[2] A. Ahmad, K. Kim, M. I. Sarfaraz, and B. Lee. OBLIVIATE: A Data Oblivious Filesystem for Intel SGX. In *25th Network and Distributed System Security Symposium (NDSS)*, 2018.

[3] I. Anati, S. Gueron, S. Johnson, and V. Scarlata. Innovative Technology for CPU Based Attestation and Sealing, 2013. https://software.intel.com/en-us/articles/innovative-technology-for-cpu-based-attestation-and-sealing.

[4] S. Angel, H. Chen, K. Laine, and S. Setty. PIR with compressed queries and amortized query processing. In *39th IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2018.

[5] ARM. ARM Security Technology: Building a Secure System using TrustZone Technology, 2015. http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf.

[6] R. Barbulescu and S. Duquesne. Updating Key Size Estimations for Pairings. Cryptology ePrint Archive, Report 2017/334, 2017.

[7] J. Bi, M. Liu, and X. Wang. Cryptanalysis of a homomorphic encryption scheme from ISIT 2008. In *IEEE International Symposium on Information Theory (ISIT)*,

2012.

[8] D. Bleichenbacher, A. Kiayias, and M. Yung. Decoding of Interleaved Reed Solomon Codes over Noisy Data. In *International Colloquium on Automata, Languages, and Programming*. Springer, 2003.

[9] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2003.

[10] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2001.

[11] Z. Brakerski and V. Vaikuntanathan. Efficient Fully Homomorphic Encryption from (Standard) LWE. *SIAM Journal on Computing*, 2014.

[12] F. Brasser, U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun, and A. Sadeghi. Software Grand Exposure: SGX Cache Attacks Are Practical. In *11th USENIX Workshop on Offensive Technologies (WOOT)*, 2017.

[13] Brave. Brave Private Tabs with Tor. https://brave.com/tor-tabs-beta, Accessed September 2018.

[14] J. V. Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx. Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution. In *27th USENIX Security Symposium*, 2018.

[15] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private Information Retrieval. In *IEEE Foundations of Computer Science (FOCS)*, 1995.

[16] D. Coppersmith and M. Sudan. Reconstructing Curves in Three (and Higher) Dimensional Space from Noisy Data. In *35th ACM Symposium on Theory of Computing (STOC)*, 2003.

[17] F. Dall, G. De Micheli, T. Eisenbarth, D. Genkin, N. Heninger, A. Moghimi, and Y. Yarom. Cachequote: Efficiently Recovering Long-Term Secrets of SGX EPID via Cache Attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018.

[18] G. Danezis and R. Clayton. Route Fingerprinting in Anonymous Communications. In *6th IEEE International Conference on Peer-to-Peer Computing*. IEEE, 2006.

[19] G. Danezis and P. Syverson. Bridging and Fingerprinting: Epistemic Attacks on Route Selection. In *8th Privacy Enhancing Technologies Symposium (PETS)*. Springer, 2008.

[20] S. Devadas, M. van Dijk, C. W. Fletcher, L. Ren, E. Shi, and D. Wichs. Onion ORAM: A Constant Bandwidth Blowup Oblivious RAM. In *Theory of Cryptography Conference (TCC)*. Springer, 2016.

[21] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *13th USENIX Security Symposium*, 2004.

[22] J. R. Douceur. The Sybil attack. In *International Workshop on Peer-to-Peer Systems*. Springer, 2002.

[23] T. Elahi, K. Bauer, M. AlSabah, R. Dingledine, and I. Goldberg. Changing of the Guards: A Framework for Understanding and Improving Entry Guard Selection in Tor. In *11th ACM Workshop on Privacy in the Electronic Society (WPES)*. ACM, 2012.

[24] C. Fletcher, M. Naveed, L. Ren, E. Shi, and E. Stefanov. Bucket ORAM: Single Online Roundtrip, Constant Bandwidth Oblivious RAM. Cryptology ePrint Archive, Report 2015/1065, 2015.

[25] I. Goldberg. Improving the Robustness of Private Information Retrieval. In *28th IEEE Symposium on Security and Privacy (S&P)*, 2007.

[26] O. Goldreich and R. Ostrovsky. Software Protection and Simulation on Oblivious RAMs. *Journal of the ACM (JACM)*, 1996.

[27] S. Gueron. Intel® Advanced Encryption Standard (AES) New Instructions Set, 2010. https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf.

[28] Intel. SGX Virtualization. https://01.org/intel-software-guard-extensions/sgx-virtualization. Accessed February 2018.

[29] Intel. Intel Trusted Execution Technology, 2007. http://www.intel.com/technology/security/. Accessed September 2018.

[30] Intel. Software Guard Extensions (Intel® SGX) Data Center Attestation Primitives: ECDSA Quote Library API, 2018. https://download.01.org/intel-sgx/dcap-1.0/docs/SGX_ECDSA_QuoteGenReference_DCAP_API_Linux_1.0.pdf.

[31] A. Kapadia and N. Triandopoulos. Halo: High-Assurance Locate for Distributed Hash Tables. In *16th Network and Distributed System Security Symposium (NDSS)*, 2008.

[32] D. Kaplan, J. Powell, and T. Woller. AMD Memory Encryption, 2016. https://developer.amd.com/wordpress/media/2013/12/AMD_Memory_Encryption_Whitepaper_v7-Public.pdf.

[33] S. Karandikar, S. Devadas, A. Ou, K. Asanovic, I. Lebedev, D. Song, and D. Lee. Keystone Open-source Secure Hardware Enclave, 2018. https://keystone-enclave.org/. Accessed September 2018.

[34] T. Kim and R. Barbulescu. Extended Tower Number Field Sieve: A New Complexity for the Medium Prime Case. In *CRYPTO*. Springer, 2016.

[35] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom. Spectre Attacks: Exploiting Speculative Execution. In *40th IEEE Symposium on Security and Privacy (S&P)*, 2019.

[36] S. Lee, M.-W. Shih, P. Gera, T. Kim, H. Kim, and M. Peinado. Inferring Fine-grained Control Flow Inside SGX Enclaves with Branch Shadowing. In *26th USENIX Security Symposium*, 2017.

[37] T. Lepoint and M. Tibouchi. Cryptanalysis of a (Somewhat) Additively Homomorphic Encryption Scheme Used in PIR. In *3rd Workshop on Applied Homomorphic Cryptography and Encrypted Computing (WAHC)*, 2015.

[38] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg. Meltdown: Reading Kernel Memory from User Space. In *27th USENIX Security Symposium*, 2018.

[39] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. *IEEE Communications Surveys & Tutorials*, 2005.

[40] W. Lueks and I. Goldberg. Sublinear Scaling for Multi-Client Private Information Retrieval. In *International Conference on*

Financial Cryptography and Data Security (FC). Springer, 2015.

[41] N. Mathewson. Proposal 300: Walking Onions: Scaling and Saving Bandwidth. https://gitweb.torproject.org/torspec.git/tree/proposals/300-walking-onions.txt, 2019. Accessed February 2019.

[42] P. Maymounkov and D. Mazieres. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *International Workshop on Peer-to-Peer Systems*. Springer, 2002.

[43] J. McLachlan, A. Tran, N. Hopper, and Y. Kim. Scalable Onion Routing with Torsk. In *16th ACM Conference on Computer and Communications Security (CCS)*, 2009.

[44] C. A. Melchor and P. Gaborit. A Fast Private Information Retrieval Protocol. In *IEEE International Symposium on Information Theory (ISIT)*, 2008.

[45] P. Mishra, R. Poddar, J. Chen, A. Chiesa, and R. A. Popa. Oblix: An Efficient Oblivious Search Index. In *39th IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2018.

[46] P. Mittal and N. Borisov. ShadowWalker: Peer-to-peer Anonymous Communication using Redundant Structured Topologies. In *16th ACM Conference on Computer and Communications Security (CCS)*, pages 161–172. ACM, 2009.

[47] P. Mittal and N. Borisov. Information Leaks in Structured Peer-to-Peer Anonymous Communication Systems. *ACM Transactions on Information and System Security (TISSEC)*, 2012.

[48] P. Mittal, F. Olumofin, C. Troncoso, N. Borisov, and I. Goldberg. PIR-Tor: Scalable Anonymous Communication Using Private Information Retrieval. In *20th USENIX Security Symposium*, 2011.

[49] Mozilla. Fusion: Firefox USIng Onions, 2018. https://wiki.mozilla.org/Security/Fusion. Accessed September 2018.

[50] A. Nambiar and M. Wright. Salsa: A Structured Approach to Large-Scale Anonymity. In *13th ACM Conference on Computer and Communications Security (CCS)*, 2006.

[51] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa. Oblivious multi-party machine learning on trusted processors. In *25th USENIX Security Symposium*, 2016.

[52] L. Øverlier and P. Syverson. Locating Hidden Servers. In *IEEE Symposium on Security and Privacy (S&P)*, 2006.

[53] A. Panchenko, S. Richter, and A. Rache. NISAN: Network Information Service for Anonymization Networks. In *16th ACM Conference on Computer and Communications Security (CCS)*, 2009.

[54] S. Patel, G. Persiano, and K. Yeo. Private Stateful Information Retrieval. In *25th ACM Conference on Computer and Communications Security (CCS)*, 2018.

[55] A. M. Piotrowska, J. Hayes, T. Elahi, S. Meiser, and G. Danezis. The Loopix Anonymity System. In *26th USENIX Security Symposium*, 2017.

[56] A. Rane, C. Lin, and M. Tiwari. Raccoon: Closing Digital Side-channels Through Obfuscated Execution. In *24th USENIX Security Symposium*, 2015.

[57] O. Regev. On Lattices, Learning With Errors, Random Linear Codes, and Cryptography. *Journal of the ACM (JACM)*, 2009.

[58] L. Ren, C. Fletcher, A. Kwon, E. Stefanov, E. Shi, M. van Dijk, and S. Devadas. Constants Count: Practical Improvements to Oblivious RAM. In *24th USENIX Security Symposium*, 2015.

[59] M. Rennhard and B. Plattner. Introducing MorphMix: Peer-to-Peer Based Anonymous Internet Usage with Collusion Detection. In *1st ACM Workshop on Privacy in the Electronic Society (WPES)*, 2002.

[60] S. Sasy, S. Gorbunov, and C. W. Fletcher. ZeroTrace: Oblivious Memory Primitives from Intel SGX. In *25th Network and Distributed System Security Symposium (NDSS)*, 2018.

[61] M. Schuchard, A. W. Dean, V. Heorhiadi, N. Hopper, and Y. Kim. Balancing the Shadows. In *9th ACM Workshop on Privacy in the Electronic Society (WPES)*, 2010.

[62] M. Schwarz, S. Weiser, D. Gruss, C. Maurice, and S. Mangard. Malware Guard Extension: Using SGX to Conceal Cache Attacks. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2017.

[63] S. Shinde, Z. L. Chua, V. Narayanan, and P. Saxena. Preventing Page Faults from Telling Your Secrets. In *11th ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, 2016.

[64] R. Snader and N. Borisov. Improving Security and Performance in the Tor Network through Tunable Path Selection. *IEEE Transactions on Dependable and Secure Computing*, 2011.

[65] E. Stefanov, E. Shi, and D. Song. Towards Practical Oblivious RAM. In *19th Network and Distributed System Security Symposium (NDSS)*, 2012.

[66] E. Stefanov, M. Van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, and S. Devadas. Path ORAM: An Extremely Simple Oblivious RAM Protocol. In *20th ACM Conference on Computer and Communications Security (CCS)*, 2013.

[67] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. *ACM SIGCOMM Computer Communication Review*, 2001.

[68] P. Tabriz and N. Borisov. Breaking the collusion detection mechanism of MorphMix. In *6th International Workshop on Privacy Enhancing Technologies (PET)*, pages 368–383. Springer, 2006.

[69] The Tor Project. Tor Directory Services v3. https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt. Accessed September 2018.

[70] The Tor Project. Tor Metrics. https://metrics.torproject.org/. Accessed September 2018.

[71] The Tor Project. Who Uses Tor? https://www.torproject.org/about/torusers.html.en. Accessed September 2018.

[72] United Nations. Universal Declaration of Human Rights (Article 12), 1948. http://www.un.org/en/universal-declaration-human-rights/. Accessed September 2018.

[73] C. Wacek, H. Tan, K. S. Bauer, and M. Sherr. An Empirical Evaluation of Relay Selection in Tor. In *20th Network and Distributed System Security Symposium (NDSS)*, 2013.

[74] D. S. Wallach. A Survey of Peer-to-Peer Security Issues. In *2002 Mext-NSF-JSPS International Conference on Software Security: Theories and Systems (ISSS)*. Springer, 2003.

[75] P. Wang and Y. Kim. Myrmic: Secure and Robust DHT Routing, 2006.

[76] Q. Wang, P. Mittal, and N. Borisov. In search of an anonymous and secure lookup: attacks on structured peer-to-peer anonymous communication systems. In *17th ACM Conference on Computer and Communications Security (CCS)*, pages 308–318. ACM, 2010.

[77] X. Wang, H. Chan, and E. Shi. Circuit ORAM: On Tightness of the Goldreich-Ostrovsky Lower Bound. In *22nd ACM Conference on Computer and Communications Security (CCS)*, 2015.

[78] R. Wojtczuk and J. Rutkowska. Attacking Intel Trusted Execution Technology. *Invisible Things Lab*, 2009.

[79] R. Wojtczuk and J. Rutkowska. Attacking SMM Memory via Intel CPU Cache Poisoning. *Invisible Things Lab*, 2009.

[80] M. Wright, M. Adler, B. N. Levine, and C. Shields. Defending Anonymous Communications Against Passive Logging Attacks. In *24th IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2003.

[81] Y. Xu, W. Cui, and M. Peinado. Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems. In *36th IEEE Symposium on Security and Privacy (S&P)*, 2015.

[82] B. Zantout and R. Haraty. I2P Data Communication System. In *10th International Conference on Networks (ICN)*, 2011.