PIR-Tor: Scalable Anonymous Communication Using Private Information Retrieval *

Prateek Mittal[1] *Femi Olumofin*[2] *Carmela Troncoso*[3] *Nikita Borisov*[1] *Ian Goldberg*[2]

[1] *University of Illinois at Urbana-Champaign*
1308 West Main Street
Urbana, IL, USA
{mittal2,nikita}@illinois.edu

[2] *University of Waterloo*
200 University Ave W
Waterloo, ON, Canada
{fgolumof,iang}@cs.uwaterloo.ca

[3] *K.U.Leuven/IBBT*
Kasteelpark Arenberg 10
3001 Leuven Belgium
carmela.troncoso@esat.kuleuven.be

Abstract

Existing anonymous communication systems like Tor do not scale well as they require all users to maintain up-to-date information about all available Tor relays in the system. Current proposals for scaling anonymous communication advocate a peer-to-peer (P2P) approach. While the P2P paradigm scales to millions of nodes, it provides new opportunities to compromise anonymity. In this paper, we step away from the P2P paradigm and advocate a client-server approach to scalable anonymity. We propose PIR-Tor, an architecture for the Tor network in which users obtain information about only a few onion routers using private information retrieval techniques. Obtaining information about only a few onion routers is the key to the scalability of our approach, while the use of private retrieval information techniques helps preserve client anonymity. The security of our architecture depends on the security of PIR schemes which are well understood and relatively easy to analyze, as opposed to peer-to-peer designs that require analyzing extremely complex and dynamic systems. In particular, we demonstrate that reasonable parameters of our architecture provide equivalent security to that of the Tor network. Moreover, our experimental results show that the overhead of PIR-Tor is manageable even when the Tor network scales by two orders of magnitude.

1 Introduction

As more of our daily activities shift online, the issue of user privacy comes to the forefront. Anonymous communication is a privacy enhancing technology that enables a user to communicate with a recipient without revealing her identity (IP address) to the recipient or a third party (for example, Internet routers). Tor [10] is a deployed network for anonymous communication, which consists of about 2 000 relays and currently serves hundreds of thousands of users a day [45]. Tor is widely used by whistleblowers, journalists, businesses, law enforcement and government organizations, and regular citizens concerned about their privacy [46].

Tor requires each user to maintain up-to-date information about all available relays in the network (global view). As the number of relays and clients increases, the cost of maintaining this global view becomes prohibitively expensive. In fact, McLachlan et al. [22] showed that in the near future the Tor network could be spending more bandwidth for maintaining a global view of the system than for anonymous communication itself. Existing approaches to improving Tor's scalability advocate a peer-to-peer approach. While the peer-to-peer paradigm scales to millions of relays, it also provides new opportunities for attack. The complexity of the designs makes it difficult for the authors to provide rigorous proofs of security. The result is that the security community has been very successful at breaking the state-of-art peer-to-peer anonymity designs [4, 6, 7, 23, 47, 48].

In this paper, we step away from the peer-to-peer paradigm and propose PIR-Tor, a scalable client-server approach to anonymous communication. The key observation motivating our architecture is that clients require information about only a few relays (3 in the current Tor network) to build a circuit for anonymous communication. Currently, clients download the entire database of relays to protect their anonymity from compromised directory servers. In our proposal, on the other hand, clients use private information retrieval (PIR) techniques to download information about only a few relays. PIR prevents untrusted directory servers from learning any information about the clients' choices of relays, and thus mitigates route fingerprinting attacks [6, 7].

We consider two architectures for PIR-Tor, based on the use of computational PIR and information-theoretic PIR, and evaluate their performance and security. We find that for the creation of a single circuit, the archi-

*This is an extended version of our USENIX Security 2011 paper [26].

tecture based on computational PIR provides an order of magnitude improvement over a full download of all descriptors, while the information-theoretic architecture provides two orders of magnitude improvement over a full download. However, in the scenario where clients wish to build multiple circuits, several PIR queries must be performed and the communication overhead of the computational PIR architecture quickly approaches that of a full download. In this case, we propose to perform only a few PIR queries and reuse their results for creating multiple circuits, and discuss the security implications of the same. On the other hand, for the information-theoretic architecture, we find that even with multiple circuits, the communication overhead is at least an order of magnitude smaller than a full download. It is therefore feasible for clients to perform a PIR query for each desired circuit. In particular, we show that, subject to certain constraints, this results in security equivalent to the current Tor network. With our improvements, the Tor network can easily sustain a 10-fold increase in both relays and clients. PIR-Tor also enables a scenario where all clients convert to middle-only relays, improving the security and the performance of the Tor network [9].

The remainder of this paper is organized as follows. We discuss related work in Section 2. We present a brief overview of Tor and private information retrieval in Section 3. In Section 4, we give an overview of our system architecture, and present the full protocol in Section 5. We discuss the traffic analysis implications of our architecture in Section 6. Sections 7 and 8 contain our performance evaluation for the computational and information-theoretic PIR proposals respectively. We discuss the ramifications of our design in Section 9, and finally conclude in Section 10.

2 Related Work

In contrast to our client-server approach, prior work mostly advocates a peer-to-peer approach for scalable anonymous communication. We can categorize existing work on peer-to-peer anonymity into architectures that are based on random walks on unstructured or structured topologies, and architectures that use a lookup operation in a distributed hash table.

Besides these peer-to-peer approaches Mittal et al. [25] briefly considered the idea of using PIR queries to scale anonymous communication. However, their description was not complete, and their evaluation was very preliminary. In this paper, we build upon their work and present a complete system architecture based on PIR. In contrast to prior work, we also consider the use of information-theoretic PIR, and show that it outperforms computational PIR based Tor architecture in many scaling scenarios. We also provide an analysis of the implications of clients not having the global system view, and show that reasonable parameters of PIR-Tor provide equivalent security to Tor.

2.1 Distributed hash table based architectures

Distributed hash tables (DHTs), also known as structured peer-to-peer topologies, assign neighbor relationships using a pseudorandom but deterministic mathematical formula based on IP addresses or public keys of nodes.

Salsa [29] is built on top of a DHT, and uses a specially designed secure lookup operation to select random relays in the network. The secure lookups use redundant checks to mitigate attacks that try to bias the result of the lookup. However, Mittal and Borisov [23] showed that Salsa is vulnerable to information leak attacks: as the attackers can observe a large fraction of the lookups in the system, a node's selection of relays is no longer anonymous and this observation can be used to compromise user anonymity [6, 7]. Salsa is also vulnerable to a selective denial-of-service attack, where nodes break circuits that they cannot compromise [4, 47].

Panchenko et al. proposed NISAN [35] in which information-leak attacks are mitigated by a secure iterative lookup operation with built-in anonymity. The secure lookup operation uses redundancy to mitigate active attacks, but hides the identity of the lookup destination from the intermediate nodes by downloading the entire routing table of the intermediate nodes and processing the lookup operation locally. However, Wang et al. [48] were able to drastically reduce the lookup anonymity by taking into account the structure of the topology and the deterministic nature of the paths traversed by the lookup mechanism.

Torsk, introduced by McLachlan et al. [22], uses secret *buddy nodes* to mitigate information leak attacks. Instead of performing a lookup operation themselves, nodes can instruct their secret buddy nodes to perform the lookup on their behalf. Thus, even if the lookup process is not anonymous, the adversary will not be able to link the node with the lookup destination (since the relationship between a node and its buddy is a secret). However, the aforementioned work of Wang et al. [48] also showed some vulnerabilities in the mechanism for obtaining secret buddy nodes.

2.2 Random walk based architectures

In MorphMix [38] the scalability problem in Tor is alleviated by organizing relays in an unstructured peer-to-peer overlay, where each relay has knowledge of only a few other relays in the system. For building circuits, an

initiator performs a random walk by first selecting a random neighbor and building an onion routing circuit to it. The initiator can then query the neighbor for its list of neighbors, select a random peer, and then extend the onion routing circuit to it. This process can be iterated a number of times to build a random walk of any desired length.

MorphMix is vulnerable to a *route capture* attack, where a malicious relay returns a list of only other colluding nodes during a random walk. This attack ensures that once the random walk hits a compromised relay, all subsequent relays in the random walk are also compromised. In particular, when the first relay in the random walk is compromised, user anonymity is trivially broken. While MorphMix proposed a collusion detection mechanism to mitigate the route capture attack, it was later shown that the mechanism can be broken by a colluding set of attackers that models the internal state of each relay [44].

ShadowWalker [24] also uses a random walk to locate relays, but instead of organizing relays into an unstructured overlay, it uses a distributed hash table. Neighbor relationships in the DHT are deterministic, and can be verified by the initiator to mitigate route capture attacks. To prevent any information leakage during verification of neighbor information, some redundancy is incorporated into the topology itself. Recently, Schuchard et al. [39] analyzed an attack on ShadowWalker, and also studied a fix for the attack.

We note that all of the peer-to-peer designs provide only heuristic security, and the security community has been very successful at breaking the state-of-art designs. This is partly because of the complexity of the designs, which make it difficult for the system designers to rigorously analyze the security of the system. We also note that all secure peer-to-peer systems are built on top of assumptions that are difficult to realize in practice. For example, security of these designs depends on the fraction of compromised relays in the system being less than 20–25%. Modern botnets can comprise of tens to hundreds of thousands of bots [19], which is likely sufficient to overwhelm the security of the system. In PIR-Tor, we target a design where it is feasible to rigorously argue about the anonymity properties of the design, and where the ability to obtain random relays both securely and anonymously does not depend on the fraction of compromised relays in the system.

3 Background

3.1 Tor

Tor [10] is a deployed network for low-latency anonymous communication. Tor serves hundreds of thousands of clients, and carries terabytes of traffic per day [45]. The network is comprised of approximately 2 000 relays as of February 2011 [20]. Tor clients first download a complete list of relays (called the *network consensus*) from *directory servers*, and then further download detailed information about each of the relays (called the *relay descriptors*). The network consensus is signed by trusted *directory authorities* to prevent directory servers from manipulating its contents. Clients select three relays to build *circuits* for anonymous communication. A fresh network consensus must be downloaded at least as often as every 3 hours, while fresh relay descriptors are downloaded every 18 hours.

To protect against certain long-term attacks [33] on anonymous communication, each client, when it starts Tor for the first time, selects a set of three *guard relays* from among fast and stable nodes. As long as the selected guards remain available, new ones will not be chosen. The first relay in any circuit constructed by the client will be one of its three guards. Also, clients select the final relay from the subset of the Tor relays which allow traffic to exit to the Internet, called the *exit relays*; each exit relay has an *exit policy*, which lists the ports to which the relay is willing to forward traffic, and the client's choice of exit relay must of course be compatible with its intended use of the circuit. Any relay is eligible to be the middle relay of a circuit. Clients can multiplex multiple TCP connections (called *streams*) over a single Tor circuit; the lifetime of a circuit is generally 10 minutes. Finally, Tor relays have heterogeneous bandwidths, and subject to the above constraints, clients select a Tor relay with a probability that is proportional to a relay's bandwidth.[1]

3.2 PIR

Private information retrieval [5] provides a means of retrieving a block of data out of a database of r blocks, without the database server learning any information about which block was retrieved. A trivial solution to the PIR problem — the one used currently by Tor — is to transfer the entire database from the server to the client, and then retrieve the block of interest from the downloaded database. Although the trivial solution offers perfect privacy protection, the communication overhead is impractical for large databases or for a system like Tor where minimizing bandwidth usage remains a high priority. PIR schemes are therefore designed to provide sublinear communication complexity.

We can classify PIR schemes in terms of their privacy guarantees and the number of servers required for

[1] Since not all relays are eligible for every position, some additional load-balancing logic is used to underweight relays eligible to be guards or exits when choosing middle relays.

the protection they provide. Information-theoretic PIR schemes (ITPIR) are multi-server schemes that guarantee query privacy irrespective of the computational capabilities of the servers answering the user's query. ITPIR schemes assume the database servers are not colluding to determine the user's query. Single-server computational PIR schemes (CPIR), on the other hand, assume a computationally limited database server that is unable to break a hard computational problem, such as the difficulty of factoring large integers. The noncollusion requirement is then removed, at some cost to efficiency.

We choose the single-server lattice-based scheme by Aguilar-Melchor et al. [1] as an example of CPIR, and the multi-server scheme by Goldberg [12] as an example of ITPIR. The CPIR scheme is the best-performing single-server scheme [32], and both are available as open-source libraries.

4 System Overview

4.1 Design goals

1. Scalable architecture: We target a design for anonymous communication that is able to scale the number of relays and clients in the network. We note that a design that is able to accommodate more relays in the network not only improves the network performance, but also improves user anonymity [9].

2. Security: Prior work on scalable anonymous communication only provides heuristic security guarantees, and the security community has been very successful at breaking the state-of-art designs. We target a design that leverages well-understood security mechanisms making it relatively easy to analyze the security of the system. Secondly, we aim to achieve similar security properties as in the existing Tor network. We show that reasonable parameters of PIR-Tor are able to provide equivalent security to the Tor network.

3. Efficient circuit creation: Architectures that impose additional latency during circuit creation may not be practical, since the user needs to wait for the circuit creation to finish before starting anonymous communication.

4. Minimal changes: We target a design that requires minimal changes to the existing Tor architecture. For instance, transitioning Tor to a peer-to-peer system will require a significant engineering effort. Our design leverages existing implementations and requires changes to only the directory functionality and relay selection mechanism in Tor and can be incrementally deployed by both clients and relays.

5. Preserving Tor constraints: The Tor network imposes several constraints on the selection of relays during circuit construction. For example, the first relay must be one of the user's guards, the final relay must allow traffic to exit to a user's desired port, and the relays must be selected in proportion to their bandwidth for load balancing the network. Some prior work like ShadowWalker [24] and Salsa [29] did not focus on these issues.

Limitations: Our architecture achieves its scalability properties by trading off bandwidth for computation; thus directory servers will be required to spend additional computational resources. In our performance evaluation we show that the computational resources required to support our architecture are feasible.

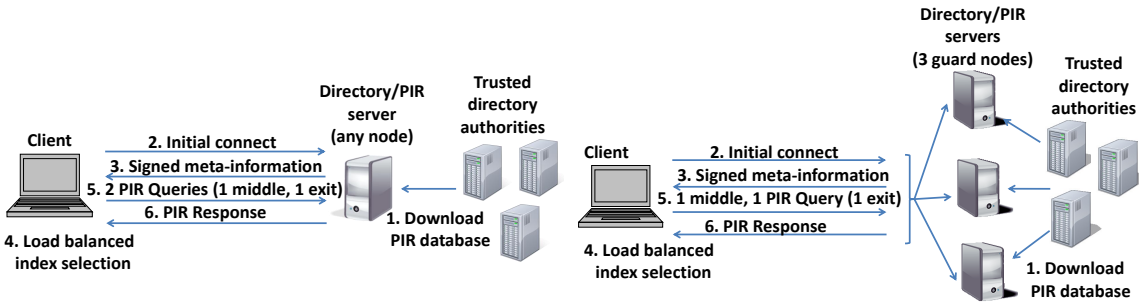
4.2 System architecture

Our key insight when designing PIR-Tor is that the client-server model in Tor can be preserved while simultaneously improving its scalability by having users download the descriptors of only a few relays in the system, as opposed to downloading the global view. However, naively doing so can enable malicious directory servers to launch fingerprinting attacks against the users, thereby compromising anonymity. We propose that users leverage private information retrieval protocols to download the identities of a few relays, thereby protecting their privacy against compromised directory servers. Note that a client does not need to use a PIR protocol to select its guard relays; a full download of the network consensus and relay descriptors suffices, since guard relay selection is a one-time operation that does not affect the scalability of the protocol.

Recall that private information retrieval has two flavors: computational PIR and information-theoretic PIR. While both CPIR and ITPIR can be used by clients, the underlying techniques have different threat models, resulting in slightly different architectures, as depicted in Figure 1.

Computational PIR at directory servers: Computational PIR can guarantee user privacy even when there is a single untrusted database. In this scenario, we propose that as in the current Tor architecture, any relay can act as a directory server. The directory servers maintain a global view of the system, and act as a PIR database. Clients can then use a CPIR protocol to query the directory servers and obtain the identities of random relays in the system.

Information-theoretic PIR at directory authorities (rejected): Information-theoretic PIR can guarantee user privacy only when a threshold number of databases do not collude. Since directory servers in the current Tor network are untrusted, they cannot be used as PIR databases. However, Tor has eight directory *authorities* sign the global system view (the network consensus).



(a) CPIR-based architecture

(b) ITPIR-based architecture

Figure 1: System Architecture: For the CPIR architecture, an arbitrary set of relays are selected as the directory servers (PIR servers) that maintain a current copy of the PIR database, while for the ITPIR architecture, guard relays are the directory servers. Directory servers download the PIR database from trusted directory authorities. To perform a PIR query, clients first obtain meta-information about the PIR database from the directory servers, and then use the meta-information to select the index of the PIR block to query, taking into consideration the bandwidths and the exit policies of the relays. Relay information from the results of the PIR queries can be used to build circuits for anonymous communication. Note that in the ITPIR architecture, clients use PIR to query for only the exit relays.

Since Tor already trusts that the majority of directory authorities are honest, one potential solution could have been to use the directory authorities as PIR databases. However, we reject this approach since the directory authorities would become performance bottlenecks in the system, in addition to targets for DDoS attacks.

Information-theoretic PIR at guard relays: Instead, we note that Tor already places significant trust in guard nodes. If all of a client's guard relays are compromised, then they can perform end-to-end timing analysis [2] in conjunction with selective denial of service attacks [4] to break user anonymity in the *current* Tor network. Thus we consider using a client's three guard nodes as the servers for ITPIR. Unless all three guard nodes are compromised they cannot learn the identities of the relays downloaded by the clients. Even if all three guard relays *are* compromised, they cannot actively manipulate elements in the PIR database since they are signed by the directory authorities; they can only learn which exit relay descriptors were downloaded by the clients. (In Tor, guards always know the identities of the middle nodes in circuits through them.) If the exit relay in a circuit is honest, then guard relays cannot break user anonymity. On the other hand, if the exit relay used is malicious, then user anonymity is broken [6], but in this scenario, the adversary could have performed end-to-end timing analysis anyway [2] (in the current Tor network).

5 PIR-Tor Protocol Details

5.1 Database organization and formatting

We first note that Tor relays are selected based on some constraints. For instance, the first relay must be an entry guard, and the last relay must be an exit relay. We propose to organize the list of relays into three separate databases, corresponding to guard nodes, middle nodes and exit nodes. Note that some relays function as entry guards as well as exit relays — such relays are duplicated in both the guard database and the exit database.

In addition to the last relay being an exit, its exit policy must satisfy the client application requirements. In a February 2011 snapshot of the current Tor network, there were 471 standard exits (default exit policy) and 482 non-standard exits sharing 221 policies. Had the number of non-standard exits been small, then clients in PIR-Tor could download all the relay descriptors for the non-standard exits, and use PIR to select descriptors for the standard exits. However, this is not the case. Instead, we propose that nodes in the exit database be grouped by their exit policies. Furthermore, in order to keep the number of groups manageable, we propose that there be a small set of standard exit policies that exit relays can choose from. Our architecture can accommodate a small set of relays with non-standard exit policies, and these outliers can be downloaded in their entirety as above.

Tor relays have heterogeneous bandwidth capabilities, and relays with higher capacities are selected with a

higher probability in order to load balance the network. Bandwidth-weighted selection is straightforward given a global view of the network. We now outline two strategies to enable clients to perform weighted relay selection without this global view. The first strategy implements the Snader-Borisov [41, 42] criterion for relay selection, where only the relative *rank* of the relays in terms of their bandwidths is used for relay selection.[2] The second strategy is more similar to the current Tor algorithm, where the entire bandwidth *distribution* of relays is taken into consideration for relay selection. In both scenarios, we first sort relays in each of the databases in order of bandwidth. Clients can use the Snader-Borisov mechanism by choosing the relay index to query with probability that depends on the index value. For example, if the relays are sorted in descending order of bandwidth, then clients can select relays having a smaller index with higher probability. To implement an algorithm similar to the current Tor network, we propose that clients download a bandwidth distribution synopsis from the directory servers, and use it to make the relay selection. Finally, we note that the exit database is treated as a special case since relays are first grouped based on their exit policies, and within each group, relays are further sorted by bandwidth. This enables a client to select an exit relay whose exit policy satisfies its application requirements in a load-balanced manner.

The PIR protocols we consider are *block-based*: the database is composed of a number of equal-sized blocks. The block size must be large enough to hold at least a single relay descriptor, but may hold more. We must also ensure that relay descriptors do not cross block boundaries by padding the database. To guard against active attacks by directory servers, each block is signed by the directory authorities; the data signed also includes the block number (index), the consensus timestamp and a database identifier. To minimize overhead, we use the threshold BLS signature scheme [3] since signatures in that scheme are single group elements (22 bytes, for example, for 80-bit security), regardless of the number of directory authorities issuing signatures.

5.2 PIR Protocols and database locations

5.2.1 Computational PIR

Computational PIR protocols can guarantee privacy of user queries even with a single untrusted relay acting as a PIR database. Thus, we can designate an arbitrary set of relays in the network as directory servers, and only the directory servers need to maintain a global view of all the relays, i.e., a current copy of the network consensus formatted as above. Then, instead of downloading the entire consensus document from the directory server, clients connecting to these directory servers use a computational PIR protocol to retrieve a block of their choice, without revealing any information about *which* block, to the directory server. While our architecture is compatible with all existing CPIR protocols, we use the lattice-based scheme proposed by Agular-Melchor and Gaborit [1] since it is the computationally fastest scheme available. Note that the lattice-based CPIR protocol is a single-server protocol, and does not require any interaction with other directory servers.

5.2.2 Information-Theoretic PIR

Information-theoretic PIR protocols guarantee privacy of user queries only if a threshold number of PIR databases do not collude. As stated above, we use a client's three guard relays as ITPIR directory servers. The parameters of the protocol are set such that the guard relays do not learn any information about the client's block unless all three of them collude.

5.3 Client query protocol and meta-information exchange

To query for a middle and exit relay, a client connects to one of its directory (PIR) servers, which responds back with the meta-information about each of the PIR databases, such as the number of blocks in the database, the block size, the distribution of exit policies, and a bandwidth distribution synopsis. Note that the meta-information is also timestamped and signed by the directory authorities. Based on this information, clients can construct a PIR query to select Tor relays while satisfying the constraints of the user. Clients can perform load balancing based on the Snader-Borisov mechanism by selecting an index to query with a probability that depends on the index value. For greater flexibility, clients can perform load balancing in a manner similar to the current Tor architecture by using the bandwidth distribution synopsis to select an index to query. The PIR queries are performed by the clients well in advance of constructing the circuit, so as not to impose extra latency during circuit construction. Note that clients may not be able to predict the exit policies required by circuits in advance. To bypass this constraint, recall that the relays in the exit database are grouped based on a small set of standard exit policies, and clients can perform a few PIR queries to obtain exit relays that satisfy all standard exit policies. Finally, clients can periodically download the relay

[2] The use of the Snader-Borisov criterion may have an impact on the performance of the Tor network. Murdoch and Watson's queueing model [28] suggests that it will cause greater congestion at Tor relays, whereas Snader and Borisov's flow-level simulations [42] predict similar or even improved network utilization.

descriptors of the small set of exit relays that have non-standard exit policies (every 3 hours).

Next, we propose an optimization that clients can perform while using guard relays as directory servers in the case of information-theoretic PIR. We note that during circuit creation, a guard relay learns the identity of the middle relay. Thus the clients could simply skip the PIR for the middle database, and directly query a single guard relay for a particular block. Note that all blocks are signed by the directory authorities, and any active attacks by the guard relay will be detected by the client. Also note that the fetched descriptors should only be used in conjunction with the guard relay from which they were obtained; otherwise, even a single compromised guard would be able to perform fingerprinting attacks [6].

5.4 Circuit Construction

The circuit creation mechanism remains the same as in the current Tor network. In the current Tor network, clients construct a new circuit every 10 minutes. As we show in Section 8, in the ITPIR scenario, the cost of all Tor clients performing one PIR query (since the middle relay is fetched without using PIR) every 10 minutes is manageable. In the CPIR setting, the communication overhead of all Tor clients performing two PIR queries in a 10-minute interval is rather high, and we propose to perform fewer PIR queries, and reuse descriptors in subsequent time intervals. We discuss this further in Section 7.

6 Traffic Analysis Resistance of PIR-Tor

In this section we evaluate the resistance to traffic analysis of PIR-Tor. We consider an adversary that can observe some fraction of the network and has the ability to generate, modify, delete, or delay traffic. She can compromise a fraction of the relays, or introduce relays of her own. Further, we consider that the adversary can observe clients' requests to the PIR-Tor directory servers, and knows that in these requests the client only learns about a fraction of the relays in the network.

As pointed out in the past [6, 7], clients' partial knowledge of the relays belonging to the anonymity network enables *route fingerprinting* attacks. In these papers it is assumed that relay discovery is a non-anonymous process. Hence, an adversary observing the discovery process can build a mapping between users and the relays they know. If clients learn unique (disjoint) sets of relays, their paths can be "fingerprinted", and the client's identity can be trivially recovered from this mapping. This problem does not exist in the current Tor, where querying the directories provides clients with a global view of the network.

In PIR-Tor the threat model slightly differs from the one in [6, 7]. Directory queries continue being identifiable, but PIR prevents the adversary from learning which exact relays were retrieved from the database, avoiding the creation of a mapping describing users' knowledge. Therefore, when route fingerprinting is performed the attack does not result in a direct loss of anonymity. Even if the choice of relays appearing in the fingerprint were unique, the adversary does not have a way to link this fingerprint to a specific client. In fact, the only way for the attacker to link the client with the destination of her traffic is to control the first and last relays in the path and perform a traffic confirmation attack [37], which in our system will happen with probability c^2, where c is the fraction of compromised bandwidth in the network — the same probability as in the current Tor network.

Although route fingerprinting does not result in a direct loss of anonymity in PIR-Tor, the information leaked could be used by the adversary to relate connections from the same user and construct *behavioral profiles*. In turn, these profiles can lead to the re-identification of users directly [16] or by combining them with publicly available databases [14, 30, 43]. We note that the linkability of circuits is not a problem unique to PIR-Tor, and that features other than partitioning the network (e.g., cookies [36], session timing [17], or frequently accessed hosts [17]) can be used in the current Tor network to profile users.

6.1 Impact of fingerprinting on PIR-Tor

Before diving into the analysis we note that the number of relays (or descriptors) in each PIR block is irrelevant for the result. Fingerprinting attacks are based on the clients' knowledge of relays in the network, but in PIR-Tor clients retrieve blocks that may contain one or more descriptors. Hence, either the client knows about all the descriptors in a block or she does not know any of them. Thus, from the point of view of the adversary all relays in a block are equivalent, regardless of how many descriptors are in this block; only the *number of blocks* matters when computing the probabilities we use in our analysis.

We consider an adversary that controls the receiver of the communication, and thus can observe the exit relay chosen by the client. Additionally, she may also control the exit relay hence also learning the middle relay in the client's circuit.

6.1.1 One PIR request per circuit construction

If the computation and communication cost for clients and directory servers in dealing with PIR queries is small (as when ITPIR is used), clients could request new descriptors for each circuit construction. Regardless of

the selection algorithm used, due to the PIR properties, the adversary cannot distinguish which block is retrieved from the database with each query and hence she gains no information as to which relays are known to the client. In this setting the adversary must assume that all relays are known to the client, and PIR-Tor fingerprinting resistance is equivalent to that of the current Tor network.

Nevertheless, when CPIR is used we must expect limitations both in bandwidth and computation capabilities. Therefore, each time the client obtains a set of descriptors with a CPIR query, these descriptors may have to be reused across multiple circuit rebuilds. In the next section we evaluate the impact of this reuse on the privacy protection offered by PIR-Tor.

6.1.2 Reusing descriptors for circuit construction

In our analysis we assume that the attacker observes the exit relay (respectively exit and middle relays) of a client's circuit. As we have already discussed, this does not directly leak information about the client's identity and anonymity is preserved. However, the adversary can still profile clients based on their network knowledge, eventually leading to de-anonymization [14, 16, 30, 43].

The adversary can construct a behavioral profile with all connections she observes coming from exit relays (respectively exit and middle relays) that belong to the same PIR block. If the selection algorithm is such that many clients have knowledge of a block (recall that all relays in the block are equivalent for the attacker) the profile recovered by the adversary is an aggregate profile of all these users, jeopardizing the de-anonymization of individual clients. On the other hand, if the choice of relays is unique to each client the profile recovered by the adversary accurately reflects the behavior of an individual user and the danger of de-anonymization grows. Therefore, it is desirable that clients share choices such that the adversary can only obtain aggregated profiles that reduce her precision when re-identifying clients. Other ways than relay selection for the attacker to link and/or discriminate clients' connections [17, 36] are left out of the scope of our analysis.

In this section, we evaluate the protection against profiling provided by PIR-Tor when descriptors have to be reused across circuit constructions. We aim to answer the question "how precisely can the adversary assign an observed connection (exit relay, or exit and middle) to a unique client?". We use as a metric the fraction α of clients that could be initiators of a connection (i.e., the expected fraction of clients that have knowledge of the PIR-Tor block containing the relay(s) observed by the adversary). The larger the fraction of clients that may know the observed relay, the better privacy users enjoy because the adversary can only construct aggregate profiles. We note that even if the adversary is actually collecting information from a single user, she cannot be sure that this is the case based on the PIR-Tor relay selection algorithm; she must assume that the profile she observes may contain sessions from multiple users. We also note that, based on the relay selection algorithm, the adversary cannot link connections from a user routed through different exit relays. This is because the PIR properties prevent the attacker from learning any relation between the descriptors retrieved by a client. Hence, the connections of one client routed through exit relays in different PIR blocks are unlinkable and the adversary must assign them to different profiles (that may or may not contain information about other users).

If the adversary observes connections coming from the exit relay e, the fraction of clients α that may know this relay are those who retrieved from the database the block containing e. In PIR-Tor we assume that clients retrieve a set B of b blocks every time they query the directory server, hence the fraction of clients that have knowledge of the block containing e is: $\alpha = (1 - (1 - \Pr[e])^b)$, where $\Pr[e]$ is the probability of choosing the block containing e as one of the b retrieved blocks, and depends on the algorithm used for the selection of relays. For simplicity in our analysis we assume that there is only a single standard exit policy.

We explained in Section 5 that for load balancing, relays with higher capacities are selected with a higher probability. We described two criteria for selecting relays: a bandwidth-based criterion (BW), and the Snader-Borisov criterion (SB). To evaluate the BW criterion according to a realistic bandwidth distribution we captured a snapshot of the Tor consensus directory on 9 February 2011. This directory includes 649 exit relay descriptors after removing the slowest one-eighth of the total relays that are not used to relay traffic at all in the current Tor network [31]. For the evaluation of SB we computed the probability $\Pr[e]$ according to the algorithm introduced in [41]. Given the function $f_s(x) = \frac{(1-2^{sx})}{(1-2^s)}$ a value x is drawn uniformly at random from $[0, 1)$, and the block with index $\lfloor N_{blocks} \times f_s(x) \rfloor$ is selected. The inverse of the function $f_s(x)$ is the function $f_s^{-1}(x) = (\log_2(1 - (1 - 2^s) \cdot x))/s$. Then, the probability of selecting a block containing the relay e in the i-th position of the list is $\Pr[e] = f_s^{-1}(i/N_{blocks}) - f_s^{-1}((i-1)/N_{blocks})$. We use $s = 1$, which results in a probability distribution near to uniform, and $s = 10$, which results in a distribution very skewed towards the relays offering high bandwidth.

Figure 2 shows box plots[3] describing the distribution

[3]The line in the middle of the box represents the median of the distribution of α. The lower and upper limits of the box correspond, respectively, to the first (Q1) and third quartiles (Q3) of the distribution. We also show the outliers: relays e which are chosen with values that are "far" from the rest of the distribution ($\alpha > Q_3 + 1.5(Q3 - Q1)$

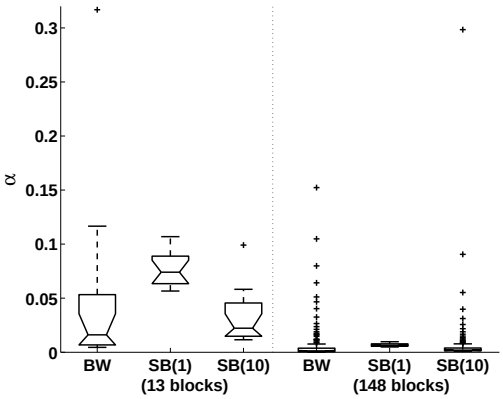


Figure 2: BW and SB(s) selection: evolution of α with the database size.

of α for the different selection algorithms. We choose two database sizes to show the performance of the algorithms when the network scales. A small database that contains 649 exit relays (as in the current Tor network) divided in 13 blocks for optimal performance of the CPIR algorithm (note that if ITPIR is used there is no need to reuse relays across circuit rebuilds).[4] The second database contains 1M relays, divided in 148 blocks. In the BW case, we construct the distribution of bandwidth amongst the relays by concatenating copies of the original list downloaded from the Tor network. We refer the reader to Appendix A for a more detailed analysis of the evolution of α when the network scales.

The median of the BW distribution is $\alpha = 0.016$; that is, 1600 clients have knowledge of each relay when the network is used by $100\,000$ users.[5] When the network grows, the median of α diminishes to 0.0012. As there are more blocks in the database clients have more choice, and so they share knowledge of fewer relays.

We can see that SB(1) offers the best protection (a larger fraction of clients know a relay), but as clients' choice of relays, and hence blocks, is nearly uniform it does not load balance the network. The means of SB(1) and SB(10) are similar; however, SB(10) has a greater variance. SB(10) yields medians of $\alpha = 0.022$ and $\alpha = 0.0019$ when there are 13 and 148 blocks in the database, respectively. In the latter case, the adversary still captures aggregated profiles of 190 clients for the median relay.

If besides the receiver of the communication the adversary also controls the exit relay, then she can observe the middle and exit relays of the client's path. Let us call the observed exit relay e, and the observed middle relay m. The fraction of clients knowing the blocks containing these relays is: $\Pr[e, m \in B] = (1 - (1 - \Pr[e])^b) \cdot (1-(1-\Pr[m])^b)$, where $\Pr[e]$ and $\Pr[m]$ depend on the path selection algorithm. Hence, $\Pr[e, m \in B]$ is orders of magnitude smaller than $\Pr[e \in B]$ increasing the accuracy of profiling, as it becomes less likely that clients share knowledge of both exit and middle relays.

We note that the results above represent the case in which clients only retrieve $b = 1$ blocks per PIR query. If the clients retrieve more blocks they can significantly improve their privacy protection (α grows approximately linearly with b). Moreover, if clients retrieve $b > 1$ blocks each time, they divide by b the number of circuits routed by each of the known exit relays. Finally, we would like to stress that client's profiles are only linkable until they refresh their network knowledge. If, as in the current Tor network, this happens each 3 hours and circuits are rebuilt every 10 minutes, the adversary can link data from only $18/b$ circuits. We have shown in this section that, even though it does not break anonymity, reusing descriptors breaks the unlinkability of circuits. In order to prevent the attack we have discussed, clients should request new blocks from the directory server (or from the guard nodes if ITPIR is used) often or in groups of several blocks such that the reuse of descriptors is minimized.

7 Performance Evaluation of Computational PIR

We now present experimental results for the CPIR architecture. We chose standard security parameters for the CPIR scheme [1] ($\ell_0 = 19$ and $N = 50$), and computed the client/server computation times and communication costs by running an implementation of this scheme [15]. The hardware was a dual Intel Xeon E5420 2.50 GHz quad-core machine running Ubuntu Linux 10.04.1. Note that for our evaluation, we used only a single core, which is equivalent to a standard desktop machine today.

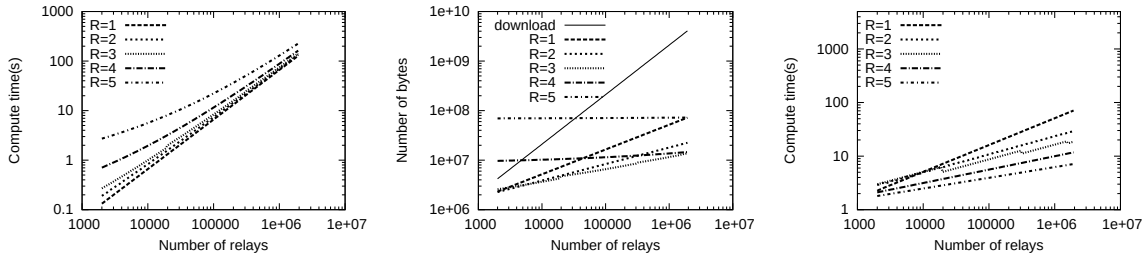
We set the descriptor size to be $2\,100$ bytes (the maximum descriptor size measured from the current Tor network), and set the exit database to be half the size of the middle database [45]. We varied the number of relays in a PIR database, and computed a) PIR server computation, b) total communication, and c) client computation.

Data transfer for CPIR schemes can be reduced using the recursive construction by Kushilevitz and Ostro-

or $\alpha < Q1 - 1.5(Q3 - Q1)$).
We have cut the figure's y axis for better visibility. The figure does not show two outliers for the 13-block BW and SB(10) plots that have $\alpha = 0.38$ and $\alpha = 0.63$, respectively.

[4] For a database of r 2100-byte descriptors and recursion parameter (see Section 7) $R = 2$, the optimal number of blocks is approximately $1.50 \cdot \sqrt[3]{r}$.

[5] The Tor project reports an estimate of Tor users between $100\,000$ and $250\,000$ in January 2011 (`http://metrics.torproject.org/users.html`). We take $100\,000$ to represent the worst-case scenario for the clients.



(a) Server computation (b) Total communication (c) Client computation

Figure 3: CPIR cost. R denotes the recursion parameter in CPIR.

vsky [21] without much increase in computational cost; this recursion can be implemented in a single round of interaction between the client and the server. We denote the recursion parameter in CPIR using R. If we denote the number of relays in the database by n, then the communication cost of CPIR in our architecture is proportional to $8^R \cdot n^{1/(R+1)}$.

Figure 3 depicts the server computation, communication, and client computation as a function of the number of Tor relays for varying values of the recursion parameter R. Increasing R reduces communication (and client computation) drastically while having only a small impact on server computation. Note that for beyond $R = 3$, communication increases again, because the term 8^R in the communication overhead becomes dominant. We can see that when the number of relays is less than $20\,000$, the server computational overhead using $R = 2$ is smaller than $R = 3$, while the communication overhead using $R = 2$ and $R = 3$ is about the same. Beyond $20\,000$ relays, using $R = 3$ results in significant communication savings as compared to $R = 2$, while the server computational overhead is about the same for both parameters. For the remainder of this discussion, we use $R = 2$. We can see that as the network size scales, the communication overhead of CPIR is an order of magnitude smaller than trivial download of the database. Interestingly, even at the current network size, the communication overhead of CPIR is smaller than a trivial download.

Now we discuss the issue of creating multiple circuits within a 3-hour interval (after which the directory databases are refreshed and clients request new descriptors). In this scenario, the trivial download has the advantage that any number of circuits can be created. Tor clients rebuild a circuit after every 10 minutes, so they could create 18 circuits every 3 hours with the communication overhead of a single trivial download. On the other hand, the PIR-based architecture would require 18 PIR queries for middle nodes and another 18 for exit nodes. We can see that unless the number of relays in the database is greater than $40\,000$, trivial download is going to be more efficient than performing multiple CPIR queries. Instead, we propose to perform $b < 18$ queries for both middle and exit nodes, and reuse existing blocks for more circuits. As we discuss in the security analysis, reusing blocks does not affect the anonymity of a single circuit, but may break the unlinkability of multiple circuits.

We now study some particular scaling scenarios in more detail. For each of the following scenarios, we will compute the number of cores required to support the clients. Figure 4 depicts the required number of CPU cores as a function of relays and clients. We also study the communication overhead of CPIR-Tor, along with a comparative analysis with the current Tor protocol. For this analysis, we set the number of blocks $b = 1$. Note that both computation and communication overhead for CPIR-Tor scale linearly with the desired number of blocks. Our results are summarized in Table 1.

Scenario 1: Current Tor Size. Total number of simultaneous relays is 2 000. Total number of simultaneous clients is 250 000. For $2\,000$ relays, server compute time is 0.2 second. The number of exit nodes is around $1\,000$, and the corresponding server compute time is 0.1 seconds. Thus to download a block from both the middle and the exit databases, the total server compute time is 0.3 seconds. Note that we are proposing to download a block every 3 hours. A single directory server would thus be able to support $36\,000$ clients $\left(\frac{3 \cdot 60 \cdot 60}{0.3}\right)$. The total number of cores required to support $250\,000$ clients is only 7. As of February 2011, the size of the Tor network consensus is 560 KB, while the total size of the relay descriptors is about 3.3 MB. Thus the communication overhead per client in the current Tor network is about 1.1 MB every 3 hours (560 KB consensus and $\frac{3300}{6}$ KB relay descriptors), while the corresponding overhead in our architecture is 2 MB. Thus, CPIR-Tor is not suited for the current Tor network size.

Scenario 2: Increasing clients. Total number of relays is fixed at 2 000. Total number of clients increases by a factor of s_c. The number of cores required to sup-

Table 1: Summary of results: Comparison of overhead in Tor, CPIR and ITPIR. The communication overhead is measured per client over a 3 hour interval.

Scenario	Relays	Clients	Tor (MB)	CPIR (MB / Cores)	ITPIR (MB / % Core Utilization per guard)
1	2000	250 000	1.1	2 / 7	0.2 / 0.425%
2	2000	2 500 000	1.1	2 / 70	0.2 / 4.25%
3	20 000	250 000	11	4 / 59	0.5 / 0.425%
4	20 000	2 500 000	11	4 / 553	0.5 / 4.25%
5	250 000	250 000	111	8 / 466	0.2 / 0.425%

port $s_c \cdot 250\,000$ clients is $s_c \cdot 7$ (linear increase). Thus if the number of clients increases to 2.5 million, about 70 cores will be required to support the architecture. Both the number of cores and the communication overhead of the system increases linearly with the number of clients.

Scenario 3: Increasing relays. Total number of relays increases by a factor of s_r. Total number of clients is fixed. The number of cores required to support $s_r \cdot 2\,000$ relays increases sublinearly with s_r. For example, when the number of relays increases from 2 000 to 20 000, the required number of cores increases from 7 to 59. Note that in this scenario, the communication overhead for CPIR-Tor also scales sublinearly, while that of current Tor scales linearly. Thus, as the number of relays increases, it becomes more and more advantageous to use CPIR-Tor. For instance, when the number of relays is 20 000, the communication overhead of Tor is 11 MB every 3 hours, while that of CPIR is only 4 MB.

Scenario 4: Increasing both clients and relays. Total number of relays and clients increases by a factor of s. The number of cores required to support $s \cdot 2\,000$ relays and $s \cdot 250\,000$ clients is strictly less than $7 \cdot s^2$. In order to support 20 000 relays and 2.5 million clients, 553 cores would be required. We note approximately 50% of the Tor relays are already directory servers, so 553 cores in this scenario is feasible. Again, as the number of relays increases, the advantage of CPIR-Tor over Tor becomes larger.

Scenario 5: Converting clients to middle-only relays. Observe that if all 250 000 clients converted to middle-only relays, then the server compute time for the middle database is 20 seconds, while that for the exit database is still 0.1 seconds. Thus, the total number of cores required to support this scenario is approximately 466. (This scenario is not shown in Figure 4.) As compared to the current Tor network, CPIR reduces the communication overhead in the network from 111 MB per client every 3 hours to only 8 MB.

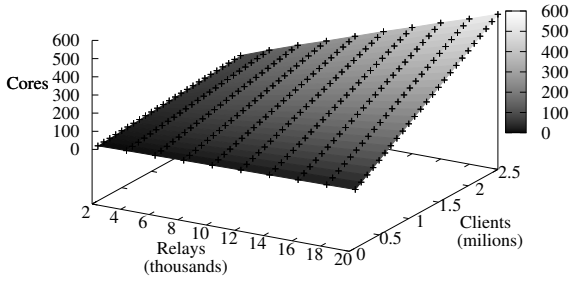


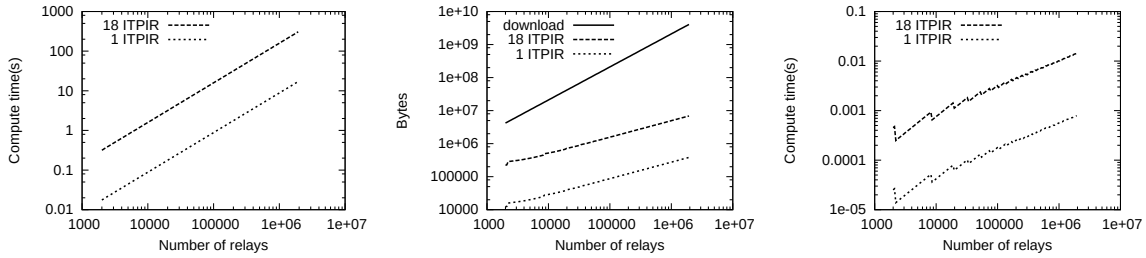
Figure 4: Number of cores as a function of the number of relays and clients (assuming half of the relays are exits).

8 Performance Evaluation of Information-Theoretic PIR

We use an implementation [13] of the multi-server PIR scheme by Goldberg [12] and compute the server computation, total communication, and client computation, for varying values of the number of relays, using a descriptor size of 2 100 bytes, and 3 servers.

Figure 5 plots server computation, total communication, and client computation as a function of the number of Tor relays, using 3 PIR servers (the entry guards). We note that the communication cost for a single ITPIR request is at least 2 orders of magnitude smaller than the cost for a trivial download for all possible scaling scenarios.

Even if we compare the ITPIR-Tor protocol with the Tor protocol over a period of 3 hours, where clients set up 18 circuits, still the communication overhead of ITPIR is an order of magnitude smaller than a full download for all scaling scenarios. Thus in this architecture, we do not need to reuse blocks, providing security equivalent to that of Tor, if at least a single guard relay is honest. Recall that if all guard relays are compromised, then the adversary can break user anonymity in both the current



(a) Server computation (b) Total communication (c) Client computation

Figure 5: 3-server ITPIR cost.

Tor network as well as in PIR-Tor, by selectively denying service [4] to circuits that have an honest exit relay (or destination server) and performing end-to-end timing analysis [2] when the exit relay (or destination server) is compromised.

We now explore various scaling scenarios for Tor, and compute the number of clients that each guard relay can support, along with a comparison of the communication cost to that of Tor. Our results are summarized in Table 1.

Scenario 1: Current Tor Size. Total number of relays is 2 000. Total number of clients is 250 000. For $2\,000$ relays, the number of exit nodes is around $1\,000$, and the corresponding server compute time is 0.005 seconds. Thus to support a single circuit, the total server compute time is 0.005 seconds (for all three guards combined). Note that each client builds a circuit every 10 minutes. A single guard relay would thus be able to support $360\,000$ clients. In the current Tor network, there are $250\,000$ clients, and approximately 500 guard relays, so each guard relay needs to service only 1500 clients on average, and would utilize only 0.425% of one core. The communication overhead of Tor is 1.1 MB per client every 3 hours. In ITPIR, the cost to build a single circuit is only 12 KB. Even if clients build 18 circuits over a three hour interval, the total communication cost of all 18 circuits is 216 KB. Thus ITPIR is useful even with the size of the current Tor network.

Scenario 2: Increasing clients. Total number of relays is fixed at 2 000. Total number of clients increases by a factor of s_c. In order to support $s_c \cdot 250\,000$ clients, guard relays would need to utilize $s_c \cdot 0.425\%$ of a core. Thus even when the number of clients increases to 2.5 million, but the number of guard relays stays fixed at 500, then each guard relay only utilizes a 4.25% fraction of a core. The total communication overhead in the system increases linearly with the number of clients, similar to the current Tor network.

Scenario 3: Increasing relays. Total number of relays increases by a factor of s_r. Total number of clients is fixed. In order to support $s_r \cdot 2\,000$ relays, guard relays would need to utilize only 0.425% of a core. This is because the size increase in the PIR database is offset by the increase in the number of guard relays. Thus, regardless of the number of relays in the system, each guard relay utilizes only 0.425% of a core. Also, as the number of relays increases, the advantage of ITPIR over a full download in terms of communication cost also increases. For instance, at $2\,000$ relays ITPIR is a factor of 5 more efficient than Tor, while at $20\,000$ relays, ITPIR is a factor of 22 more efficient than Tor (516 KB per client every 3 hours as compared to 11.1 MB in Tor).

Scenario 4: Increasing both clients and relays. Total number of relays and clients increases by a factor of s. In order to support $s \cdot 250\,000$ clients, and $s \cdot 2\,000$ relays, each guard relay would need to utilize $s \cdot 0.425\%$ of a core. Thus when the number of clients is 2.5 million, and the number of relays is $20\,000$, each guard relay utilizes 4.25% of a core. Even at 100 times the current client base (25 million), 42% of one core is required, which may be reasonable in multi-core settings. As the number of clients increases, the communication overhead in both ITPIR and Tor increases linearly, while as the number of relays increases, it becomes a lot more advantageous to use ITPIR as compared to Tor.

Scenario 5: Converting clients to middle-only relays. Observe that if all $250\,000$ clients converted to middle-only relays, then the server compute time for the guard relays remains unchanged, since PIR is not performed over the middle database. Thus each guard relay would still utilize only 0.425% of a core.

To further highlight the scalability of ITPIR, we also consider a scenario where all $250\,000$ clients convert to relays, with a similar distribution of guard/middle/exit relays as in the current Tor network. The communication overhead of ITPIR in this scenario is 1.7 MB per client every 3 hours, while that of Tor is 137 MB — two orders of magnitude higher.

9 Discussion

We now discuss some issues in, and ramifications of, our design.

Comparison of CPIR vs. ITPIR. The CPIR-Tor architecture does not require all guard relays to be directory servers, and is more easily integrated into the current Tor network, where a random subset of the relays are directory servers. Moreover, it is ideal for the scenarios where either a client's browsing time is small (possibly estimated using the client's past Tor browsing history), or the client is not interested in the unlinkability of its connections. On the other hand, the ITPIR-Tor architecture requires all guard relays to be directory servers, thus requiring them to maintain a global view of the system, but results in significant communication savings for the clients. The ITPIR-Tor architecture can support a variety of client workloads, while providing a high level of security. In particular, ITPIR-Tor can enable a very attractive scenario where all clients become middle-only relays, without any additional cost to the network, since the middle relays are fetched for free (without doing PIR) by the clients.

Robustness. Recall that each block of the descriptor database is digitally signed by the trusted directory authorities. These signatures prevent malicious PIR servers from tricking clients into accepting false information. However, such malicious servers could still deny service to clients by returning garbage, or by not returning a response at all. As we discuss next, in both CPIR-Tor and ITPIR-Tor clients can easily detect this attack and can stop using those malicious servers.

In CPIR-Tor, a malicious directory server could modify its own copy of the descriptor database in order to corrupt blocks containing, for example, many honest nodes, and leave with correct signatures those blocks containing collaborating malicious nodes. Clients retrieving these "malicious blocks" will be successful, but clients retrieving "honest blocks" will not. In order to defend against this, a CPIR-Tor client that receives even one corrupted block (out of b requests) from a given (Byzantine) directory server should discard the entire response, and make a *new, freshly randomly chosen* query for all b blocks from a different server. It should also avoid using that Byzantine server in the future.

In ITPIR-Tor, on the other hand, such a selective-corruption attack is not possible unless all three guard nodes are colluding. In the ITPIR-Tor setting, Byzantine guard nodes can corrupt the result of the query, but not in a way that depends on which block was requested. Unfortunately, with ITPIR-Tor as presented, although the client will detect the corruption, it will not learn *which* of the guard nodes was Byzantine. This can be rectified, however, using the Byzantine robustness techniques of the underlying ITPIR protocol [12]. In particular, a client receiving blocks with correct signatures may safely use those blocks. If there are corrupted blocks, the client can identify which guard node(s) were Byzantine, and causing the corruption, by extending the queries for just the corrupted blocks to additional guard nodes. When three honest guard nodes are reached, even though the client does not know *a priori* which are the honest ones, the Byzantine nodes will be identified. However, this may come at the cost of the Byzantine nodes (if there are at least three) learning which exit block the client was interested in. Therefore, the client should not use the resulting information to build circuits; it should only use it to learn which nodes were Byzantine and thus should be avoided in the future.

Additional scaling strategies. The Tor Project has been actively working on improving its scaling properties. We now discuss some strategies under consideration that may be implemented in the future. The first strategy is to download relay descriptors on demand [34] during the circuit construction process, as opposed to periodically fetching them in advance. Fetching descriptors on demand would significantly reduce the communication overhead in Tor. However, note that fetching descriptors on demand does not satisfy our goal of efficient circuit creation, since descriptor downloads increase circuit creation times.

The second strategy introduces the idea of microdescriptors [8], which contain all relay descriptor fields that rarely change. All frequently changing fields are placed in the network consensus. Clients download the network consensus document frequently, but the microdescriptors are cached on a long-term basis. We note that this proposal is orthogonal to our architecture, and can be incorporated in the PIR-Tor protocol. In this case, the PIR database would consist of only the network consensus information. The size reduction in the PIR database because of the removal of microdescriptors would translate into both computational and communication savings in our architecture.

Computational puzzles to prevent DoS. In our architecture, directory servers act as PIR databases and perform computation to respond to user queries. This provides an opportunity to the attacker to launch a denial of service (DoS) attack against the directory servers by issuing multiple PIR queries. We propose to use computational puzzles to mitigate the impact of this attack. When a directory server begins to get computationally congested, it starts to issue computational puzzles to

clients. Clients solve the computational puzzle and return the solution to the directory server. The directory server verifies the puzzle solution, and only then starts to spend computational resources to process the client's PIR query.

Impact of churn. In the current Tor network, as the churn in the network increases, clients will have to download the full list of network consensus and relay descriptors more frequently. On the other hand, the impact of churn on PIR-Tor is minimal, since only a small number of directory servers or guards will need to download the global view more frequently. In fact, as long as the rate of database updates is longer than 10 minutes (it is currently set to 3 hours), we can expect the number of client PIR queries to be the same.

Impact of number of circuits. The communication overhead of PIR-Tor is directly proportional to the number of circuit constructions, since for optimal security, clients need to perform 1 or 2 PIR queries per circuit. Tor developers are already working on a proposal to have a separate circuit for each application, to prevent certain kinds of profiling [18]. In this scenario, since there is a separate circuit per application, the timeout period for each circuit can be increased from the current value of 10 minutes, to keep the impact of additional circuits on our architecture minimal (since the timeout period is set to 10 minutes in order to prevent those same profiling attacks).

Incorporating future path constraints. There have been several proposals that incorporate more constraints in the Tor path selection protocol. For example, it has been suggested that relays must be chosen to minimize the chance of an end-to-end timing analysis attack [11, 27]. Also, Sherr et al. [40] proposed to enable applications to choose relays based on different performance constraints like node-based selection, link-based selection, and end-to-end path-based selection. We note that PIR-Tor is able to incorporate these ideas to the extent that each block fetched from the database contains multiple descriptors, and clients could apply similar algorithms to select the descriptor that best fits their constraints.

Preserving option to download global view. We note that many use cases may require a global view of the system. For example, it may be helpful to researchers or developers working on improving the security and performance of the Tor network to have a global view of the system. Thus we propose that directory servers also support an option to download the full database.

Limitations. The Tor network is comprised of volunteer nodes that contribute their bandwidth for anonymous communication. Our proposal essentially trades off bandwidth for computation at the directory servers, and thus directory servers are required to volunteer some extra computational resources. We show in our performance evaluation that only a small fraction of CPU resources need to be volunteered by the designated directory servers, especially in the case of ITPIR-Tor. We believe that PIR-Tor offers a good tradeoff between bandwidth and computational resources, and results in an overall reduction in resource consumption at volunteer nodes. Secondly, our design is not as scalable as alternate peer-to-peer approaches, which can scale to tens of million relays. However, our design provides improved security properties over prior work. In particular, reasonable parameters of PIR-Tor provide equivalent security to that of the Tor network. The security of our architecture mostly depends on the security of PIR schemes which are well understood and relatively easy to analyze, as opposed to peer-to-peer designs that require analyzing extremely complex and dynamic systems. The only exception to this is the scenario of CPIR-Tor with descriptor re-use, where the security analysis is more complex. Moreover, for all scaling scenarios, the communication overhead in our architecture is at least an order of magnitude smaller than that of Tor. Finally, PIR-Tor assumes the use of a small set of standard exit policies for nodes to select from, though a few outliers can be tolerated by downloading their information in their entirety.

10 Conclusion

In this paper, we presented PIR-Tor, an architecture for the Tor network where clients do not need to maintain a global view of the system, and instead leverage private information retrieval techniques to protect their privacy from compromised directory servers. In our evaluation, we find that PIR-Tor reduces the communication overhead of the Tor network by at least an order of magnitude. We analyzed two flavors of our architecture, based on computational PIR and information-theoretic PIR respectively. In computational PIR, clients fetch only a few blocks from the PIR database, and reuse blocks to build additional circuits. While this modification has no impact on client anonymity, it slightly weakens the unlinkability of circuits. On the other hand, in information-theoretic PIR, clients perform a PIR query per circuit creation and do not reuse blocks, resulting in a level of security that is equivalent to the Tor network. While information-theoretic PIR requires all guard relays to be directory servers, computational PIR is more easily integrated into the current Tor network.

Acknowledgments

We are grateful to Roger Dingledine for helpful discussions about Tor. This work also benefited from the feedback of HotSec 2010 attendees, particularly Micah Sherr. We would also like to thank the anonymous reviewers for their comments on earlier drafts of this paper. Femi Olumofin and Ian Goldberg were supported in part by NSERC, MITACS, and The Tor Project. Carmela Troncoso is a research assistant of the Fund for Scientific Research in Flanders (FWO), and was supported in part by the the IAP Programme P6/26 BCRYPT of the Belgian State. Prateek Mittal and Nikita Borisov were supported in part by an HP Labs IRP grant and an NSF grant CNS 09–53655.

References

[1] C. Aguilar-Melchor and P. Gaborit. A lattice-based computationally-efficient private information retrieval protocol, 2007. Presented at WEWORC 2007. http://eprint.iacr.org/2007/446.pdf, Accessed February 2011.

[2] K. S. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. C. Sicker. Low-resource routing attacks against Tor. In Ting Yu, editor, *ACM Workshop on Privacy in the Electronic Society (WPES 2007)*, pages 11–20. ACM, 2007.

[3] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *7th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2001)*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001.

[4] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz. Denial of service or denial of security? In S. De Capitani di Vimercati and P. F. Syverson, editors, *ACM Conference on Computer and Communications Security (CCS 2007)*, pages 92–102. ACM, 2007.

[5] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *IEEE Annual Symposium on Foundations of Computer Science (FOCS 95)*, pages 41–50, 1995.

[6] G. Danezis and R. Clayton. Route fingerprinting in anonymous communications. In A. Montresor, A. Wierzbicki, and N. Shahmehri, editors, *International Conference on Peer-to-Peer Computing (P2P 2006)*, pages 69–72. IEEE Computer Society, 2006.

[7] G. Danezis and P. F. Syverson. Bridging and fingerprinting: Epistemic attacks on route selection. In N. Borisov and I. Goldberg, editors, *8th Privacy Enhancing Technologies Symposium (PETS 2008)*, volume 5134 of *Lecture Notes in Computer Science*, pages 151–166. Springer, 2008.

[8] R. Dingledine. Clients download consensus + microdescriptors. https://gitweb.torproject.org/tor.git/blob/master:/doc/spec/proposals/158-microdescriptors.txt, January 2009. Accessed February 2011.

[9] R. Dingledine and N. Mathewson. Anonymity loves company: Usability and the network effect. In *5th Workshop on the Economics of Information Security (WEIS 2006)*, 2006.

[10] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *13th USENIX Security Symposium*, pages 303–320. USENIX, 2004.

[11] M. Edman and P. Syverson. AS-awareness in Tor path selection. In S. Jha and A. D. Keromytis, editors, *ACM Conference on Computer and Communications Security (CCS 2009)*, pages 380–389. ACM, 2009.

[12] I. Goldberg. Improving the robustness of private information retrieval. In *IEEE Symposium on Security and Privacy (S&P 2007)*, pages 131–148. IEEE Computer Society, 2007.

[13] I. Goldberg. Percy++, 2010. https://sourceforge.net/projects/percy/.

[14] P. Golle and K. Partridge. On the anonymity of home/work location pairs. In H. Tokuda, M. Beigl, A. Friday, A. J. Bernheim Brush, and Y. Tobe, editors, *7th International Conference Pervasive Computing (Pervasive 2009)*, volume 5538 of *Lecture Notes in Computer Science*, pages 390–397. Springer, 2009.

[15] GPGPU Team. High-speed PIR computation on GPU on Assembla. http://www.assembla.com/spaces/pir_gpgpu/.

[16] S. Hansen. AOL removes search data on vast group of web users, October 2006. New York Times.

[17] D. Herrmann and lexi. Contemporary profiling of web users, December 2010. Presented at the 27th Chaos Communication Congress in Berlin on December 27, 2010.

[18] R. Hogan, J. Appelbaum, D. McCoy, and N. Mathewson. Separate streams across circuits by connection metadata. https://gitweb.torproject.org/tor.git/blob/master:/doc/spec/proposals/171-separate-streams.txt, December 2010. Accessed February 2011.

[19] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. Freiling. Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm. In F. Monrose, editor, *1st USENIX Workshop on Large-scale Exploits and Emergent Threats (LEET 2008)*. USENIX Association, 2008.

[20] J. B. Kowalski. Tor network status. http://torstatus.blutmagie.de/. Accessed February 2011.

[21] E. Kushilevitz and R. Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *IEEE Annual Symposium on Foundations of Computer Science (FOCS 97)*, pages 364–373, 1997.

[22] J. McLachlan, A. Tran, N. Hopper, and Y. Kim. Scalable onion routing with Torsk. In S. Jha and A. D. Keromytis, editors, *ACM Conference on Computer and Communications Security*, pages 590–599. ACM, 2009.

[23] P. Mittal and N. Borisov. Information leaks in structured peer-to-peer anonymous communication systems. In P. F. Syverson and S. Jha, editors, *ACM Conference on Computer and Communications Security (CCS 2008)*, pages 267–278. ACM, 2008.

[24] P. Mittal and N. Borisov. ShadowWalker: peer-to-peer anonymous communication using redundant structured topologies. In S. Jha and A. D. Keromytis, editors, *ACM Conference on Computer and Communications Security (CCS 2009)*, pages 161–172. ACM, 2009.

[25] P. Mittal, N. Borisov, C. Troncoso, and A. Rial. Scalable anonymous communication with provable security. In *5th USENIX conference on Hot topics in security (HotSec'10)*, pages 1–16. USENIX Association, 2010.

[26] Prateek Mittal, Femi Olumofin, Carmela Troncoso, Nikita Borisov, and Ian Goldberg. PIR-Tor: Scalable anonymous communication using private information retrieval. In *Proceedings of the 20th USENIX Security Symposium*, August 2011.

[27] S. J. Murdoch and P. Zielinski. Sampled traffic analysis by internet-exchange-level adversaries. In N. Borisov and P. Golle, editors, *Privacy Enhancing Technologies Workshop (PET 2007)*, volume 4776 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 2007.

[28] Steven J. Murdoch and Robert N.M. Watson. Metrics for security and performance in low-latency anonymity systems. In *Proceedings of the 8th Privacy Enhancing Technologies Symposium (PETS 2008)*, July 2008.

[29] A. Nambiar and M. Wright. Salsa: a structured approach to large-scale anonymity. In R. N. Wright and S. De Capitani di Vimercati, editors, *13th ACM Conference on Computer and Communications Security (CCS 2006)*, pages 17–26. ACM, 2006.

[30] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy (S&P 2008)*, pages 111–125. IEEE Computer Society, 2008.

[31] T.-W. Ngan, R. Dingledine, and D. S. Wallach. Building incentives into Tor. In *14th International Conference on Financial Cryptography and Data Security (FC 2010)*, volume 6052 of *Lecture Notes in Computer Science*, pages 238–256. Springer, 2010.

[32] F. Olumofin and I. Goldberg. Revisiting the computational practicality of private information retrieval. In *15th International Conference on Financial Cryptography and Data Security (FC 2011)*, Lecture Notes in Computer Science. Springer, 2011.

[33] L. Øverlier and P. Syverson. Locating hidden servers. In *IEEE Symposium on Security and Privacy (S&P 2006)*. IEEE Computer Society, 2006.

[34] P. Palfrader. Download server descriptors on demand. `https://gitweb.torproject.org/tor.git/blob/master:/doc/spec/proposals/141-jit-sd-downloads.txt`, June 2008. Accessed February 2011.

[35] A. Panchenko, S. Richter, and A. Rache. NISAN: network information service for anonymization networks. In S. Jha and A. D. Keromytis, editors, *ACM Conference on Computer and Communications Security (CCS 2009)*, pages 141–150. ACM, 2009.

[36] M. Perry. Securing the Tor network, July 2007. Presented at Black Hat USA on July 2007.

[37] J.-F. Raymond. Traffic analysis: Protocols, attacks, design issues, and open problems. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *Lecture Notes in Computer Science*, pages 10–29. Springer-Verlag, 2000.

[38] M. Rennhard and B. Plattner. Introducing MorphMix: Peer-to-peer based anonymous internet usage with collusion detection. In S. Jajodia and P. Samarati, editors, *ACM Workshop on Privacy in the Electronic Society (WPES 2002)*, pages 91–102. ACM, 2002.

[39] M. Schuchard, A. W. Dean, V. Heorhiadi, N. Hopper, and Y. Kim. Balancing the shadows. In K. Frikken, editor, *9th ACM workshop on Privacy in the electronic society (WPES 2010)*, pages 1–10. ACM, 2010.

[40] M. Sherr, A. Mao, W. R. Marczak, W. Zhou, B. Thau Loo, and M. Blaze. A3: An extensible platform for application-aware anonymity. In *Proceedings of the Network and Distributed System Security Symposium (NDSS 2010)*, pages 247–266. The Internet Society, 2010.

[41] R. Snader and N. Borisov. A tune-up for Tor: Improving security and performance in the Tor network. In *Proceedings of the Network and Distributed System Security Symposium (NDSS 2008)*. The Internet Society, 2008.

[42] Robin Snader and Nikita Borisov. Improving security and performance in the Tor network through tunable path selection. *IEEE Transactions on Dependable and Secure Computing*, 2010. `http://dx.doi.org/10.1109/TDSC.2010.40` (preprint).

[43] L. Sweeney. Weaving technology and policy together to maintain confidentiality. *Journal of Law, Medicine and Ethics*, 25:98–110, 1997.

[44] P. Tabriz and N. Borisov. Breaking the collusion detection mechanism of MorphMix. In G. Danezis and P. Golle, editors, *Privacy Enhancing Technologies (PETS 2006)*, volume 4258 of *Lecture Notes in Computer Science*, pages 368–383. Springer, 2006.

[45] The Tor Project. Tor metrics portal, February 2011. http://metrics.torproject.org/.

[46] The Tor Project. Who uses Tor. `http://www.torproject.org/about/torusers.html.en`. Accessed Februrary 2011.

[47] A. Tran, N. Hopper, and Y. Kim. Hashing it out in public: common failure modes of DHT-based anonymity schemes. In S. Paraboschi, editor, *8th ACM Workshop on Privacy in the Electronic Society (WPES 2009)*, pages 71–80. ACM, 2009.

[48] 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 A. D. Keromytis and V. Shmatikov, editors, *ACM Conference on Computer and Communications Security (CCS 2010)*, pages 308–318. ACM, 2010.

A Study of the evolution of α

In this section we present an extended study of the evolution of α, the fraction of clients that know a relay e, depending on the size of the network and the selection algorithm. The larger α is, the more users participate in the aggregate profiles constructed by the adversary providing better privacy to the clients. Nevertheless this does not guarantee that clients' connections are unlinkable, or that they cannot be profiled; in the current Tor network, for example, this can still be done at the application layer [17, 36].

We recall that the fraction of clients that have knowledge of e is: $\alpha = (1 - (1 - \Pr[e])^b)$, where $\Pr[e]$ is the probability of choosing the PIR block containing e and depends on the algorithm used for the selection of relays. We consider three algorithms in our analysis:

Uniform: Clients select relays (and thus blocks) uniformly at random.

B/W: Clients select blocks with probability proportional to the proportion of the network bandwidth they contain. We use a snapshot of the Tor consensus directory collected on 9 February 2011, which includes $N_{relays} = 649$ eligible exit relays. Figure 6 shows the distribution of $\Pr[e]$ if the database held one descriptor per block.

SB(s): Clients choose blocks based on the relay index(es) they contain, given that relays are sorted

Figure 6: Bandwidth distribution in the current Tor network on 9 February 2011

in order of decreasing bandwidth. We implement the weighted selection algorithm by Snader and Borisov [41] with parameter s=1 and s=10. The former results in almost uniform choice amongst the relays, and the latter favors the selection of relays with high bandwidth, resulting in a very skewed distribution.

We test the protection offered by PIR-Tor for different network sizes, that we construct by scaling the current network (composed of 649 exit relays) by a factor of $s_r = \{1, 10, 50, 100, 400, 800, 1000, 1200, 1500\}$ (note that when $s_r = 1500$ the networks is formed by about $N_{relays} = 1$M relays). For each database size, we choose the number of blocks for optimal performance of the CPIR algorithm. We note that the results obtained in our analysis can easily be extrapolated to all scenarios considered in the performance evaluation in Sections 7 and 8.

Uniform selection. If clients select relays uniformly at random, $\Pr[e] = 1/N_{blocks}$, where N_{blocks} is the size of the exit relay database. In Figure 7 we show the distribution of the fraction α of clients knowing a relay. As expected, α decreases with the size of the database, but even for the largest database, $\alpha = 0.0067$. Therefore when the network has 100 000 users the adversary can only recover aggregate profiles of on average 670 clients. The downside of this selection algorithm is that it does not balance the load of the network.

BW selection. When the BW algorithm is used, relays are selected according to their bandwidth. As we only have a snapshot of 649 relays, to simulate the network growth we construct the distribution of larger network by concatenating copies of the original list downloaded from the Tor network. In Figure 8 we show the distribution of the fraction α when BW selection is implemented. For the largest database (rightmost box plot), the median relay is shared by a fraction $\alpha = 0.0012$, and the

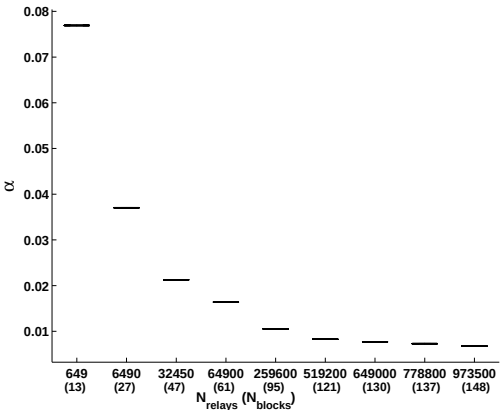


Figure 7: Uniform selection: evolution of α with the database size

least known relay is shared by a fraction $\alpha = 0.00036$ of clients. If there are 100 000 clients, the profile gathered contains information of on average 120 users for the former relay, and 36 users for the latter. The distant outlier in every box plot corresponds to the block that that holds the nodes with largest bandwidth. Hence, the aggregate bandwidth in this block is much larger than the bandwidth held by the next block in the database (and so is the probability of being chosen). As the network grows, blocks contain a smaller percentage of the relays in this database thus the distance between the most likely chosen block and other choices diminishes.

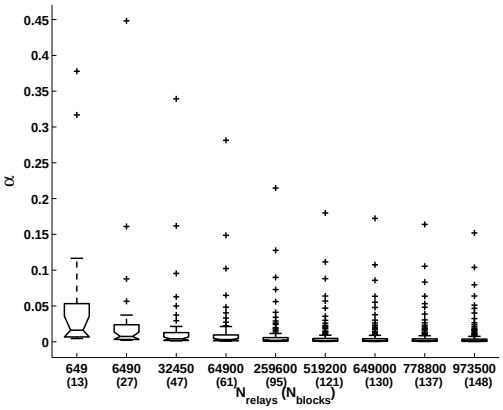


Figure 8: BW selection: evolution of α with the database size

SB selection. We show in Figures 9 and 10 the distribution of α for the SB(1) and SB(10) algorithms. In general, implementing SB(1) results in larger values of α than using SB(10), but SB(10) presents more variance, with the higher-bandwidth relays offering better protection than SB(1). When the database has 1M relays, the

median of α for SB(1) is $\alpha = 0.0065$, and for SB(10) it is $\alpha = 0.0019$. That means that the adversary collects aggregate profiles of 650 and 190 clients, respectively, when there are $100\,000$ users. We see that for SB(10) we get the same effect as for BW selection and the most chosen block of relays is far away from the rest of the distribution.

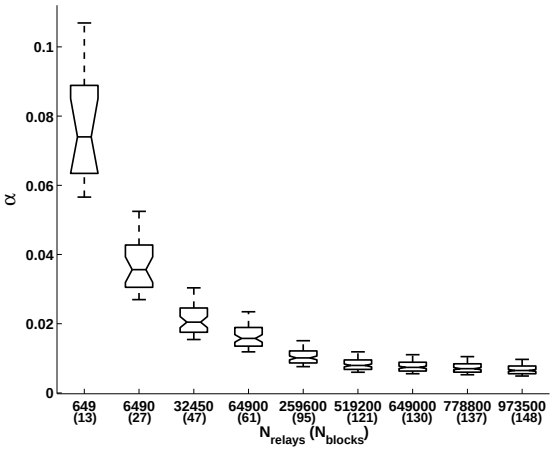


Figure 9: SB(1) selection: evolution of α with the database size.

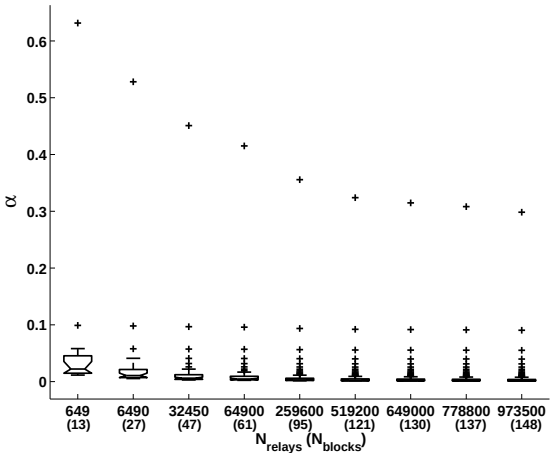


Figure 10: SB(10) selection: evolution of α with the database size.

Amongst the cases we have discussed in this section the worst-case scenario is for the client to select the least-chosen block for 152-block BW. In this case $\alpha = 3.6 \cdot 10^{-4}$. If there are only $100\,000$ clients in the network, the profile gathered at this relay would contain information from 36 clients. This number grows linearly with the number of clients. For the average number of Tor users in January 2011, roughly $175\,000$,[6] there are 63 users contributing to the profile, and this number increases up to 155 for the peak number of users in this period, $432\,075$. Further, we recall that linkability of circuits is only possible in intervals of 3 hours and 18 circuits, and that the amount of information gathered by the adversary at each exit node can be reduced by increasing the number of blocks b retrieved from the directory servers. Hence, it is unlikely that the adversary can isolate circuits belonging to a unique client based on relay selection algorithms.

[6] `http://metrics.torproject.org/users.html`