GuardedGossip: Secure and Anonymous Node Discovery in Untrustworthy Networks

Andriy Panchenko[1(✉)], Asya Mitseva[1], Torsten Ziemann[1], and Till Hering[2]

[1] Brandenburg University of Technology, Cottbus, Germany
{andriy.panchenko,asya.mitseva,torsten.ziemann}@b-tu.de
[2] RWTH Aachen University, Aachen, Germany
till.hering@rwth-aachen.de

Abstract. Node discovery is a fundamental service for any overlay network. It is a particular challenge to provide unbiased discovery in untrustworthy environments, e.g., anonymization networks. Although a major line of research focused on solving this problem, proposed methods have been shown to be vulnerable either to active attacks or to leak routing information, both threatening the anonymity of users. In response, we propose GuardedGossip—a novel gossip-based node discovery protocol—that achieves an unbiased random node discovery in a fully-decentralized and highly-scalable fashion. It is built on top of a Chord distributed hash table (DHT) and relies on witness nodes and bound checks to resist active attacks. To limit routing information leakages, GuardedGossip uses gossiping to create uncertainty in the process of node discovery. By incorporating the principles of DHTs with the unstructured nature of gossiping in a subtle way, we profit from the strengths of both techniques while carefully mitigating their shortcomings. We show that GuardedGossip provides a sufficient level of security for users even if 20% of the participating nodes are malicious. Concurrently, our system scales gracefully and provides an adequate overhead for its security and privacy benefits.

Keywords: Node lookup · DHT · Tor · Onion routing · Anonymity

1 Introduction

The Internet has become the most powerful medium for communication and information retrieval. Concurrently, an increasing number of repressive governments and other dominant entities endangers the free circulation of speech and information on the Internet by using censorship methods to restrict the open access to online content or even by prosecuting citizens who exercise their right to freedom of expression. For many people, the use of anonymization techniques is the only way to hide their identity (i.e., IP address) on the Internet and to bypass country-level censorship. Currently, Tor [11] is the most popular low-latency anonymization network comprising over 8,000 nodes run by volunteers and more than two million users daily [3]. Despite its continuous evolution, Tor still struggles with a

J. Garcia-Alfaro et al. (Eds.): SecureComm 2021, LNICST 398, pp. 123–143, 2021.
https://doi.org/10.1007/978-3-030-90019-9_7

limited scalability caused by its centralized node discovery method, i.e., a small set of trusted nodes keeps track of all active volunteer nodes. To preserve anonymity, each Tor user is required to have a complete network view and detailed information for at least 60% of the nodes (i.e., the current default value [2] versus 100% in the initial specification of Tor). The distribution of the global network view to each user creates at least three major problems. First, the maintenance of a consistent complete network view introduces a significant bandwidth overhead on Tor nodes and users. This overhead is foreseen to increase into such a level in near future that Tor would spend more bandwidth for maintaining a global network view than for anonymizing user connections [14]. Second, the list of all active volunteer nodes (except a special set of *hidden* volunteer nodes, called *bridges*), can be easily fetched by anybody. This makes them trivially blockable by any-level attackers [1]. Third, the small set of trusted nodes creates a single point of trust and failure, which makes them an attractive target for attacks from repressive governments and other dominant entities.

Several works focused on solving the scalability problem of Tor [14,18,20,23] or even proposed alternative anonymization systems with improved node discovery methods [12,19,21]. These methods are often divided into client-server based and peer-to-peer (P2P) based approaches [18,23]. While the client-server based methods provide an easy-deployable solution how to extend the capabilities of the current node discovery mechanism in Tor, they create additional hardware dependencies [23], increased computational costs to Tor nodes [18], and do not prevent the blocking of nodes by Internet censors. P2P-based node discovery services rely on either structured topologies, e.g., distributed hash tables (DHTs), or random walks and, thus, enable an efficient and scalable search of other nodes without having a global network view. However, these methods have been shown to be vulnerable to active attacks [6,24,27] (e.g., eclipse attack) and to passive information leakages, threatening the anonymity of users [16]. Despite the evolution of the proposed P2P-based node discovery methods, none of them provides a practical solution for a fully decentralized and scalable secure node discovery.

To address this challenge, we propose *GuardedGossip*—a novel gossip-based node discovery protocol, which achieves an unbiased random node discovery in a fully decentralized and highly scalable fashion. It is built on top of a Chord DHT and relies on witness nodes and bound checking to resist active attacks. To limit routing information leakages, GuardedGossip applies gossiping to introduce uncertainty in the process of node discovery. By incorporating the principles of structured topologies with the unstructured nature of gossiping, we aim to profit from the strength of both types of techniques while mitigating their shortcomings by carefully distributing the task. We provide a prototypic implementation and an evaluation of our method and show its superiority over existing approaches.

2 Problem Description and Attacker Model

Problem Description. A node discovery system aims to enable a random and unbiased selection of nodes from the set of all active peers in the network. A straightforward solution would be to use a centralized directory, from which

one fetches the list of available nodes and locally performs the random choice. Thus, the attacker has no influence on the selection process (except by deploying more malicious nodes) and there is no information leakage as all users have the same knowledge base for node selection. However, this solution creates several issues: (*i*) the need to trust a central directory keeping track of active nodes, (*ii*) the central directory is a single point of failure, (*iii*) it is easy to block such a system by either blocking the directory or the addresses of all the nodes, and, most notably, (*iv*) the use of a central directory limits scalability. To overcome these issues, it is appealing to use distributed methods, where each peer knows only a small portion of the network. However, this partial knowledge imposes new challenges. First, the randomness of the selection should not be affected by malicious peers (this is usually referred to as an *active attack*). In particular, the fraction of malicious nodes in lists discovered by users should not be significantly higher than their overall fraction in the network. Second, if only a small set of nodes gets discovered, it should be kept secret which node knows which other nodes. Otherwise, *bridging* and *fingerprinting* attacks [9] can be mount, in which the attacker restricts the set of possible users for (partially) observed anonymization connections based on user's knowledge about active nodes (these are known as *passive attacks*).

Attacker Model. In our work, we assume the attacker to be an observer and an active participant in the network that can generate, drop, or modify transmitted node discovery information. The attacker has a limited view on the network by controlling a restricted number of *colluding* nodes participating the network. We also assume that the fraction of the malicious nodes is static, i.e., the adversary cannot compromise other peers in the wild, and is below 30%—a typical hypothesis in the field of anonymous node discovery research [11,26].

3 Related Work

In the following, we review previously proposed secure node discovery systems.

P2P Solutions. Tarzan [12] is the only state-of-the-art method relying on gossiping for node discovery. In Tarzan, nodes need to maintain a complete network view [8], which was shown to not scale beyond 10,000 nodes [19]. Contrary to Tarzan, our system is built on top of a DHT and does not aim at discovering the whole network. We use gossiping to create uncertainty in the node discovery and to limit passive attacks against our underlying DHT-based methods. Other early proposed node discovery methods [28] rely on plain DHTs without any security mechanisms [6–8]. In response, Castro et al. [7] propose a set of defenses including a secure assignment of node identifiers (IDs) by using certificates and a secure message forwarding by using routing failure tests and redundancies. Mislove et al. [15] incorporate the secure lookup mechanism of [7] in a Pastry DHT [22]. Still, these methods remain vulnerable to active attacks [8,13].

More recent works [4,13,19] rely on redundant searches to build a secure node discovery system. The goal here is to execute several redundant searches for a single target, which traverse different nodes and, thus, reduce the impact of malicious

nodes trying to sabotage the lookup. Salsa [19] uses a custom DHT that has been shown to be insecure [6]. Kapadia and Triandopoulos [13] argue that the use of custom DHTs increases the overhead and propose to use a Chord DHT with indirect lookups. While being effective against active attacks, Mittal and Borisov [16] raise the issue that these methods leak routing information due to the redundancy and are prone to passive attacks. Panchenko et al. [20] show that redundant routing does not scale in terms of security. To mitigate routing information leakages while keeping resistance against active attacks, NISAN [20] relies on a Chord DHT, where each node retrieves and processes entire finger tables (FTs) of other nodes locally to keep the lookup destination hidden. Each NISAN node further applies bound checking to assess the plausibility of the retrieved FT. As in [20], Backes et al. [5] aims to hide the identity of the requested node by using oblivious transfer. Instead of adding anonymity into the lookup itself, Torsk [14] executes random walks to select secret buddy nodes for each peer. Each Torsk node uses one of its secret buddies as a proxy to perform a lookup on its behalf and, thus, hides the relationship between itself and the lookup destination. However, neither NISAN nor Torsk provide a sufficient level of security against passive and active attacks [27]. Octopus [26] aims to overcome known security issues by using redundancy, proactive identification of malicious peers, and dummy lookup requests to limit passive attacks. However, it still requires trusted third party and creates high communication overhead and complexity.

To prevent information leakage, other works use random walks for node discovery. In MorphMix [21] each node knows only a few neighbors. The user initiates a connection and each subsequent intermediate node is randomly selected by the next hop along the path. Each MorphMix node relies on witness nodes and a collision detection mechanism to detect manipulations. However, the collision detection method of MorphMix has been shown to be broken [25] and, hence, MorphMix does not have direct practical impact on new designs anymore. Unlike MorphMix, where witness nodes are used to select the next hop along the path, we rely on them in a passive way (to detect malicious replacements of node IDs in a fetched FT). ShadowWalker [17] is another proposal using random walks. To avoid a compromise of the walk, it organizes the nodes in a DHT and relies on a set of shadow nodes for each peer in the network. The goal of the shadows is to check the node's neighborhood information by signing the FT of that node. However, ShadowWalker is vulnerable to eclipse and DoS attacks [24].

Client-Server Solutions. Tor [10,11] relies on a set of predefined trusted nodes, called *directory authorities* (DAs), to maintain a global network view on available nodes, known as onion relays (ORs). The ORs periodically report their parameters to the DAs in a self-signed *relay descriptor*. Based on this data, the DAs agree on a network view and publish a list of active ORs together with their description in the form of a *consensus* document once per hour [2]. When joining Tor, users download the consensus to select ORs for their connections. Next, they fetch complete descriptors of at least 60% of all ORs (as long as it is not stated otherwise by the Tor users or instructed by the DAs). The fallback from initially 100% to 60% is one of the workarounds done to address the scalability issues faced by Tor. To reduce the load on the DAs, ORs can act as *directory caches* (DCs) that fetch copies of

the consensus and node descriptors from the DAs and host them for global distribution to the users and other ORs. The Tor Project further introduced the use of a *microdescriptor consensus* [2] and diffs of consensuses. While these countermeasures slow down the scalability issues, they do not provide a long-term solution for Tor's problem. Therefore, several works [18,23] focused on designing improved centralized methods combined with private information retrieval (PIR) techniques or Oblivious RAMs with trusted execution environments to allow users to obtain information about only a few ORs in an anonymous way. Mittal et al. [18] suggest PIR-Tor, where DCs can act as PIR servers and clients retrieve individual ORs from these caches. While PIR-Tor significantly decreases the network load, it puts additional computational overhead on the DAs and DCs due to the operation mode of the applied PIR methods. In response to the latter, Sasy and Goldberg [23] propose the use of Oblivious RAMs with trusted execution environments to reduce this bandwidth and computational overhead. However, in both methods the DAs still need to be trusted and to keep a global network view and handle joining and leaving nodes.

4 GuardedGossip: Secure and Anonymous Node Lookup

The evolution of P2P-based node discovery systems indicates a constant conflict between their increased level of security against active attacks and their reduced level of anonymity due to the efficacy of passive attacks [16,27]. To address the latter, we propose *GuardedGossip*—a novel gossip-based node discovery protocol that incorporates the use of a Chord DHT with witness nodes and bound checking to resist active attacks and the concept of gossip communication to add uncertainty in the process of node discovery and, thus, prevent passive attacks.

4.1 Design Overview

We assume that all nodes in GuardedGossip are organized in a Chord DHT with IDs generated by a pseudo-random function with a seed based on deterministic, node-specific input (as it will be described below). Thus, the nodes get distributed uniformly at random over the ID space of Chord and cannot influence their placement. In GuardedGossip, each node periodically sends gossip requests to get information about other nodes in the network. The use of the Chord DHT ensures that peers can send and receive gossip messages from a restricted set of other peers. Next, nodes directly collected though gossiping are neither considered for user connections nor for further distribution via gossiping. The reason for this is that gossiping is vulnerable to active attacks where malicious nodes share knowledge preferably about other malicious peers. To avoid this, we perform two additional checks to verify the trustworthiness of the collected peers. As in [20], we first retrieve the complete FTs of the nodes collected via gossiping and apply bound checking over them, i.e., we check whether the distance observed between optimal nodes (as if all IDs in the Chord would be occupied) and active nodes (an actually occupied optimal ID or the nearest existing successor of that optimal ID

in the direction of the Chord ring) in a FT corresponds to the average distance between any two nodes in the network. In the second verification step—witness check—we use nodes that have already been received by gossiping as witnesses to recognize malicious replacements in the newly obtained FTs (whether some nodes were skipped). Finally, only peers passing both security checks are used to transmit user traffic or to be further propagated via gossiping. Overall, the key principles of our method can be summarized as follows:

(*i*) *Limit Your Contacts.* The use of a Chord DHT ensures that gossip messages are sent or received by a limited number of nodes. A peer accepts gossip requests only from nodes that have it in their FTs and sends requests and accepts gossip replies only from nodes that are in its FT. This limits active attacks and allows to maintain scalability while using gossiping. Contrary to the common gossip concept [12], in GuardedGossip nodes *actively request* gossip information instead of passively listening for it. As gossips are coming only from nodes that were asked for, the number of potential gossip messages exchanged between peers is reduced. This ensures a moderate communication overhead in the network. The requestee can also limit the rate of answering gossip queries to protect against DoS attacks by another peer flooding it with a large number of gossip messages.

(*ii*) *Never Trust Your Gossips.* As gossiping is vulnerable to active attacks where the attacker primarily sends IDs of other malicious nodes, our method does not use the peers collected via gossiping directly for user connections and further propagation, but relies on them as a source to retrieve information about other network participants. GuardedGossip fetches the whole FTs of nodes received via gossiping and verifies their plausibility. Only if the check of a FT is successful, a random subset of peers from that FT is considered for further use (i.e., either for user connections or for further propagation via gossiping). Finally, after the FT was processed, the gossiped source peer becomes obsolete and is discarded.

(*iii*) *Know Your Witnesses.* GuardedGossip combines a distance check between nodes in a FT with a novel witness check. The witness check keeps track of nodes already observed via gossiping (including a timestamp of a last-seen event) and verifies whether any node in a FT was bypassed (i.e., there exist a node between an optimal node ID and the reported one), which indicates a manipulation.

(*iv*) *Spawn Uncertainty.* Once the collected FTs are successfully validated, nodes from these FTs are used either for user connections or for further distribution via gossiping. To further obscure which set of nodes is used, GuardedGossip spawns additional uncertainty by selecting only a *random subset* of newly learned nodes for further processing (except for the witness list, where all nodes are kept, as this list is used in a passive way only). Thus, we guard from passive attacks, in which one tries to estimate the search target range by evaluating information about the requested FTs [20,27].

4.2 Protocol Details

Figure 1 illustrates the operation of GuardedGossip. Here, we assume that each node has already joined the Chord DHT and has correctly built its FT. Later, we

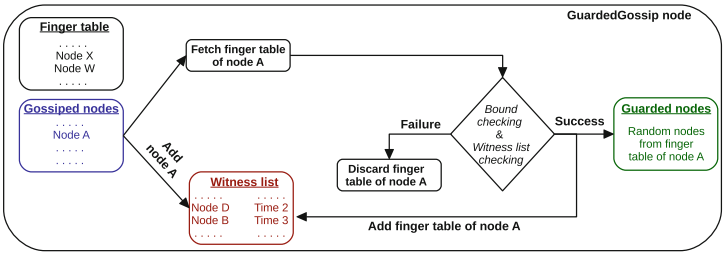


Fig. 1. GuardedGossip protocol workflow.

will discuss how to achieve this assumption. Beside its FT, each GuardedGossip node keeps track of three additional lists: *gossiped nodes*, *guarded nodes*, and *witness nodes*. While the list of gossiped nodes contains the set of new peers retrieved via gossip requests, the list of guarded nodes comprises those nodes that can be used for anonymizing user connections and for further propagation via gossiping. Initially, the list of gossiped nodes is empty, the list of witness nodes consists of all nodes seen during bootstrapping, and the list of guarded nodes contains peers, retrieved by performing secure lookups (i.e., using redundancies and bound checking as described below and in [20]) for random node IDs. Once a sufficient number of inputs for the list of guarded nodes is collected by the regular protocol workflow, the initial bootstrapping nodes (retrieved from secure lookups for random node IDs) become obsolete and are removed from the list. The bootstrapping nodes are never used for anonymizing user connections.

Exchange of Gossip Information. To keep the list of gossiped nodes and the list of guarded nodes loosely in sync and to reduce a potential linkability of both lists, the maintenance of these lists happens in regular random intervals initiated by the GuardedGossip node. To update its list of gossiped nodes, the GuardedGossip node picks one of its fingers (i.e., other nodes comprising the FT of that node) uniformly at random, and sends it a gossip request. The finger returns a set of node IDs in the interval $[0, 2]$, established experimentally as a reasonable trade-off between system security and performance. Only nodes that have received gossip requests and are included in the FT of the GuardedGossip node are allowed to send gossips. Thus, we limit possible gossip information exchange and keep a moderate communication overhead in the network. Next, for each received peer the GuardedGossip node checks whether the ID of that peer is already in the witness list. If it was last-seen recently, the peer is discarded (to avoid an adversary from repeatedly sending the same nodes via gossiping) and the last-seen timestamp of that peer in the witness list is updated. Otherwise, the peer is added to the list of gossiped nodes and the witness list with the current timestamp. After all node IDs retrieved via gossiping are processed, the GuardedGossip node checks whether the length of the gossiped list is within a predefined range and, if not, random elements are deleted (to spawn additional uncertainty about further use of nodes). The size of the witness list is limited by the timestamp of last-seen events. Outdated witness nodes are removed.

On the other side, the node receiving a gossip request checks if the requester is eligible to ask for gossips, i.e., it has that node in its FT. It replies only after successful check. Each of the to-be-sent node IDs are randomly chosen from the list of guarded nodes and, simultaneously, deleted from that list with a probability $p_r = \frac{1}{3}$. Thus, we limit the information leakage about the nodes that can be used for anonymizing user connections. Simply removing all sent IDs would make the receiver aware that these IDs are definitely no longer known by the sender and not used by it for user connections.

Verification of Collected Gossip Information. To build and maintain the list of guarded nodes, we rely on information collected via gossiping. However, the data in gossip messages is not trustworthy. As a consequence, an adversary could bias the lookup process by propagating mostly other malicious nodes if we would directly use gossiped nodes. Instead, we use the gossip information in an indirect way. We arbitrarily choose a set of nodes $\{v_0, \ldots, v_m\}$ from the list of gossiped nodes, where $m \in [0, 3]$ is a random number. We established this interval experimentally as a reasonable trade-off between system security and performance. Simultaneously, the selected nodes are removed from the list of gossiped nodes. Next, each of these nodes v_j, $j = 0, \ldots, m$, is requested to provide its full FT. Every received FT is validated using *bound checking* and *witness list checking*. If and only if both checks succeed, a uniformly random subset of nodes of size up to $n_o = 10$ is selected from the FT and added to the list of guarded nodes (we identified the boundary of 10 as a reasonable trade-off between the uncertainty gain and the frequency of newly collected guarded nodes). Otherwise, the FT is discarded. Thus, we achieve two key properties. First, only nodes from verified FTs are added to the list of guarded nodes. This significantly reduces the probability of biased selection of nodes due to manipulated FTs. Second, we create uncertainty in our choice as the node, from which the FT was fetched, does not get any information which of its fingers are selected for further usage. On average, five nodes are added to the list of guarded nodes for each peer selected from the list of gossiped nodes. As shown in Sect. 5, this ensures a system stability, i.e., at any given time a sufficient number of nodes are available to execute gossiping and to build anonymized user connections. Moreover, all nodes seen in a validated FT are also added to the witness list (or, if already present, their timestamps are updated).

Bound Checking. GuardedGossip uses the same bound checking method applied in NISAN [20]. Due to space constrains, we do not describe the operation of bound checking but refer interested readers to Appendix A for more details.

Witness List Checking. Although bound checking significantly reduces the probability for a biased node selection, it cannot completely eliminate the possibility of manipulating some node IDs in a FT. Therefore, we propose a second verification step, called witness check. Each GuardedGossip node maintains a witness list containing active nodes that have been seen before. On the receipt of a new FT from a gossiped node that has already passed the bound checking, the witness list is next used to further verify the plausibility of this FT. To this end, the optimal IDs of the given FT are initially computed. Then, we search in

the witness list for nodes whose IDs are closer to one of the optimal IDs than the nodes currently presented in that FT, i.e., a peer that has been skipped in favor of another node located more far away in the DHT. If such a peer is discovered, this is an indication for a potential malicious manipulation of the FT under verification. In this case, we either silently discard the complete FT with probability $p_c = \frac{1}{2}$ or further investigate this incident with probability $(1 - p_c)$. For our additional verification, we contact the closer node found in our witness list and check its availability. If the availability check succeeds, we discard this FT completely and update the timestamp of our witness node in the witness list. If the availability check fails, we remove the witness node from our witness list and accept the FT. The goal of the non-deterministic incident checking is to reduce the information leakage. It becomes uncertain for the adversary if a (possibly) slightly manipulated FT is accepted or not. Moreover, without non-determinism, one would possibly reveal the reason (i.e., the presence of an active witness node) for rejecting the manipulated FT and, thus, the attacker would gain intelligence about the content of client's witness list. Finally, an extended canonical Chord stabilization is periodically executed. Except Chord update, it purges all outdated nodes in the witness list according to their timestamps.

Creation of Anonymized User Connections. In GuardedGossip, the first node used for an anonymized user connection is selected from the list of guarded nodes. The nodes in this list have passed both bound checking and witness list checking. To further obscure the choice of nodes for anonymized user connections, the initiator of a transmission first creates a one-hop connection to the selected node. Then, the extension of this connection is executed via tunneling. This means that the initiator of the connection randomly selects a node from its list of gossiped nodes and performs a regular GuardedGossip operation via the tunnel to the first node. A random node from the validated FT is further used to extend the anonymized user connection. This process protects from inferences based on destinations of initiator's queries that can be observed by an adversary and is repeated until the necessary path length is reached. Typically, three nodes are used for a path creation in onion routing [11].

Bootstrapping New Nodes. When a new node joins the network, it first gets assigned a pseudo-random, unused ID from the ID space of the underlying DHT. However, the node should not be able to deterministically influence the selection of its ID and, thus, its placement at a certain position, e.g., close to another specific node. To this end, one can use a combination of an IP address and a global consensus, e.g., the hash of a public blockchain. Besides getting an ID, the newly joining node has to know an initial set of *bootstrapping* nodes. The IDs of these nodes are usually obtained out-of-band, e.g., announced by trusted third parties, received via email or from a friend-to-friend network, or distributed together with the software (as in Tor). As in related work [17,26], we assume that at least one of these nodes is not malicious. Next, the newly joining node uses a slightly modified version of NISAN node lookup [20] to bootstrap and introduce itself in the network. It uses its node ID to compute optimal node IDs for its FT and sends lookup requests for them (to fill in its own FT) and randomly selected

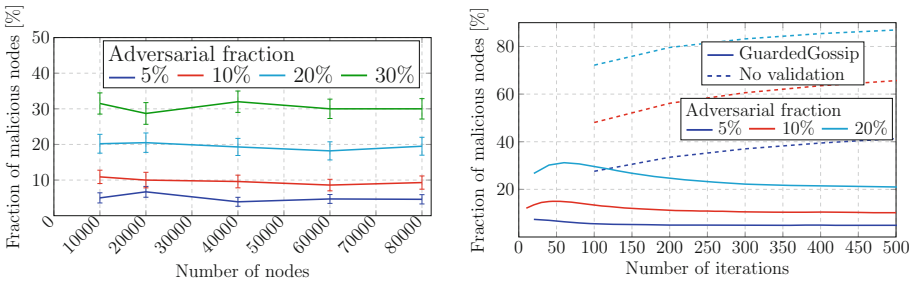
IDs (to fill in its guarded list) through its bootstrapping nodes. All nodes that are discovered during this lookup process and passed the bound checking are added to the witness list, together with a timestamp. If a peer already exists in the witness list, only its timestamp will be updated. Moreover, a randomly selected subset of nodes retrieved from the verified FTs is used to initialize the list of guarded nodes. During bootstrapping, the list of gossiped nodes is empty. This list is getting filled once the regular operation of GuardedGossip has started.

5 Evaluation

GuardedGossip aims to provide scalable node discovery while resisting active, e.g., route capture, and passive, i.e., information leakage, attacks. Here, we analyze whether GuardedGossip achieves these goals by means of simulations. As it relies on partial knowledge about active nodes in the network to gain scalability (i.e., each node knows only a portion of the whole network), it is important that this knowledge is unbiased, i.e., every node has equal chance to get known by a particular discovering peer. To this end, we also analyze the randomness of the node discovery process and the impact of churn on our method. We show that GuardedGossip achieves a sufficient level of security for users even if 20% of the participating nodes are malicious. Concurrently, our system scales gracefully and provides an adequate overhead for its security and privacy benefits.

Experimental Setup. The simulation of GuardedGossip is implemented in Python. All interactions between nodes are executed in discrete time steps. We repeated each experiment up to several hundred times to increase the significance of our measurements, presenting mean values together with 95% confidence intervals. The ID space size $N = 2^{32}$ of the Chord ring and the total number of nodes n in GuardedGossip, comprising up to $n = 10^5$ active nodes, are chosen as a trade-off between the goal to simulate as large network sizes as possible, and simultaneously limiting the amount of computational resources needed to handle our simulations. Our setup includes significantly more nodes compared to Tor, and the ID space is sufficiently sparsely populated to model realistic conditions.

Protection against Active Attacks. In GuardedGossip, the indicator for a successful protection against active attacks is the fraction of malicious nodes in the guarded list of each node, which should be similar to the overall fraction of malicious nodes in the network. On the other hand, the attacker aims to maximize the fraction of malicious nodes contained in the guarded lists of other honest nodes. To do this, the adversarial nodes respond on gossip requests with a random sample of IDs belonging to other malicious nodes instead of reporting nodes from their guarded lists. To avoid this, GuardedGossip uses witness list and bounds checking. To simulate a strong adversary, we assume global knowledge of the FT tolerance factor γ for bound checking (see Appendix A). Thus, malicious nodes can optimally maximize the fraction of colluding nodes FTs without being immediately detected. This simulates the worst-case active attacker scenario.



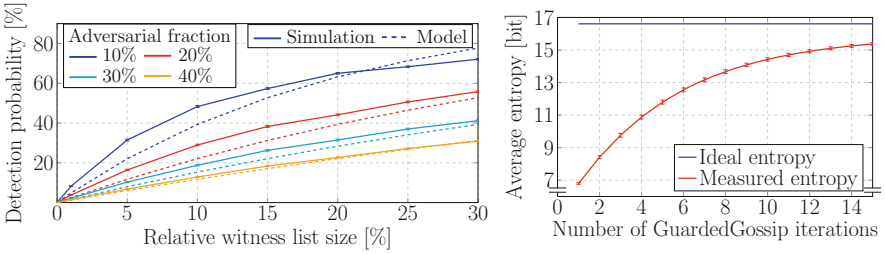
(a) Different network sizes.

(b) A network size of 10,000 nodes.

Fig. 2. Fraction of malicious nodes in guarded lists of honest nodes.

Figure 2a shows that the combination of both checking methods is an effective measure against active attacks. The fraction of malicious nodes in the guarded lists remains stable for different network sizes and fractions of malicious nodes. We also see that in each of the considered cases the fraction of malicious nodes contained in the lookup results is close to the theoretical minimum, i.e., the overall attacker fraction in the network. Figure 2b shows the dependency on the number of iterations (i.e., a single run of our method as shown in Fig. 1) and highlights the impact of FT checks. While without any validation the fraction of malicious nodes in guarded lists dramatically increases over time, our checking methods can stabilize the fraction of malicious nodes to a steady state close to the theoretical minimum. It is recommended to stay for at least 100 (or even 200) iterations in the system before starting building anonymized user connections to benefit from an effective witness check and filled guarded lists, as also suggested in Appendix C.

The witness check itself mainly depends on the size of the witness list measured relatively to the total number of nodes in the network (let ω denote this fraction). Given the ID space of size N, n active nodes, and a fraction f of malicious nodes in the system, the total number of adversarial nodes is $f \cdot n$ and the number of nodes in honest node's witness list is $n_\omega = \lfloor n \cdot \omega + 0.5 \rfloor$. Then, the number of possible witness lists of size n_ω is $t = \binom{N}{n_\omega}$. The average distance between two consecutive malicious nodes can be calculated as $\frac{N}{f \cdot n}$. Assuming uniform distribution of nodes in the ID space, the average distance between a randomly selected node and the next (in the direction of the Chord ring) malicious node is in the expectation value on the half of the distance between two consecutive malicious nodes, i.e., $n_m = \lfloor \frac{N}{2 \cdot f \cdot n} + 0.5 \rfloor$. On the other hand, this is exactly the average number of possible witnesses that would reveal a manipulation (as each of the nodes counted by the distance is a potential witness). Each witness list that contains at least one of these witnesses helps to reveal the manipulation. Let the number of these lists be d. First, we compute the number of all possible witness lists that do not have a single member to detect the manipulation $\overline{d} = \binom{N-n_m}{n_\omega}$. We then derive the detection probability that uses of the number of all possible witness lists having at least one member that reveals the



(a) Detection probability of single manipulated FT entries.

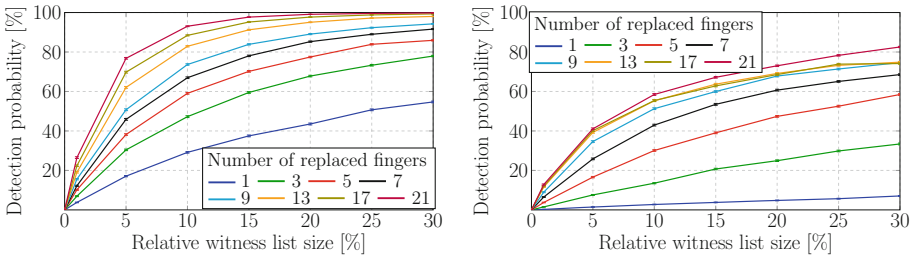
(b) Entropy per node in guarded lists for 100,000 active nodes.

Fig. 3. Detection of single manipulated FT entries and entropy of guarded lists.

manipulation of a FT: $p_{\text{detection}}(N, n, f) = \frac{d}{t} = 1 - \frac{\overline{d}}{t} = \frac{\binom{N}{\lfloor n\omega + 0.5 \rfloor} - \binom{N - \lfloor \frac{N}{2 \cdot f \cdot n} + 0.5 \rfloor}{\lfloor n \cdot \omega + 0.5 \rfloor}}{\binom{N}{\lfloor n \cdot \omega + 0.5 \rfloor}}$.

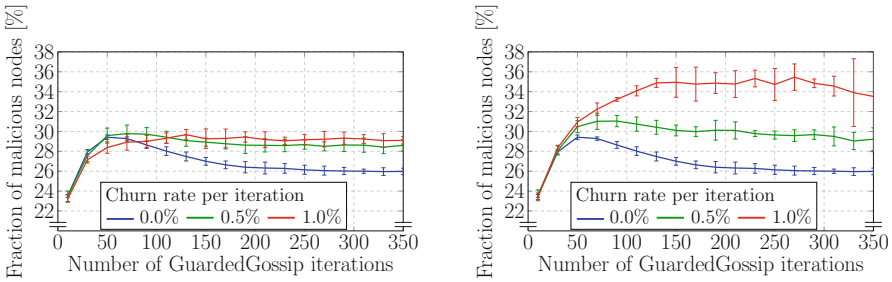
Figure 3a shows the effectiveness of this check in detecting a single manipulated member of a FT using the previous mathematical model and simulations. The success probability increases with the size of the witness list. However, a single manipulation is detected effectively only when the adversarial fraction in the network is low. While to achieve a detection probability of 50%, a relative witness list size of 15% is sufficient for 10% attacker fraction, we need a relative witness list size of 25% to achieve the same result for 20% attacker fraction. Overall, the witness check does not scale well with respect to the attacker fraction when only a single FT entry is manipulated. However, the situation becomes much better if the adversary replaces more than one entry. As shown in Fig. 4, if the attacker manipulates the FT in a random manner, a replacement of more than three entries will be detected with probability over 60% using a relative witness list size of 15% only. If the adversary replaces FT entries with the closest malicious nodes, changing less than five entries is still hard to detect with a reasonable relative witness list size, e.g., less than 30% of the total nodes. A good detection is achieved by changing at least seven nodes and assuming relative witness list size of 15%. Even a relatively small witness list size of 5%-10% already helps to detect manipulations. Still, as we saw before, the use of both witness checking and bound checking is the most effective protection against active attacks.

Network Churn. We analyze the resistance of our method to network churn. We performed a simulation where 0.5% and 1%, respectively, of all nodes leave the network per iteration, while the same number of new nodes joins the network. Hence, the network size remains stable. Please note that the change of 0.5% and 1% of the entire network per iteration implies is a huge fluctuation of the nodes. Figure 5 shows the impact of churn on the fraction of malicious nodes in the FTs of honest nodes. Here, we distinguish between honest nodes including or excluding the newly joined ones. Due to a deterministic bootstrapping process without gossiping, the newly joined nodes have more secure FTs and, hence, lower number of malicious nodes in their guarded lists. However, it is wrong to deduce that it is better to freshly join the network instead of staying within for some



(a) Random malicious nodes. (b) Closest malicious nodes.

Fig. 4. Probability to detect a malicious replacement of FT entries using witness checking in a network of size 10,000 nodes with 20% malicious nodes.



(a) Including newly joined nodes. (b) Excluding newly joined nodes.

Fig. 5. Fraction of malicious nodes in guarded lists of honest nodes for different levels of churn in a network of size 5,000 nodes with 20% malicious nodes.

time. Though performing a good job against active attacks, the nodes contained in the guarded lists of freshly joined nodes are deterministic in the beginning and do not provide protection against passive attacks (our method starts using the guarded list for anonymizing user connections only after the bootstrapping nodes are replaced by those obtained from a regular protocol behavior). High churn mostly impacts the witness list as many of the witness nodes become outdated. In this case, the major protection is achieved by the bound checking. Even under extremely high churn rate of 1% per iteration, honest nodes have a controllably stable fraction of malicious nodes in their guarded lists over the number of iterations (see Fig. 5a). Although the churn negatively influences the level of protection against active attacks, its impact is almost negligible as it is just 3% above the steady state of the system without churn.

Protection against Passive Attacks. We aim to obscure the identity of nodes that will be used for anonymizing connections. To measure the difficulty of estimating the nodes in the guarded list of other honest nodes, we evaluate the entropy of the lists elements. The theoretical maximum possible entropy of an element in a guarded list is given by $H_{\max} = \log_2(n)$. This ideal entropy could be achieved by assuming that lists are sampled uniformly at random from all

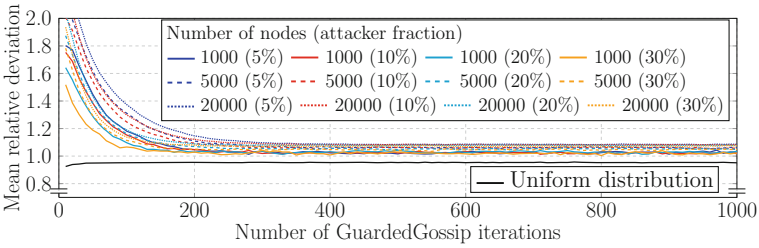
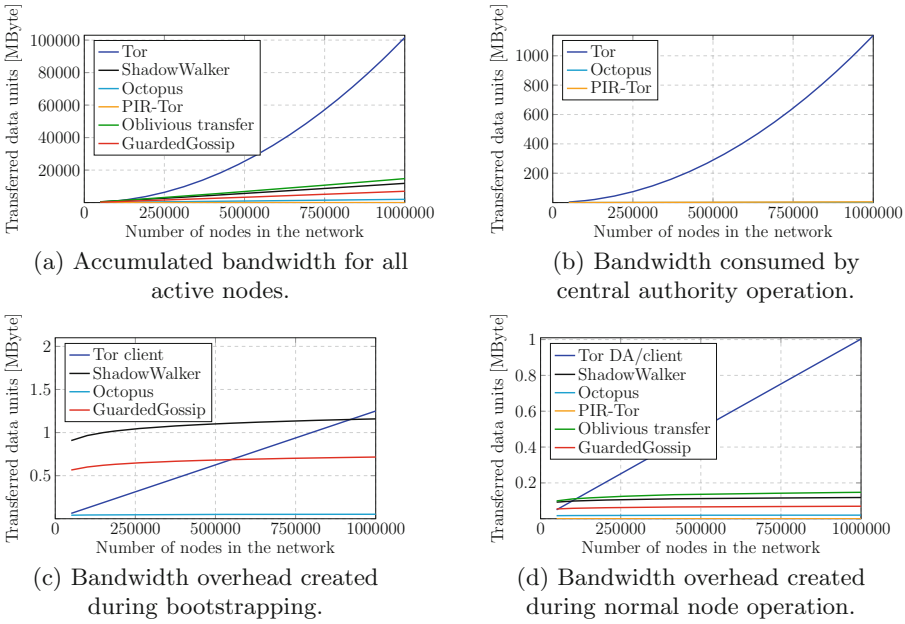


Fig. 6. Mean relative deviation of distances between nodes in lists of guarded nodes and the expected distances based the uniform distribution.

active nodes in the network. The average entropy per guarded node list member is monotonously converging to the ideal value above mentioned (see Fig. 3b). An acceptable value of entropy is already reached after only 15 iterations of our method. Hence, GuardedGossip leads to fast network information propagation and achieves sufficiently large entropy to render passive attacks impossible.

Randomness of Node Discovery. During bootstrapping, GuardedGossip undergoes deterministic measures to achieve its security properties. To identify the duration of the bootstrapping phase for different number of malicious nodes (i.e., how long does it take to achieve a sufficient level of node randomness in a guarded list), we analyze the distribution of nodes collected in individual guarded lists. Figure 6 shows the mean relative deviation of distances between nodes collected in guarded lists and the expected distances based on the uniform distribution. To compute the mean relative deviation, we first calculate the optimal average distance $\overline{d}$ between nodes in our Chord DHT, i.e., $\overline{d} = \frac{N}{e}$, where e is the size of a guarded list and N is the size of the ID space in the DHT. Then, we obtain the mean relative deviation of distances $mrd = \sqrt{\frac{1}{e}\sum_{j=0}^{e}\left(\frac{d_j - \overline{d}}{\overline{d}}\right)^2}$, where d_j, $j = 1, \ldots, e$, are the actual distances between members in guarded lists. Initially, the mean relative deviation is very high, compared to the uniform distribution. However, almost independently on the network size and the fraction of malicious nodes, the mean relative distance decreases significantly within the first 200 iterations. Then, the relative mean distance behave stable, being only slightly above the optimum deviation defined theoretically by the uniform distribution. Thus, the distribution of nodes in guarded lists is well normalized over time and it is safe to use these nodes after about 200 iterations. We also explore the average number of nodes in both, the gossiped list and the guarded list, and show that the number of nodes in both lists becomes sufficient after approximately 150 iterations. For more details about our results, see Appendix C.

To sum up, our method effectively hampers active attacks while being able to cope with churn and, thanks to its gossiping component, achieves high entropy in the guarded lists, rendering passive attacks impractical.



(a) Accumulated bandwidth for all active nodes.

(b) Bandwidth consumed by central authority operation.

(c) Bandwidth overhead created during bootstrapping.

(d) Bandwidth overhead created during normal node operation.

Fig. 7. Bandwidth overhead of GuardedGossip and other node discovery systems.

Overhead. We compare the amount of bandwidth overhead created by the most popular prior node discovery systems the Tor's node discovery method [11], ShadowWalker [17], Octopus [26], oblivious transfer [5], and PIR-Tor [18] to GuardedGossip. Here, we assume that GuardedGossip operates in a churn rate of 0.1% per iteration, whereas the total number of available nodes is kept constant by letting join the same number of nodes that have left the network. Each GuardedGossip node is assumed to perform 200 iterations before start building anonymization connections to ensure that the bootstrapping of that node is complete and it has a sufficient number of guarded nodes. For Tor, we assume the existence of 10 DAs, which agree on a new consensus every time step. All parts of the Tor directory protocol that are not needed for path construction are omitted and the distribution of node descriptors is done using *microdescriptors* to ensure higher bandwidth efficiency. ShadowWalker is simulated using 40 shadow nodes as in [24] to achieve sufficient security. Octopus uses a path length of four nodes as in [26] and is configured according to the original experimental setup in [26]. PIR-Tor uses three PIR servers providing the network consensus. For oblivious transfer, we use the method in [5] with a quorum size of 100 nodes.

Figure 7a shows the bandwidth overhead accumulated for all active nodes. While the amount of bandwidth required by Tor for increasing network sizes rises dramatically, the bandwidth consumption of the other methods remains nearly constant over the different network sizes. GuardedGossips achieves lower level of bandwidth overhead compared to ShadowWalker and oblivious transfer.

Although Octopus still creates the smallest amount of bandwidth overhead, it is not a pure P2P approach as it relies on a centralized mechanism for the detection of active attacks and the blacklisting of malicious nodes. Thus, the scalability of its centralized mechanism needs to be further explored for increasing network sizes. We also analyze the bandwidth overhead created due to the use of centralized servers. We focus on Tor, Octopus, and PIR-Tor, as they are the only node discovery systems that rely on centralized mechanisms. As shown in Fig. 7b, the bandwidth overhead created by the operation of Tor DAs rises almost quadratically for increasing network sizes. In contrast, the amount of bandwidth consumed by the centralized servers of Octopus and PIR-Tor is negligible.

Figure 7c shows the amount of bandwidth created during the bootstrapping phase of each method for different network sizes. GuardedGossips has comparably high bandwidth costs during bootstrapping due to the security checks applied when nodes join the network. Octopus consumes two orders of magnitude less bandwidth for bootstrapping compared to GuardedGossips. The amount data units transferred by Tor, ShadowWalker, and oblivious transfer during bootstrapping is in the same order of magnitude. We also observe that the bandwidth overhead created during bootstrapping depends on the node discovery method used and is not influenced by the network size. This, in turn, promises a graceful scalability in case of GuardedGossip and makes it an attractive candidate for larger network sizes. Finally, Fig. 7d shows the bandwidth consumption created during normal node operation (i.e., once the bootstrapping phase completes). While the amount of bandwidth overhead created by GuardedGossip and Octopus is similar, the bandwidth consumption of PIR-Tor and oblivious transfer is slightly higher. In contrast, the amount of bandwidth required by Tor rises dramatically for all network sizes.

6 Discussion and Conclusion

Contrary to Tor [11] and PIR-Tor [18], GuardedGossip is a pure P2P approach. Hence, it weakens the trust assumption and does not require any trusted third party and a single point of failure (even if it is distributed over several nodes). Although Octopus is also P2P based, it requires the use of a trusted authority actively involved in node operations. The major advantage of GuardedGossip is its scalability as its underlying structure is a well-studied scalable DHT. The gossiping component is influenceable by the nodes themselves. Depending on their demands and capacities, they can limit or increase the gossiping rate. As in Tor, in GuardedGossip it is easy to collect the complete list of available nodes (see Appendix B). Though not having this property would be beneficial to allow for better blocking resistance, our method and, to the best of our knowledge, all the prior related work fail to achieve this. Future work is required to design mechanisms that would make it difficult to enumerate all active nodes.

Since GuardedGossip, as a DHT-based approach, allows for secure and scalable node lookups in P2P designs, an interested reader could wonder how it can be integrated into Tor. If Tor would switch to a P2P concept, which would be

from the authors' perspective the preferred option for getting rid of trusted DAs and improving scalability, then the GuardedGossip could serve as a good design candidate for node lookups. In the current Tor concept, a possible solution could be to let users select their trusted entry OR (similar to the secret buddy concept in Torsk [14]) as a proxy to execute the GuardedGossip protocol on their behalf. This trusted entry OR should be used for longer time period (as the guard ORs in Tor). Although such a mechanism offers less security than in a P2P design, this method would allow for scalable network information distribution without requiring to trust the same third party (i.e., the Tor DAs) by all users.

To sum up, node discovery and lookup methods form an integral part of any overlay network. Their use in untrustworthy environments—such as the Internet—brings additional challenges with respect to possible attacks. Especially in anonymous communication, it is important to provide secure and unbiased discovery as otherwise the anonymity of millions of people using these networks and reling on their properties can be compromised. In this paper, we proposed GuardedGossip—a secure, decentralized network information distribution system based on the combination of gossiping with a Chord DHT. This is done without trusting any third party. Our system scales gracefully and provides an adequate overhead for achieving its superior security properties.

Acknowledgments. Parts of this work have been funded by the EU and state Brandenburg EFRE StaF project INSPIRE.

A Bound Checking

Given a FT, bound checking is performed as follows: Initially, the node density d is computed. Ideally, this can be done by deriving $d = \frac{N}{n}$, where N is the ID space size of the DHT and n is the number of active nodes. However, n is usually not known. Thus, each peer computes means of the distance between the actual IDs in its FT and optimal IDs (as if all IDs would exist). The mean distance is then multiplied with a *finger table tolerance factor* $\gamma > 0$. Finally, to verify the plausibility of a given FT g the GuardedGossip node applies the following constraint $d_g < \gamma d$ to check whether the FT g is manipulated. In our work, we use $\gamma = \sqrt{\frac{1}{f}}$, where f is the (supposed) fraction of colluding malicious nodes. As the actual fraction of colluding malicious nodes is not known by the users, γ corresponds to the estimated maximum fraction of malicious nodes that is supposed to be tolerated by our approach.

To assess the optimal value of γ, we rely on the approach used to detect routing failures in [7]. As the value of γ depends on the false positives rate α (i.e., a correct FT is falsely detected as manipulated) and the false negatives rate β (i.e., a manipulated FT is detected as correct), here we aim to minimize both, α and β, simultaneously. To this end, we deal with the total number n of all uniformly sampled active nodes within the ID space N. According to [7], the distances between consecutive node IDs within a FT can be modeled by independent exponentially distributed random variables with a mean of $\frac{N}{n}$. Similarly, the

distance between consecutive malicious nodes can be modeled using exponential distribution with a mean of $\frac{N}{f \cdot n}$, where f is the fraction of malicious nodes. We construct a new random variable by summing up the distances between nodes in the FT of the verifying node (denoted as S_o) and the distances between nodes in the retrieved FT to be verified by that node (denoted as S_v). Given that k denotes the FT size, the values of $\frac{1}{k} \cdot S_o$ and $\frac{1}{k} \cdot S_v$ measure the mean distances between node IDs in the FT of the verifying node and the retrieved FT to be verified by that node using bound checking. Now, the false positive rate α can be expressed by the probability $\alpha(k, \gamma, f) = Pr\left(\frac{1}{k} \cdot S_v > \gamma \frac{1}{k} \cdot S_o\right)$. Similarly, the false negative rate β is computed as follows: $\beta(k, \gamma, f) = Pr\left(\frac{1}{k} \cdot S_v < \gamma \frac{1}{k} \cdot S_o\right)$. According to [7], the false positives and false negatives get minimized if $\alpha = \beta$. Thus, we obtain the following equation: $\alpha(k, \gamma, f) = \beta(k, \gamma, f)$, from which γ can be computed. Moreover, S_o and S_v, are γ-distributed with a shape parameter k and scale parameters $\frac{n}{N}$ and $\frac{f \cdot n}{N}$, correspondingly. The use of the gamma distribution and solving the equation by γ finally yields $\gamma = \sqrt{\frac{1}{f}}$.

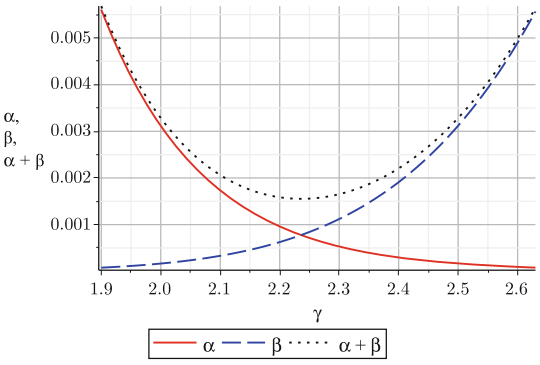


Fig. 8. Bounds checking false positive rate α, false negative rate β, and objective function for optimization $\alpha + \beta$ over the FT tolerance factor γ for $f = 0.2$.

We performed experiments to check these theoretical results. Figure 8 shows the bounds checking false positive rate α, the false negative rate β and the objective function for the optimization of $\alpha + \beta$, which is intended for minimization over the threshold $\gamma > 0$. As suggested in [7], the minimal cumulated error is achieved in case of $\alpha = \beta$, which justifies our computation of γ.

B Completeness of Node Discovery

To achieve a sufficient level of anonymity, GuardedGossip should deliver random nodes from the set of all active nodes in the network. Thus, the attacker cannot influence the selection of nodes for anonymized user connections except by injecting more peers in the network. As the list of guarded nodes is the only

source in our approach providing nodes for building user connections, we analyze its content with respect to its completeness. Figure 9 shows the fraction of nodes collected in individual lists of guarded nodes during the operation of GuardedGossip over time. For an increasing number of iterations, the fraction of nodes observed in guarded lists (measured by 95% quantiles) converges to 100%. Already after 1000 iterations, almost all GuardedGossip nodes have seen other active nodes forming the half of the network. This also confirms the expectation that staying longer in the network increases the coverage of discovered nodes. Every node also gets a chance to be discovered by all network participants.

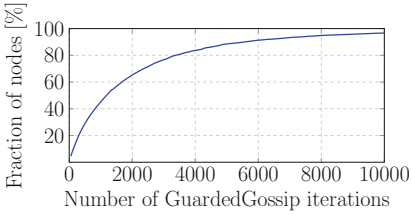


Fig. 9. Fraction of nodes in guarded lists measured by 95% quantiles for a network of size 2,000 nodes.

C Number of Nodes in Gossiped and Guarded Lists

We also explore the average number of nodes contained in both the gossiped list and the guarded list. Figure 10 shows that the size of the guarded list rapidly increases already in the beginning of operation of our approach, i.e., after 20 iterations. On the other hand, the number of nodes in the gossiped list increases slower, i.e., after 100 iterations it becomes stable. The guarded list grows faster as it gets filled by the whole FTs whereas the gossiped list only by single entries. We observe that the size of the network does not influence the bootstrapping process of the guarded list and has only a moderate impact on the size of gossiped list. Naturally, for an increasing network size the steady state of the gossiped list is reached earlier as more nodes participate in the gossiping process. However, even for small networks with 1000 nodes we achieve a sufficient number of nodes after

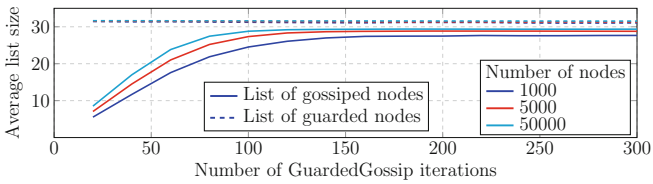


Fig. 10. Average number of nodes in gossiped and guarded lists for different network sizes with 10% adversarial nodes.

approximately 150 iterations. This ensures fluctuations within the lists (once both lists are completely filled, some nodes are discarded in favor of new nodes) so that it becomes more difficult for an attacker to promote favorable nodes due to the constant rotation of nodes in the gossiped list.

References

1. Torproject.org Blocked by GFW in China: Sooner or Later? (2008). https://blog.torproject.org/torprojectorg-blocked-gfw-china-sooner-or-later
2. Tor Directory Protocol, Version 3 (2018). https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt
3. Tor Metrics (2021). https://metrics.torproject.org/
4. Artigas, M.S., et al.: Cyclone: a novel design schema for hierarchical DHTs. In: P2P (2005)
5. Backes, M., et al.: Adding query privacy to robust DHTs. In: ASIA CCS (2012)
6. Borisov, N., et al.: Denial of service or denial of security? How attacks on reliability can compromise anonymity. In: CCS (2007)
7. Castro, M., et al.: Secure routing for structured peer-to-peer overlay networks. In: OSDI (2002)
8. Danezis, G., Clayton, R.: Route fingerprinting in anonymous communications. In: P2P (2006)
9. Danezis, G., Syverson, P.: Bridging and fingerprinting: epistemic attacks on route selection. In: Borisov, N., Goldberg, I. (eds.) PETS 2008. LNCS, vol. 5134, pp. 151–166. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70630-4_10
10. Dingledine, R., Mathewson, N.: Tor Protocol Specification (2018). https://gitweb.torproject.org/torspec.git/plain/tor-spec.txt
11. Dingledine, R., et al.: Tor: the second-generation onion router. In: USENIX Security Symposium (2004)
12. Freedman, M.J., Morris, R.: Tarzan: a peer-to-peer anonymizing network layer. In: CCS (2002)
13. Kapadia, A., Triandopoulos, N.: Halo: high-assurance locate for distributed hash tables. In: NDSS (2008)
14. McLachlan, J., et al.: Scalable onion routing with Torsk. In: CCS (2009)
15. Mislove, A., et al.: AP3: cooperative, decentralized anonymous communication. In: ACM SIGOPS European Workshop (2004)
16. Mittal, P., Borisov, N.: Information leaks in structured peer-to-peer anonymous communication systems. In: CCS (2008)
17. Mittal, P., Borisov, N.: ShadowWalker: peer-to-peer anonymous communication using redundant structured topologies. In: CCS (2009)
18. Mittal, P., et al.: PIR-Tor: scalable anonymous communication using private information retrieval. In: USENIX Security Symposium (2011)
19. Nambiar, A., Wright, M.: Salsa: a structured approach to large-scale anonymity. In: CCS (2006)
20. Panchenko, A., et al.: NISAN: network information service for anonymization networks. In: CCS (2009)
21. Rennhard, M., Plattner, B.: Introducing MorphMix: peer-to-peer based anonymous internet usage with collusion detection. In: WPES (2002)

22. Rowstron, A., Druschel, P.: Pastry: scalable, decentralized object location and routing for large-scale peer-to-peer systems. In: IFIP/ACM Middleware (2001)
23. Sasy, S., Goldberg, I.: ConsenSGX: scaling anonymous communications networks with trusted execution environments. In: PETS (2019)
24. Schuchard, M., et al.: Balancing the shadows. In: WPES (2010)
25. Tabriz, P., Borisov, N.: Breaking the collusion detection mechanism of MorphMix. In: PETS (2006)
26. Wang, Q., Borisov, N.: Octopus: a secure and anonymous DHT lookup. In: ICDCS (2012)
27. Wang, Q., et al.. In search of an anonymous and secure lookup: attacks on structured peer-to-peer anonymous communication systems. In: CCS (2010)
28. Zhuang, L., et al.: Cashmere: resilient anonymous routing. In: NSDI (2005)