NISAN: Network Information Service for Anonymization Networks

Andriy Panchenko
Computer Science dept.
RWTH Aachen University
D-52074 Aachen, Germany
panchenko@cs.rwth-aachen.de

Stefan Richter
Computer Science dept.
RWTH Aachen University
D-52074 Aachen, Germany
richter@cs.rwth-aachen.de

Arne Rache
Computer Science dept.
RWTH Aachen University
D-52074 Aachen, Germany
arne.rache@rwth-aachen.de

ABSTRACT

Network information distribution is a fundamental service for any anonymization network. Even though anonymization and information distribution about the network are two orthogonal issues, the design of the distribution service has a direct impact on the anonymization. Requiring each node to know about all other nodes in the network (as in Tor and AN.ON – the most popular anonymization networks) limits scalability and offers a playground for intersection attacks. The distributed designs existing so far fail to meet security requirements and have therefore not been accepted in real networks.

In this paper, we combine probabilistic analysis and simulation to explore DHT-based approaches for distributing network information in anonymization networks. Based on our findings we introduce NISAN, a novel approach that tries to scalably overcome known security problems. It allows for selecting nodes uniformly at random from the full set of all available peers, while each of the nodes has only limited knowledge about the network. We show that our scheme has properties similar to a centralized directory in terms of preventing malicious nodes from biasing the path selection. This is done, however, without requiring to trust any third party. At the same time our approach provides high scalability and adequate performance. Additionally, we analyze different design choices and come up with diverse proposals depending on the attacker model. The proposed combination of security, scalability, and simplicity, to the best of our knowledge, is not available in any other existing network information distribution system.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*

General Terms

Security

Keywords

Anonymous Communication, Peer-to-Peer, DHT, Node Lookup, Privacy

1. INTRODUCTION

While cryptography can be used to protect integrity and confidentiality of the data part of packets, everyone along their route can still observe the addresses of the communication parties. Anonymous communication deals with hiding relationships between communicating parties.

Many approaches have been proposed in order to provide protection on the network layer. Still, only some of them have been implemented in practice, e.g. [8, 3, 19]. The most popular and widespread system is Tor [8]. The Tor network is a circuit switched, low-latency anonymizing network for preserving privacy on the network layer. It is an implementation of the so-called *onion routing* technology [22], that is based on routing TCP streams through randomly chosen paths in a network of routers using layered encryption and decryption of the content. The number of servers in the network is currently about two thousand whereas the number of users is estimated to be hundreds of thousands [18, 24].

In order to discover the network information, most systems make use of a centralized directory service and assume that each node has to know all other nodes in the network. This has several advantages, such as making it hard for an attacker to poison the directory with forged entities, mount the eclipse attack [21], or bias path selection. However, this approach has scalability problems and requires trusting the directory. In Tor, for example, clients choose paths for creating circuits by selecting three suitable servers from a list of all currently available routers, the so-called *directory*. To this end, certain trusted nodes act as *directory servers* and provide signed documents that are downloaded by users periodically via HTTP. Such a network status document contains *router descriptors* of all currently known servers, including several flags that are used to describe their current state. Due to scalability issues, there have been changes in the original protocol and its successors, so that Tor is now already using the third version of the directory protocol. In the design document, the authors admit that requiring each node to know all the others will not work out in the long run because of scalability issues [7]. On that account, we strive to make network information scale by carefully distributing the task.

2. RELATED WORKS

The Tarzan [9] P2P network requires every peer in the network to know all the others. To achieve this, the authors use a simple *gossiping* protocol to find every other peer in the network (the protocol, though, is only described on a high level). For the initial join to the network a Tarzan peer needs to know at least a few other peers. The approach is believed to not scale beyond 10.000 nodes [17]. In contrast to this, MorphMix [20] requires nodes to know only a few neighbors. For the circuit setup so-called *witness nodes* are used to facilitate the selection of nodes for circuit extension; this is a unique feature in MorphMix, as in most other networks the users choose the path themselves for security reasons. In order to alleviate possible security issues arising from this feature, MorphMix uses a collusion detection mechanism, detecting malicious nodes that misbehave by offering other colluding[1] nodes for traversal. However, due to the vulnerability of their collusion detection mechanism [23], this approach does not have direct practical impact on new designs any more.

Salsa [17] is a DHT which has been specifically developed for anonymization networks. Identities are based on hashes of the nodes' IP addresses and are organized in a tree structure. Redundancy and bounds checking are used while doing lookups in order to prevent malicious nodes from returning false information. According to simulations, the scheme prevents attackers from biasing the path selection as long as the fraction of malicious nodes in the system does not exceed 20%. However, further analysis [15] has shown that even in this case Salsa is less secure than previously thought: with the number of corrupt nodes below 20%, still more than a quarter of all circuits are compromised because of information leakage. Because defending against active attacks requires higher redundancy, this directly increases the threat of passive attacks since a small fraction of malicious nodes can observe a significant part of the lookups. This compromises anonymity by leaking information about the initiator and the nodes involved in a tunnel.

Castro et al. [5] consider security issues in structured P2P networks. The attacker model consists of a fraction of colluding nodes. To thwart the attacks, the authors propose three techniques: (*i*) secure assignment of node identifiers, (*ii*) secure routing table maintenance, and (*iii*) secure message forwarding. They show that the overhead of their technique is proportional to the fraction of malicious nodes and is efficient if the attacker controls up to 25% of nodes in the system.

Their approach is not applicable in our scenario because of the following reasons: recursive queries, a centralized trusted third party, and information leakage. The use of recursive queries is not practical because it allows an attacker to execute DoS attacks with very small effort. It has been shown by Borisov et al. [4] that DoS attacks applied in anonymous communications considerably reduce the provided anonymity. That is why there is a strong need to assure reliability against adversaries, not just against random failures. Additionally, a client in the approach of Castro has the possibility to check only the final result of the query, while not being able to control and influence intermediate steps. Moreover, the need of a trusted third party hardens practical realization of the approach. Finally, information leakage [15] enables bridging and fingerprinting attacks [6].

Baumgart and Mies [2] propose several improvements for Kademlia [13] in order to make it more resilient against attacks. The improvements are threefold: using crypto puzzles for restricting node ID generation; sibling broadcast for ensuring replicated data storage; using multiple disjoint paths for node lookups. The most significant proposal is the secure node ID generation, since the latter two are merely refinements of mechanisms already existing in Kademlia. The crypto puzzles restrict the node ID generation in the sense that it is computationally expensive to generate valid node IDs. However, the adversary is free to generate valid IDs offline without any time bounds before actually joining and subverting the network. Moreover, in Section 4 we prove that the security properties of the latter three approaches cannot be upheld with growing network size.

Westermann et al. [25] study default Kademlia behavior w.r.t. colluding nodes that answer queries with malicious nodes closest to the searched-for ID. Their finding is that, due to redundancy, the fraction of malicious nodes found in queries is not significantly larger than the overall fraction of malicious nodes in the system. The simulations, however, have only been conducted up to $50,000$ nodes.

Kapadia et al. [10] propose a method for performing redundant searches over a Chord-based DHT. In a network of 10,000 their approach is able to tolerate up to 12% colluding nodes while applying a non-recursive search.

Recently, Awerbuch and Scheiderler [1] have published a theoretical approach to building a scalable and secure DHT that might potentially solve many of the problems we are facing in this application. A multitude of active attacks are actually proven to be impossible on this structure, even though passive information leakage attacks are not considered. Unfortunately, the downside of the formal approach taken is that the high level of vantage does not allow for instant practical implementation. In fact, the authors "believe that designing such protocols is possible though their design and formal correctness proofs may require a significant effort" [1]. Even if this belief is correct, it is foreseeable that it would lead to a very complex system, hard to analyse and implement correctly, while in the networking/anonymity community we observe a trend towards security by simplicity.

3. ATTACKER MODEL

We consider a local attacker with the following capabilities:

- Passively observe some portion of network traffic;
- Actively operate its own nodes or compromise some fraction of honest nodes;
- Actively delete, modify and generate messages.

Further we assume that the adversary cannot break cryptographic primitives. This is a standard assumption in the area of anonymous communication.

4. WHY REDUNDANCY THROUGH INDEPENDENT PATHS DOES NOT SCALE

One basic idea for finding a random node in a DHT is choosing a random number x in the node identity hash space and then use the DHT to look up the owner of x. Most often this is the node whose identity is closest to x in the distance metric of the DHT. This idea lies at the heart of approaches like Salsa [17] and AP3 [14], and its employment opens up the possibility of using any DHT as a building block that deems to fit for the purpose.

In this paper, we will follow this basic approach while analyzing possible attacks and giving remedies in a cumulative fashion. A naïve implementation of searching for x in an environment with a fraction f of collaborating adversarial nodes has success probability $p_s = (1-f)^l$, where l is the length of the search path, when we assume the adversary to either simply drop requests in a denial of service fashion or return false information such as claiming to be the owner of x. Because l has to grow larger with the size of the network, typically on the order of $\log(n)$ [12], this success probability approaches zero for growing networks asymptotically, when

[1]We use *colluding* and *malicious* w.r.t. nodes as synonyms in this paper.

f is, say, constant. Thus, we can say that, under this simple attack, the naïve implementation does not scale.

Of course, earlier scholarship [17, 5, 2] has recognized this problem. As a solution, redundancy in the form of several search paths has been introduced without fail. The cited papers all try to ensure routing towards x using multiple, preferably independent, paths. This is itself a difficult proposition, because the structure of DHTs usually leads to path convergence in a lot of cases, and quite some auxiliary constructions have been taken to still ensure independence. Yet, even when we assume independent paths, the redundancy required endangers scalability in the limit:

THEOREM 1. *The number of paths required to reach constant success probability p_s is at least $\frac{p_s}{(1-f)^l} \in n^{\Omega(1)}$ for $l \in \Omega(\log n)$ and f constant.*

PROOF. Let I_i be a random indicator variable that takes the value 1 if path i is attacker-free, and 0 otherwise. A single path i is free from attackers with probability $(1-f)^l$, thus $\mathbf{E}(I_i) = (1-f)^l$. Then $\sum_{i=1}^{\alpha} I_i$ is a variable that counts the number of successes out of α paths, and

$$p_s = P[\sum_{i=1}^{\alpha} I_i \geq 1] \leq \mathbf{E}(\sum_{i=1}^{\alpha} I_i) = \sum_{i=1}^{\alpha} \mathbf{E}(I_i) = \alpha(1-f)^l$$

by the Markov inequality and linearity of expectation. Notice that we have not made any assumptions about path independence.

Solving for α, substituting $\Omega(\log n)$ for l and assuming f, p_s constant yields

$$\alpha \geq \frac{p_s}{(1-f)^l} \in (1/(1-f))^{\Omega(\log n)} = n^{\Omega(1)}.$$

□

With simple greedy routing, the lower bound of $\Omega(\log n)$ path length is valid for a large class of small world networks, including all the DHTs we know of, as well as skip graphs [12]. Therefore, the theorem shows that the number of independent paths needed to route successfully with some fixed probability grows at least polynomially in n. This is exponentially greater than, e.g., the number of neighbors a node has in the most common DHTs. Thus, even in the first step of routing, we rapidly run out of independent possibilities to choose from with growing networks. The aforementioned research efforts [17, 5, 2, 25] evaluated their approaches experimentally in environments between 10,000 and 100,000 nodes. While their results suggest practicability in this range, the above considerations show that we cannot expect this to hold up when the networks grow larger. Moreover, our simulation of [25] suggests that already starting from 40,000 peers their approach fails due to almost double fraction of found colluding nodes by random look-ups compared to the overall fraction of colluding nodes in the system.

One might argue that a constant fraction of collaborating nodes assumes a very strong adversary, and there have been results that only work with fewer adversarial nodes [11], yet there are multiple reasons why we consider this adversarial model here: Firstly, most practical suggestions to solving the problem have assumed fixed percentages of malicious nodes [17, 5, 2]. Secondly, real world phenomena like botnets suggest that we might have to deal with strong attacks like this.

Thirdly, as we will show in the next section, we can do better. Instead of striving to make paths independent, we want them to work together in order to reach x more reliably.

5. NISAN

In this section we describe our approach. It consists of several steps. We begin with a simple Chord-like DHT, and start building protection measures on top of it in order to reach the required properties. Chord [16] has been chosen because of its simplicity, and, most notably, because its finger table entries are deterministic in the sense that there is exactly one correct neighbor for each finger in a given DHT. Moreover, since the Chord distance metric is directed, there is asymmetry in the sense that a node does not usually belong in the finger tables of its neighbors. We will use the first property to restrict malicious behavior in Section 5.3, and the second one against a stronger adversarial model in Section 6.3. Still, both our protection measures and scalability bounds, especially outside of these two sections, carry over to a very generic class of DHTs. For example, the results in the following section have serious implications for the security of Kademlia [13]. We do, however, decidedly prefer iterative search over recursive search, because, among other reasons, firstly we will use the added control we gain over the course of the search in our protection measures, and secondly, we consider the increased potential for denial-of-service attacks in recursive approaches as a too big threat to ignore.

5.1 Better Redundancy: Aggregated Greedy Search

Most of the approaches studied before make use of redundant independent lookups. This leads to a convergence on many paths. Therefore we follow another approach: instead of making independent lookups, we propose to use an aggregated search which combines the knowledge available on each of the independent branches. We call this *aggregated greedy search.*

It proceeds as follows. To find a random node, the searcher v generates a random ID x. At this point we assume that node IDs are uniformly distributed in the ID space (another case is considered in Section 6.3). In each round v chooses the α nodes closest to x that he is aware of and queries them for x. The search terminates when after one iteration the list of α closest peers[2] has not been changed. The owner of x (the peer which is closest to the searched ID) is the result of the query.

Interestingly, this description almost fits the behavior described in the Kademlia specification [13]. And indeed, it has been demonstrated [25] that Kademlia, and thus aggregated greedy search, works well against an active adversary in networks of up to 50,000 nodes. Unfortunately, we believe that these results can be explained by the overwhelming redundancy employed for a relatively small network. This is because the following theorem shows that aggregated greedy search, on its own, does not scale.

THEOREM 2. *Let α be an upper bound to the number of nodes queried in every round, β the maximum number of neighbors accepted from any one queried node, and f the fraction of corrupted nodes, a constant. Then, as long as $(\alpha\beta)^{O(\log\alpha)} \subseteq o(n)$, there is an attack that makes the success probability of the search approach 0 in the limit. For example, this holds when $\alpha, \beta \in O(\log n)$.*

PROOF. The attacker proceeds by returning as many different corrupt nodes as possible, in the order of proximity to the search goal x, until only corrupted nodes are queried. If this attack succeeds, the attacker then has complete control over the course of the search.

Let $B_k = \{b_1, \ldots, b_{|B_k|}\}$ be the set of corrupt nodes that the searching node v knows after k rounds, and say that the b_i are ordered by

[2]We use *peer* and *node* as synonyms in this paper.

proximity to x, that is, $\forall 1 \leq i \leq j \leq |B_k|,\ \ d(x,b_i) \leq d(x,b_j)$. Notice that the attacker, by returning the collaborators closest to x first, can make sure that this order is static during the whole search.

Let us first find an upper bound to the number of rounds k required such that $|B_k| > \alpha$ with high probability. Of course, this presupposes that fn, the number of colluded nodes, exceeds α in the first place, a trivial side condition that is guaranteed in the long run by the assumption $(\alpha\beta)^{O(\log \alpha)} \subseteq o(n)$.

W.h.p., at least one corrupted node is queried within the first round, if either α or β is in $\omega(1)$. Let us assume this for the moment. Moreover, without loss of generality $\beta \geq 2$, because for $\beta = 1$, aggregated greedy search degenerates to simple greedy search with redundancy α, and Theorem 1 can be applied to yield the claim, because $(\alpha\beta)^{O(\log \alpha)} \subseteq o(n)$ implies $\alpha \notin n^{\Omega(1)}$.

Let us further assume that in this phase, every colluded node that v learns of will be queried, since v does not yet encounter any honest nodes closer to x than b_α. We will justify this in short. It is then easy to see by induction that in each following round, $|B_k|$ at least doubles, because every corrupted node will return at least two more hitherto unknown corrupted nodes. Thus, $|B_k| > \alpha$ for some $k \in O(\log \alpha)$ with high probability. Of course, this is a very weak bound, but it will suffice for our needs here.

During these k rounds, v will learn of at most $(\alpha\beta)^{O(\log \alpha)}$ nodes altogether. Assuming randomly distributed nodes (both honest and corrupted), the size of the set Y of honest nodes closer to x than b_α is Pascal distributed with parameters α and f, and the expected value is $\mathbf{E}(|Y|) = \alpha(1/f - 1)$. Each of these nodes has probability $(\alpha\beta)^{O(\log \alpha)}/n$ of being known to v before the attacker can make v query only corrupt nodes after k rounds. As in Theorem 1, we define indicator variables I_y that indicate this event for every $y \in Y$. Then, the expected total number of nodes in Y that v learns of in time k is

$$\mathbf{E}(\sum_{y\in Y} I_y) = \mathbf{E}(\mathbf{E}(\sum_{y\in Y} I_y)) = \mathbf{E}(\sum_{y\in Y} \mathbf{E}(I_y)) = \mathbf{E}(|Y|)\mathbf{E}(I_y).$$

Employing the Markov inequality just like in Theorem 1 shows that this is an upper bound to the probability that v gets to know any node that is closer than b_α before the attacker can make him query only corrupt nodes:

$$P[\sum_{y\in Y} I_y \geq 1] \leq \mathbf{E}(|Y|)\mathbf{E}(I_y) \in \frac{\alpha(1/f-1)(\alpha\beta)^{O(\log \alpha)}}{n} \subseteq o(1)$$

when $(\alpha\beta)^{O(\log \alpha)} \subseteq o(n)$. This proves the first claim in the non-constant case.

In the case that both α and β are constant, getting to know at least one corrupted node with high probability might take a little longer, say $\log(\log n)$ rounds. Asymptotically this dominates the remaining $O(\log \alpha)$ rounds to make $|B_k| > \alpha$. With the same reasoning as above, the success probability is now at most $(\alpha\beta)^{O(\log\log n)}/n$, but for $\alpha, \beta \in O(1)$, this is also $o(1)$.

Finally, when $\alpha, \beta \in O(\log n)$, the success probability is

$$\frac{(\alpha\beta)^{O(\log \alpha)}}{n} = \frac{(\log n)^{O(\log(\log n))}}{n} = \frac{e^{O(\log^2(\log n))}}{n} \subseteq o(1).$$

$\square$

The theorem suggests that for all realistic choices of search parameters, a rather simple eclipse attack defeats aggregated greedy search (and thus, Kademlia) in the limit. Notice, however, that this attack is based on the malicious nodes knowing x even in the first round. In order to overcome the problem, we propose to hide the search value. This protection strategy will be discussed in the next section.

5.2 Hiding the Search Value

We modify our search as follows. As before, in each round v chooses the α known nodes closest to x. From each of these nodes, instead of asking for x, v requests their whole finger table (FT).

Let $\alpha = \log_2(n)$ from now on. This value maximizes redundancy, yet might still be tuned in real applications to avoid excessive network load. In the first step, v queries all peers in his finger table. Each of the retrieved finger tables contains $\log_2(n)$ entries. These are all aggregated, and the best (closest to the searched ID) $\log_2(n)$ are selected for the next iteration. Only hitherto unqueried nodes are requested to provide their finger tables. The search continues until the top list of $\log_2(n)$ closest peers is not modified at the end of an iteration. The closest peer is then returned as the result of the search.

We choose to retrieve the whole FT because of the following reasons: First, we get extra redundancy while executing the lookup; Second, the queried node does not know which ID v is interested in. This keeps the adversary from responding with targeted malicious nodes, which are close to the searched ID.

Figure 1 shows the simulation results for aggregated greedy search while trying to hide the searched value. All the plots include 95% confidence intervals in order to show the solidity of the results. Malicious nodes provide only other malicious nodes in their FTs. Since the searched ID is not known, malicious nodes deliver random colluding nodes. Surprisingly, the rate of found malicious nodes seems to approach $2f - f^2$. This phenomenon might intuitively be explained as follows: Since the first round is unbiased, we can expect a rate of f colluding nodes to be queried. However, when we assume that all these nodes reply with malicious nodes only, while the honest nodes' finger tables still have attacker ratio f, the new attacker ratio in the replies from round 1 is expected to be $f \cdot 1 + (1-f)f = 2f - f^2$. Notice that from the second round on, all queried nodes have an ID bias towards x. While the attacker nodes still answer with random colluded nodes, the honest nodes' finger tables, due to the exponentially increasing finger distances in Chord (see Section 5.3 for more details), contain more nodes in their own vicinity, and thus in the vicinity of x. Thus, the attacker quickly loses its advantage in the course of the search process and may not be able to increase the attacker ratio from the second stage on. This explanation does not yet give a full formal model for the search process, still it might help account for the results we have seen.

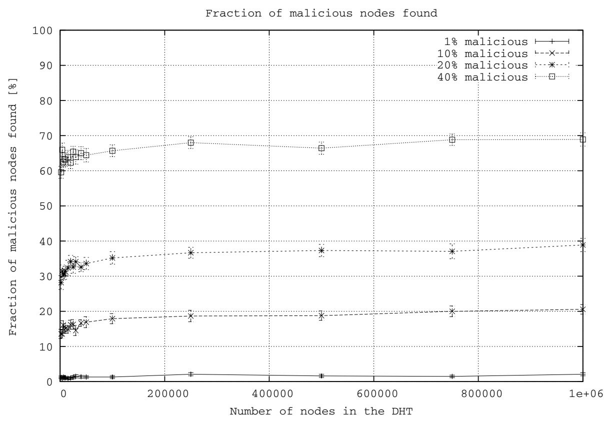

Figure 1: Malicious nodes respond with random collaborating nodes

While these results are a step in the right direction, they are far from ideal on their own. Moreover, there is an even more efficient attack against this search strategy. Due to the fact that the search in a DHT in the follow-up iterations depends on the results in the previous step, colluding nodes can combine their knowledge in order to find out the searched-for ID. Because the Chord ring is directed, the searcher will not ask for IDs laying "behind" the search value x. Thus, the attacker can estimate the interval of the ID by looking at which colluding nodes the searcher knows of (they were communicated by colluding nodes to the searcher in the previous iteration) but does *not* query. It can then return malicious nodes which are as close as possible to the searched-for ID (i.e. malicious nodes within the estimated interval of the ID). Figure 2 shows the simulation results for the case when the malicious nodes estimate the queried ID. Note, that this is only possible after the second search iteration since the first round does not leak information about the direction of search. We see that even having only 10% of malicious nodes in the system leads to more that 50% attacker success.

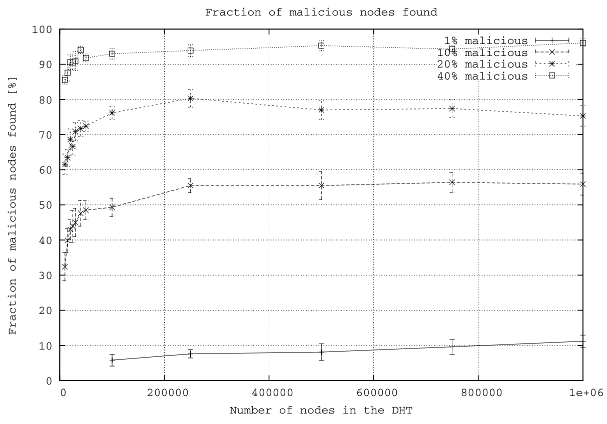

Figure 2: Malicious nodes respond with collaborating nodes closest to the search value

Thus, even aggregated greedy search without explicitly telling the search value does not offer enough protection: an adversary can still learn x, eclipse the searcher and guide him into a cluster consisting of malicious nodes only. Therefore, we need one additional building block.

5.3 Bounds Checking in Finger Tables

The success of the active attack described in the previous section is based on the fact that colluding nodes can provide arbitrary other nodes in their FTs. In order to mitigate this, we utilize properties of DHTs with a deterministic choice of peers in the FT. Chord is one such DHT. In Chord, the ith finger of a node with ID m is supposed to point to the node whose ID is closest to $m+2^{i-1}$. Since v knows the IDs of the nodes it queries, it can calculate these values and compare them to the actual finger table values in the responses it receives.

Since we already retrieve and have the whole FT, the check can be performed "for free" – without transmitting any additional information. In contrast to earlier approaches [17, 5], we do not only check the final result of the query, but all the intermediate steps. As our evaluation shows, this significantly improves the success probability.

We propose to perform bounds checking in finger tables as follows. Each peer calculates means of the distance between the actual IDs in its FT and optimal IDs (as if all IDs would exist). Let

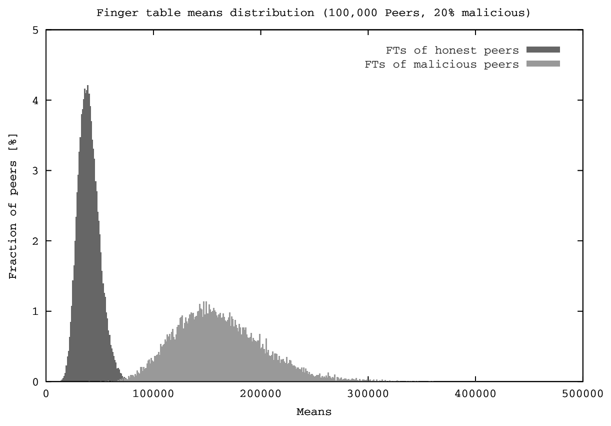

Figure 3: Finger table analysis: malicious nodes change 10 entries

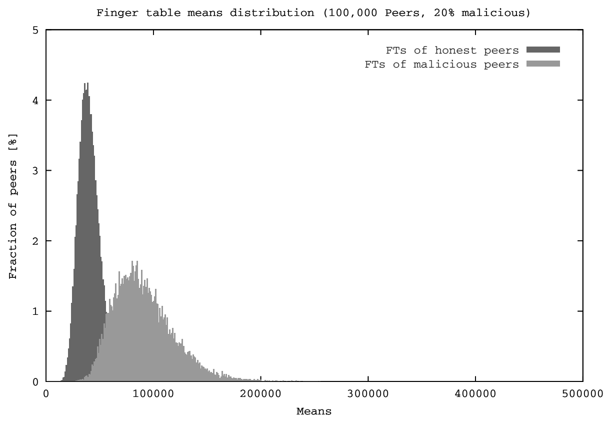

Figure 4: Finger table analysis: malicious nodes change 4 entries

us denote this as a *mean distance*. The mean distance is further multiplied with a factor – we call it *FT tolerance factor*.

The search is now modified as follows: in each iteration, FTs are only accepted and considered for finding the $\log_2(n)$ nodes closest to x, if they pass the FT test. The test yields a positive result if and only if the mean distance of the considered FT is smaller than the average sampled mean distance of the own FT times the tolerance factor.

Figures 3 and 4 show differences in the mean distances of honest and colluding nodes' FTs. While Figure 3 shows the case where malicious nodes change 10 honest node entries to malicious one, in Figure 4 only 4 entries are changed. Malicious nodes change their entries from the actual value to the malicious node closest to this value. This is the best strategy for them to make the FT still look plausible.

Since there are $100,000$ nodes in the considered scenario, on average there are 16 nodes in each finger table. We see that the mean distance is clearly distinguishable when many FT entries are changed, and becomes closer and less distinguishable when only a few entries are modified. Even though this finding is obvious, the real values give an intuition to which extent the FTs of malicious nodes can be changed while still successfully passing the FT check test.

Figure 5 shows the influence of the FT tolerance factor on false

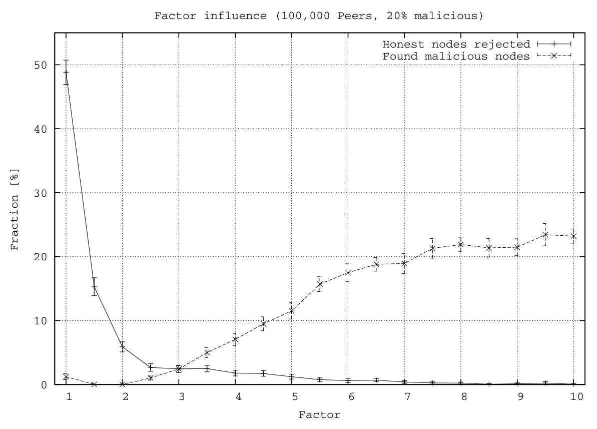

Figure 5: Factor influence on success probability (1)

positives and the rate of found malicious nodes, when the adversaries make their FT contain only colluded nodes by changing entries to the next colluded node where necessary. Even if the attacker is the correct owner of the searched ID, in case his FT check fails, his ID is thrown out resulting in a smaller fraction of malicious nodes found than present in the system.

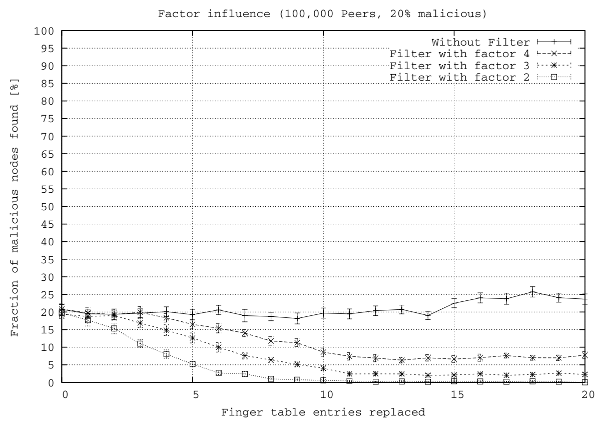

Figure 6: Factor influence on success probability (2)

In contrast to that, in Figure 6 malicious nodes detect the search directions and intelligently replace honest nodes with colluded ones, starting with the honest nodes closest to the searched area. This is done in such a way that the closest malicious node to the to-be-replaced honest node is selected for this operation. This increases the plausibility of modified FTs. The other entries remain unchanged. We can see the dependency of the attacker success rate on both the FT tolerance factor and the number of replaced entries. A staggering finding here is that replacing all the entries in the FT with the closest malicious nodes does not help the adversary to significantly increase the rate of its nodes in the final results of the queries, even if the searcher were to believe all these FTs. Thus, as long as the FTs look plausible, the attacker is not able to significantly bias the user selection, even if he provides the searcher with only malicious nodes. This provides a clue into why NISAN, or the combination of aggregated greedy search with finger table checking, provides such strong protection against eclipse attacks: by forcing the attacker to conform closely to the original structure of the DHT and aggregating results, we can always correct misleading information by taking into account FTs of close-by honest nodes. We conjecture that this combination could probably deter eclipse attacks even without hiding the search value, but we keep this feature because it does not cost us anything at this point and might be useful against information leakage attacks (see Section 6 for more details).

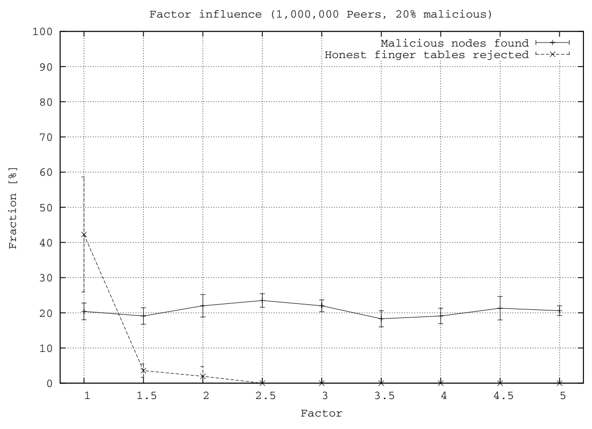

Figure 7: Malicious nodes know the FT acceptance threshold

In Figure 7 we additionally assume that the adversary knows the FT acceptance threshold of the users. As in the previous plot the attacker learns the search value. He replaces FT entries in the search direction, but only up to the acceptance threshold, so that his FT would be still accepted by an honest user. This is the strongest adversarial behavior that we tested, because it seems hard to conceive of an easy way for the attackers to optimize their responses beyond this point. The optimization problems arising are hard enough to describe, let alone solve efficiently. Certainly, there is ample space for future research in this direction. Because the structure of DHTs like Chord seems so benign to our approach, we conjecture though, that the effect of further optimization, at least when restricted to efficiently solvable problems, will be limited.

As our simulations show, bounds checking on FTs is a very promising technique. A FT tolerance factor of 3 seems to be a good choice for the considered setup. As Figure 7 shows, virtually no FTs of honest users are rejected when we apply this factor. Furthermore, malicious nodes are not able to increase their rate in the system while changing more than 3 entries (see Figure 6). Knowing the searched ID and the acceptance threshold of the honest users does not help malicious nodes to significantly increase their rate in the queries beyond their rate in the system as Figure 7 suggests. Castro et al. [5] give a very precise statistical method for optimizing the tolerance factor when checking final results. Their method could easily be adapted to our FT checking. Yet, we find that simply selecting an ad-hoc value such as 3 works pretty well in our simulations. Further inspection of Figure 6 helps us to understand this: We are relatively free to minimize the false positive rate by choosing a rather high tolerance factor, because there is no need to strictly minimize the false negatives. This is because as long as the fraction of malicious nodes found remains below the actual attacker rate f, it just makes no sense for an attacker to lie about its FT.

Performing the FT check test based on the mean distance only is certainly not the only possibility. Possibly, other strategies would yield even better results. In fact, the whole armament of statistical classification techniques can be imagined to be brought to use. However, the results show that even our simple strategy works well in the considered simulations. That is why we leave improvements in this direction for future work. See Section 6.3 for arguments why research in this direction might still be worthwhile. One possible improvement might be to consider only (or to weight additionally) those entries which are leading into the direction of search and are considered for the top list of $\log_2(n)$ best nodes seen so far. If an adversary is able to learn the search direction, these are the first entries he would replace.

As the results show, our protection works well against an active attacker. Most of the simulations provided here have been conducted with 20% of malicious nodes in the system. However, we have also conducted simulations for 1,10, and 40% of colluding nodes, respectively. Due to the space limit and similarity of the results we forego their presentation in this paper.

5.4 Further Improvements

We have analyzed the rate of the malicious nodes in the $\log_2(n)$ best nodes list depending on the iteration. Our initial idea was to also accept the nodes found in the intermediate steps of a search as onion router candidates in order to mitigate bridging and fingerprinting attacks. However, as the results in Figure 8 show, this is not a good idea. Still, the rate of colluded nodes in the last search iteration corresponds to the rate of malicious nodes if we only look at the single result of the search (cf. Figure 7). Thus, it makes sense to consider the whole list of $\log_2(n)$ closest nodes to the search value instead of the single value only as possible routers. This increases the uncertainty of an attacker about the nodes which are known to honest users (as their number is significantly increased), and thus helps us counter passive information leakage attacks, as explained in the following section.

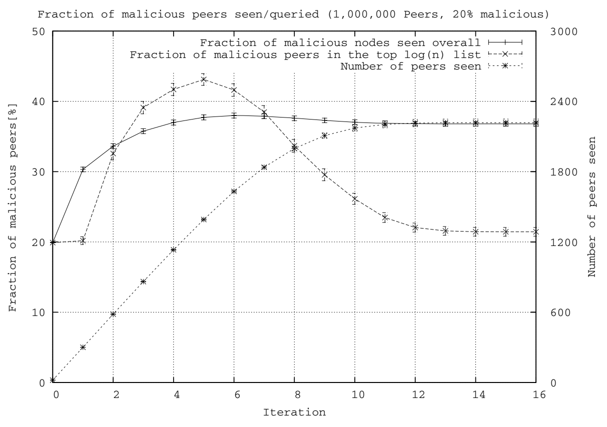

Figure 8: Fraction of malicious nodes depending on the iteration

6. DISCUSSION AND ALTERNATIVES

In this section, we discuss challenges to network information services that have rarely been seriously addressed when proposing new methods, but have to be considered when deploying a system in the real world. One of these is information leakage, which invites passive attacks such as bridging and fingerprinting.

The *fingerprinting* attack relies on the fact that an adversary, observing some tunnel through the network can associate it to a small set of possible initiators. *Bridging* an honest node assumes that, similarly as with the fingerprinting, the nodes constructing the paths only know a fraction of all routers. Thus, not all combinations of in-/output links of a router are valid: possibly no node knows all the routers needed to construct them. In order to deanonymize a user, the attacker has to control (or observe) at least the exit node involved in the user's tunnel. Observing or controlling the middle node can be used to further reduce the set of possible initiators. For a full deanonymization, however, the involved combination of nodes should be known to a single user only. In other cases this information merely reduces the anonymity set (the set of possible initiators), but does not necessarily give confidence in the conjecture.

Even though from our point of view bridging and fingerprinting attacks have been of rather theoretical interest so far and could not yet be shown to significantly compromise anonymity in open networks like Tor, where the number of users is estimated to be in the hundreds of thousands, we still find it challenging to look for defenses against these attacks. Because all approaches with partial knowledge about the network known so far are susceptible to them, we try to overcome these attacks by discussing a radically different alternative to NISAN, namely random walk, as well as a combination of both methods.

After evaluating these two approaches, in 6.2 we address the often overlooked question of bootstrapping. While keeping a rather generic view, we give implementation pointers and argue why we believe that NISAN can be bootstrapped and maintained securely when carefully implemented. Finally, in 6.3 a more powerful adversarial model is examined, as we look into NISAN's behavior under the assumption that the attacker is free to choose their positioning within the ID space of the DHT.

6.1 Information Leakage and Random Walks

All known DHT-based information distribution services, including NISAN, are endangered by passive information leakage attacks [15, 6]. These attacks generally use the fact that searching in a DHT entails talking to many colluded nodes in the process. The redundancy typically used to prevent active attacks only makes this exposure worse, so that we may even talk about a trade-off in protection against active and passive attacks [15]. Through these search queries, the attacker learns who is searching whom, either directly or through linking multiple queries. Measures such as recursive routing or hiding the search goal may make this information harder to obtain or less precise, yet in general cannot keep the adversary from gaining significant insight. This kind of information can then be abused in a number of subtle ways that are out of scope in this paper. In fact, the types of attacks possible have become a very active research topic lately [15, 6]. Still, in all of these attacks the worst case that can arise is the attacker gaining knowledge of the complete routing circuit. Though this is clearly not desirable, there is little research about the consequences in real-world systems. Arguably, we are worse off if the attacker *controls* the routing circuit, as is the goal of eclipse attacks. That is why so far, we have placed emphasis on avoiding this kind of attack.

The information leakage in NISAN lies in giving away x, or, more precisely, the link between the searching node v and x. Through hiding the search value (cf. 5.2) and taking into consideration the whole top list at the end of the search (cf. 5.4) we have already introduced some uncertainty for the attacker. In general, a good way to raise the entropy for the attacker in our choice of circuit nodes is to simply conduct multiple searches, sequentially or in parallel, thereby learning a greater part of the network. Moreover, such behavior, especially when conducted by a great many nodes, may serve to obscure the links between searchers and searched-for nodes.

Although these measures may foil most attacks in practice, and though it is not clear how much an attacker could profit at all from knowledge gained this way, it would obviously be preferable from a

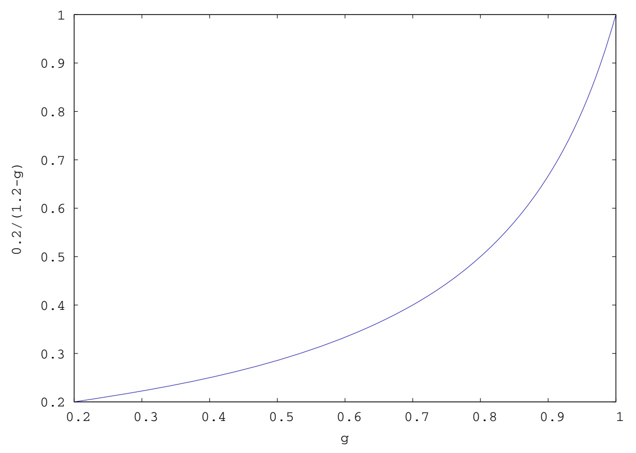

Figure 9: Theoretical lower bound to attacker success on random walks for attacker ratio $f = 0.2$ and increasing attacker finger table corruption g

theoretical point of view if we could provably avoid passive attacks altogether. Hence we looked for an entirely different approach that cannot give away a search goal since there is no such thing. A random walk (RW) through the network is a relatively obvious solution.

In a RW, we randomly select one of our neighbors, ask this neighbor for its finger table, and, again, randomly select one of its neighbors, iterating this method for a path length l. On average, l should at least be $\log_2(n)$, since this ensures that each peer in the DHT can be reached by the RW. Intuitively, there is little information leakage in this process, probably as little as we can reach when routing in a DHT, since there really is no direction to the search that might be leaked.

It is clear that this approach can be combined with our finger table checking to keep the attacker from presenting arbitrary neighbor tables and thus hijacking a path with certainty. Still, we have to assume a different ratio g of colluding nodes in attacker finger tables that is potentially higher than f, especially when we allow for arbitrary attacker positioning as discussed in Section 6.3. Let $p_f(l) = 1 - p_s(l)$ be the probability of selecting a colluded node after l steps. Obviously, $p_f(1) = f$. Moreover, for $l > 1$, $p_f(l) \geq gp_f(l-1) + fp_s(l-1)$. Solving this recurrence relation using geometric series yields $p_f(l) \geq f\frac{1-(g-f)^{l+1}}{1-g+f}$, which can easily be checked by using induction.

We observe that with growing path length, this probability rapidly becomes $\frac{f}{1-g+f}$. Figure 9 plots this predicted attacker success rate for increasing g with $f = 0.2$.

Unfortunately, this strong dependency on g turns out to be problematic in the real world, because it compounds with another property of RWs. Figure 10 displays the results of our simulations. Specifically, we look at the impact of the tolerance factor, when the attackers know this factor and try to modify as many fingers as possible without being detected. Unlike aggregated greedy search, where it is important for the attacker to change FT entries close to x, with RW, every FT entry is equally important and so it becomes a good adversarial strategy to adapt as many entries as possible, which translates into making small changes first. We can see that already with a factor of 3, the attackers can modify about 12 out of 16 FT entries on average, which, consistent with our prediction, translates to almost 50 percent attacker success for $f = 0.2$. The sweet spot seems to be at a lower factor here, but even at factor 2 we get a failure rate of about 0.35, with already significant false positives.

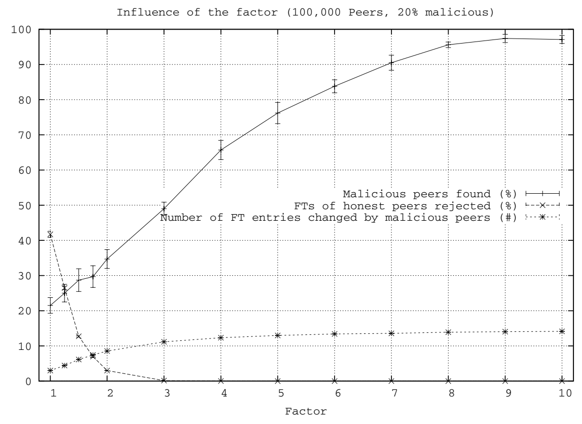

Figure 10: Tolerance factor influence with random walk

This shows that using RWs for network information is only advisable when the active attacker ratio is low and information leakage is a serious issue. On the other hand, the results highlight the effective protection against an active adversary that is provided by our improved aggregated greedy search scheme with finger table checking.

We have been thinking of a combined approach that might bring together the advantages of both aggregated greedy search (security against active attacks) and RWs (security against passive attacks). Unfortunately, the straightforward idea of searching for a few rounds and then randomly switching the goal x seems to have limited applicability, as Figure 11 suggests. It makes clear that during the search process, the ratio of colluding nodes in the total set of nodes surveyed, as well as in the closest nodes top list is rising rather quickly in the first few steps, a finding echoed in Figure 8. It is mitigated only later on, when the search converges towards x. By this time, however, the attacker already has a pretty good idea of the search direction.

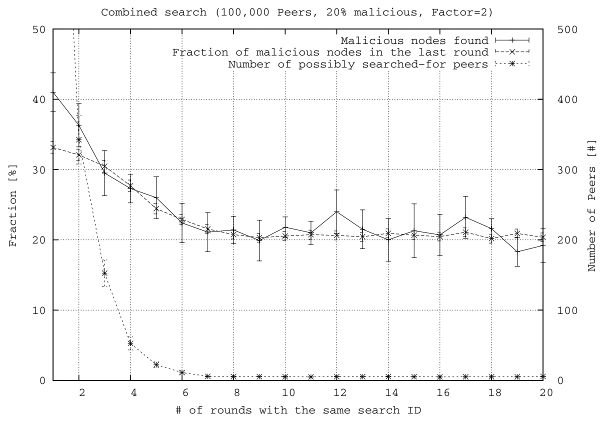

Figure 11: Combined: random walk and search

6.2 Bootstrapping Process

So far in our analysis we have assumed a correct bootstrapping of the network. Under this assumption we have shown that our approach NISAN is able to provide an adequate protection to its users: the fraction of malicious nodes found in random look-ups is not significantly larger than the overall fraction of malicious nodes in the system. Additionally, sampling a significant fraction of the network considerably hardens bridging and fingerprinting attacks. In this section we discuss how the users can overcome the problem of malicious nodes while joining the network, i.e. bootstrapping.

We assume that before joining the network a user knows a few DHT members and at least one of them is not colluding. This assumption is meaningful since it is unlikely that any approach would work if only malicious nodes are known to the user. The user generates its ID (which might be a hash of its DHT public key, say), and asks the known DHT members to execute the bootstrapping for this ID. Each of these nodes executes the lookups (in the way we proposed before) for the entries in the new nodes' FT and communicate them to the new node. The new node selects the entries closest to the optimum values. Notice that even a majority of evil nodes could not break this process, as long as there is one honest node whose searches succeed. After the stabilization protocol run [16] the new node is a regular member of the network. By basing bootstrapping (and maintenance, which can be conducted in a similar manner) on our secure routing primitive, we are confident not to introduce additional security hazards.

6.3 Arbitrary Positioning of Malicious Nodes

So far we have considered the case where malicious nodes are uniformly distributed along the ID space. It is beyond the scope of this paper to discuss the realism of this assumption and possible measures for enforcing it. Still, we briefly look at a stronger adversarial scenario: what if the colluded nodes could arbitrarily position themselves within the whole ID space of the Chord ring? Clearly, if this were to work instantly, or the DHT remained very stable for a long time, the adversaries could eclipse a single user if they knew his ID. We consider the case where they do not have a concrete victim but are rather interested to be in as many paths in the system as possible, thus trying to get as much information as possible about the whole system.

At this point, the asymmetry of our DHT distance metric comes into play: It assures that, typically, a (colluded) node is not a neighbor of its neighbors. Thus, it is a nontrivial feat for the adversary to construct positions for its nodes such that their FTs may contain many of their own while still being plausible. Again, a presumably hard problem arises for the attacker, and we can only give a simple solution that we believe close to optimal, without being able to prove this conjecture.

From our point of view, the so-called *bisection* would be a very good attacker positioning strategy in this case: recursive division of the ID space into two equal parts (halves) and placing the malicious nodes on the dividing points. This would lead to "perfect" FTs in the sense that the mean distance would rapidly approach zero with an increasing number of malicious nodes. Figure 12 shows the simulation results for this scenario. Even though the results are worse than in the regular case when the arbitrary positioning is not possible, the rate of found colluding nodes in random searches is still fixed with increasing network size, while non-linearly dependent on the total attacker rate. Therefore, when the attackers are able to mount this kind of attack, we can say that NISAN – as it is – still scales, yet is more vulnerable to high attacker rates.

However, having perfect finger tables is also conspicuous and can be detected by the honest users. Note that malicious nodes cannot easily improve their positioning by being only close to the optimal value – this would work only for a few of them. The rest would have "regular" FTs.

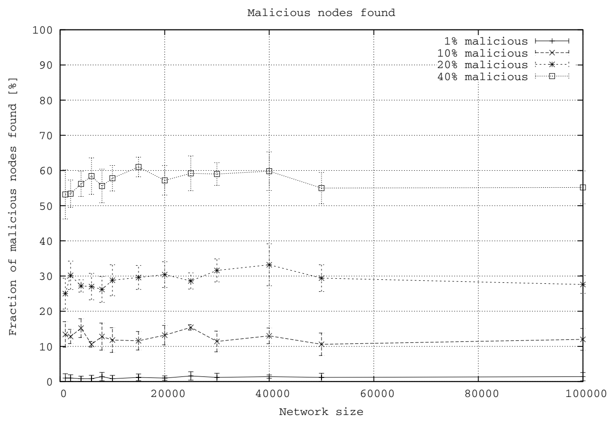

Figure 12: Arbitrary positioning of malicious nodes

It is not hard to come up with slight modifications to our DHT scheme that foil the simple bisection attacker plan. For example, we might think of requiring the ith FT entry of node m to be strictly greater than $m+2^{i-1}$, instead of just greater or equal. This already breaks symmetry. However, it seems much harder to come up with a DHT structure that can actually be shown to allow for no or not much advantage through any chosen positioning. We have to leave this fascinating research topic for future work, yet remark that it is a standard problem in security research that unknown attacks can never be ruled out in general. In light of our findings, we therefore propose NISAN as the most secure and scalable approach to truly distributed network information distribution that we know of; and while we cannot guarantee the absence of unthought-of attacks, the above considerations inspire us with confidence that NISAN will defend against them gracefully, or at the very least can be adapted to do so easily.

7. CONCLUSIONS

Although peer discovery and anonymization are two disjoint tasks, the network information distribution has a direct impact on the anonymization. Therefore, during the design of new anonymization systems the network information distribution has to be addressed as well.

In this paper we proposed a DHT-based practical approach for distribution of network information. Our scheme prevents malicious nodes from biasing the node look-ups, while requiring each node to know only a small subset of the network. The approach is highly scalable and does not require to trust any third party.

Just like every other known approach that does not lead to a full network view, our approach is still susceptible to bridging and fingerprinting attacks. Even though their practical seriousness and impact are under research, we hardened NISAN against these attacks by hiding the searched goal and learning a significantly large part of the network. If information leakage is intolerable in a given scenario, we propose the alternative approach of random walks, while acknowledging that this method does not feature equally strong protection against active attacks. In practice, a more typical way of dealing with the problem would be to adjust the number of searches before actually selecting a router. This way, a greater part of the network is known over time, making fingerprinting-type inference significantly harder for the attacker. In the limit, this poses the question if gossiping-like alternatives that lead to discovery of the entire network can be made scalable, and adversarial exploitation of a full network view as in intersection attacks can be prevented.

These alternatives have to be further researched in order to find the most appropriate solution for the addressed problem.

8. REFERENCES

[1] B. Awerbuch and C. Scheideler. Towards a scalable and robust DHT. In *SPAA '06: Proceedings of the eighteenth annual ACM symposium on parallelism in algorithms and architectures*, pages 318–327, New York, NY, USA, 2006. ACM.

[2] I. Baumgart and S. Mies. S/Kademlia: A practicable approach towards secure key-based routing. In *Proceedings of the 13th International Conference on Parallel and Distributed Systems (ICPADS '07), Hsinchu, Taiwan*, volume 2, Dec. 2007.

[3] O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 115–129. Springer-Verlag, LNCS 2009, Jul 2000.

[4] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz. Denial of service or denial of security? In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 92–102, New York, NY, USA, 2007. ACM.

[5] M. Castro, P. Druschel, A. J. Ganesh, A. I. T. Rowstron, and D. S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *Symposium on Operating Systems Design and Implementation (OSDI 02)*, Boston, MA, USA, December 2002.

[6] G. Danezis and P. Syverson. Bridging and fingerprinting: Epistemic attacks on route selection. In N. Borisov and I. Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 151–166, Leuven, Belgium, July 2008. Springer.

[7] R. Dingledine and N. Mathewson. Tor Directory Protocol Specification. https://www.torproject.org/svn/trunk/doc/spec/dir-spec.txt.

[8] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, 2004.

[9] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, Washington, DC, November 2002.

[10] A. Kapadia and N. Triandopoulos. Halo: High-assurance locate for distributed hash tables. In *NDSS*. The Internet Society, 2008.

[11] C. Lesniewski-Laas. A sybil-proof one-hop DHT. In *Workshop on Social Network Systems*, Glasgow, Scotland, April 2008.

[12] G. S. Manku, M. Naor, and U. Wieder. Know thy neighbor's neighbor: the power of lookahead in randomized p2p networks. In *In Proceedings of the 36th ACM Symposium on Theory of Computing (STOC)*, pages 54–63, 2004.

[13] P. Maymounkov and D. Mazieres. *Kademlia: A Peer-to-Peer Information System Based on the XOR Metric*. 2002.

[14] A. Mislove, G. Oberoi, A. Post, C. Reis, and P. Druschel. AP3: Cooperative, decentralized anonymous communication. In *In Proc. of SIGOPS European Workshop*, 2004.

[15] P. Mittal and N. Borisov. Information leaks in structured peer-to-peer anonymous communication systems. In P. Syverson, S. Jha, and X. Zhang, editors, *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS 2008)*, pages 267–278, Alexandria, Virginia, USA, October 2008. ACM Press.

[16] R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *ACM SIGCOMM 2001*, San Diego, CA, September 2001.

[17] A. Nambiar and M. Wright. Salsa: A structured approach to large-scale anonymity. In *Proceedings of ACM CCS 2006*, Alexandria, VA, USA, October 2006. ACM Press.

[18] P. Palfrader. Number of running tor routers. http://www.noreply.org/tor-running-routers/.

[19] A. Panchenko, B. Westermann, L. Pimenidis, and C. Andersson. Shalon: Lightweight anonymization based on open standards. In *Proceedings of the 19th International Conference on Computer Communications and Networks (IEEE ICCCN 2009)*, San Francisco, CA, USA, August 2009. IEEE Press.

[20] M. Rennhard and B. Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2002)*, Washington, DC, USA, November 2002.

[21] A. Singh, T.-W. Ngan, P. Druschel, and D. S. Wallach. Eclipse attacks on overlay networks: Threats and defenses. In *INFOCOM*. IEEE, 2006.

[22] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 44–54, Oakland, California, USA, 1997. IEEE Computer Society.

[23] P. Tabriz and N. Borisov. Breaking the collusion detection mechanism of morphmix. In G. Danezis and P. Golle, editors, *Proceedings of the Sixth Workshop on Privacy Enhancing Technologies (PET 2006)*, pages 368–384, Cambridge, UK, June 2006. Springer.

[24] Tor Documentation. https://www.torproject.org/documentation.html.

[25] B. Westermann, A. Panchenko, and L. Pimenidis. A kademlia-based node lookup system for anonymization networks. In *Advances in Information Security and Assurance: Proceedings of the Third International Conference on Information Security and Assurance (ISA 2009)*, volume 5576 of *LNCS*, pages 179–189, Seoul, South Korea, Jun 2009. Springer.