A Kademlia-based Node Lookup System for Anonymization Networks

Benedikt Westermann[1], Andriy Panchenko[2], and Lexi Pimenidis[3]

[1] Center for Quantifiable Quality of Service in Communication Systems*
NTNU, 7491 Trondheim, Norway
westermann@q2s.ntnu.no
[2] Computer Science Department, Informatik IV,
RWTH Aachen University, D-52074 Aachen, Germany
panchenko@cs.rwth-aachen.de
[3] Chair for IT Security
University of Siegen, Siegen, Germany
pimenidis@fb5.uni-siegen.de

Abstract. Node lookup mechanisms constitute an integral part of any overlay network, and hence also of anonymous communication networks. Today, most anonymizers use centralized directories, which leads to scalability problems in the long run. Additionally they require the user to trust the directory provider.
In this paper we revisit the concept of distributed hash tables to address these issues. We propose a scalable node lookup system based on Kademlia and show how it notably hardens the eclipse attack and node fingerprinting. Additionally we provide comparative scalability analyses for our approach and Tor's directory protocol.

1 Introduction

Anonymous communication techniques are a fundamental building block for privacy-friendly web browsing as well as privacy-aware identity management, eGovernment, eCommerce and eHealth technologies. While cryptography can protect the integrity and confidentiality of the data part of the packets, everyone along a route can still observe the addresses of the communicating parties. Anonymous communication deals with hiding relationships between communicating parties.

Currently, the most popular and widespread anonymous communication network is Tor [1]. The Tor network itself is a circuit switched, low-latency anonymization network which targets on providing protection at the network layer against a non-global adversary. Currently, the number of Tor servers is about two thousand[4] [2], whereas the number of users is estimated to be hundreds of thousands.

* *"Center for Quantifiable Quality of Service in Communication Systems, Center of Excellence" appointed by The Research Council of Norway, funded by the Research Council, NTNU and UNINETT.* http://www.q2s.ntnu.no

[4] as in February 2009.

On the one hand this large user base is seen as one of Tor's strengths, since the degree of anonymity is usually linked to the number of active users. On the other hand this is a problem at the same time for the directory: in Tor every user knows about every router. Clearly, this limits scalability.

In this paper, we address the problem of distributing information about the identity of anonymizing networks' servers to its clients by revisiting the distribution via a distributed hash table. Similar to most low-latency anonymization networks, which are build to withstand a local (active) attacker, this work will focus on the same attacker model.

2 Related Work

Most deployed anonymous communication networks, e.g. AN.ON/JAP [3] and Tor [1], use a centralized directory service. They require each user to know all nodes in the network. This has several advantages, among others: users are able to make their decision based on the same knowledge base; also a number of attacks are made more difficult [4].

However, approaches enforcing users to know every node in the network entail scalability problems. Moreover, the usage of central directories requires users to trust the directory provider. In Tor, an onion proxy[5] (OP) creates circuits by selecting three suitable onion routers[6] (ORs) from a list of all currently available ORs, the *directory*. To this end, certain trusted Tor nodes provide the *directory service* by serving signed documents containing information about the available ORs. Such a network status document contains *router descriptors* of all currently known ORs, including meta information describing their current status. Due to scalability issues stemming from an increased growth of the Tor network, there have been several changes in the directory protocol since its initial release. Today, Tor is already using the third version of the directory protocol. The authors admit in the design document that requiring each Tor node to know all about all other Tor nodes is maybe not a viable solution in the long run [5].

In the first version of Tarzan its authors [6, 7] propose the use of a *DHT* to distribute network information. Due to security problems with respect to their attacker model this approach was later replaced by a *gossiping* protocol. In their original approach, each peer is represented by a key which is the cryptographic hash of its IP address. By searching for a random lookup key, a peer can discover a random host's IP address together with its public key. However, the authors have not proposed any mechanism to check the integrity of a reply. In its second version, similar to Tor, Tarzan also requires all nodes to know about all other nodes. To achieve this, the authors proposed a simple *gossiping* protocol, which is described on a high level only.

MorphMix [8] requires each user to possess information only about a limited amount of other users, even during the operational phase of the MorphMix protocol. For the circuit setup so-called *witness nodes* are used to facilitate the

[5] The client-side.
[6] The server-side.

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 facilitate possible security issues arising from this feature, MorphMix uses a collusion detection mechanism to detect malicious nodes that misbehave by offering other colluded nodes for traversal. The detection bases on the hypothesis that colluding nodes have a different behavioral pattern and that this can be pinpointed on the long run. However, this protection scheme has been shown to be broken [9].

Salsa [10] is a DHT which was specially developed for anonymization networks. The identities are based on hashes of the nodes' IP addresses which are organized in a tree structure. Redundancy and bound 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 has shown [11] that even in this case Salsa is less secure than previously thought: if the number of corrupt nodes is below 20%, still more than a quarter of all circuits are compromised because of the information leak.

3 Attacker Model

In this section we describe the assumptions on our 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. We also assume that a client knows at least one semi-trustworthy[7] entry point in the network which is not corrupt. Please note, that this is a weaker assumption than the assumption of a single semi-trusted third party *everybody* needs to trust. The latter is an assumption of Tor and AN.ON respectively.

4 Protocol Description

The scalability problems regarding centralized approaches for network information distribution constitutes an incentive to study decentralized alternatives. Therefore, we have applied a *distributed hash table* (DHT) to implement a network information service. A DHT has a set of attractive features: distributed data storage, scalability, load balancing, and fault tolerance. In our approach we have used the Kademlia DHT [12] which is based on the XOR-metric[8].

[7] I.e. trusted with respect to node lookups.

[8] The larger the XOR of two identities is the more far away are the identities from each other.

In Kademlia, objects and nodes are represented by 160-bit identities (IDs). Below, we use the term *nodeID* to refer to an ID that represents a node, while *objectID* denotes an ID that refers to an object. Every node provides four remote procedures callable by other nodes: *findnode*, *findvalue*, *ping* and *store*[12].

Nodes in Kademlia are found by a local procedure called *nodelookup*. A nodelookup returns the k closest nodes w.r.t. a XOR-metric and a given 160-bit word. The parameter k is a global parameter in the network. A node picks the k closest locally known nodes to start the nodelookup. From this selected set the node picks α different nodes on which it calls the *findnode* procedure. Every call returns again the k closest nodes known by the queried node. Similarly, in the next iteration, a set of α from the k closest nodes, which have not been contacted yet, will be queried. This repeats until the closest node remains the same. In the following step all k closest nodes not queried yet will be queried. Finally, the k closest nodes will be returned.

In our approach, an object in the DHT represents a detailed description of a network node. This object is called a *descriptor*. It contains not only mandatory contact information (IP address, port), but also a public key as well as a signature to prevent malicious modification of the descriptor during its transfer. The descriptor can also be equipped with additional information. The objectID of a descriptor equals the nodeID of the network node described by the descriptor. All servers have a unique nodeID, which equals a cryptographic fingerprint of their public key. Hence it is legitimate to assume that nodeIDs are uniformly distributed in the 160-bit space.

Only servers which provide the anonymization service are members in the DHT. Clients are not registered as members, and thus end-users cannot be found by issuing a nodelookup[9]. To execute queries, a user maintains encrypted connections to a few (at least one) semi-trusted servers it knows and forwards all requests to these servers. The servers execute the queries and then send the results back to the user. In the remainder of the text we implicitly assume this behavior for clients, whereas servers directly execute their queries. This procedure aims to harden fingerprinting attacks [13]: Due to the encryption a local attacker observing the link between the node and client can not gather information about the client's local directory.

It is not required that users must know about all nodes in the network (i.e., lookup all descriptors). To build geographically widespread circuits containing nodes from the whole network we randomly search for nodes. This can be done by executing a nodelookup for a random 160-bit word. The result of this nodelookup is the set of nodeIDs which are closest to the generated word. The procedure *findvalue* issued on one of these nodeIDs will return the corresponding descriptor, since each node stores its own descriptor under his nodeID. This facilitates the search for nodes.

The results of a sequence of searches will be stored in a local database on the client's side. This information will later be used to build up a connection over

[9] This makes it hard to enumerate all users of the system.

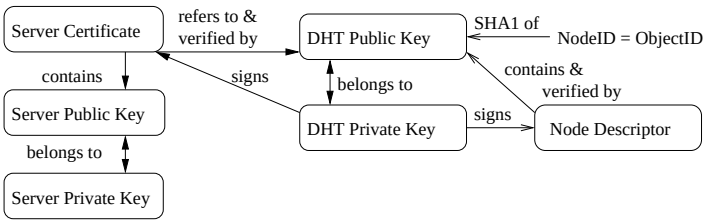

Fig. 1. Relationship between DHT Keys and Server Certificates

a subset of servers in the directory. This caching prevents an attacker to gather information on the initiator of a path due to time correlations.

As mentioned above the descriptor is a signed document. The public key used to verify the signed descriptor is shipped together within the signed descriptor. On the first glance this does not help to protect against malicious modification, since a attacker can simply generate a valid signature by replacing the original public key and signature with its own. However, this attack as well as a set of other attacks (like MitM) are prevented due to the tight connection (see Figure 1) between nodeIDs, objectIDs and the used keys in the certificates.

In our design, the nodeID is equal to the SHA1 hash of the DHT public key of the node. This limits the attacker's ability to freely choose its position in the DHT. Additionally, this also prevents an attacker from placing the same descriptor various times under different IDs. Lastly, this also hinders him from updating the DHT with erroneous descriptors for already existing honest nodes[10].

The private key used for signing the descriptor (the DHT key) is also used for signing the server certificate which is used during the establishment of an encrypted connection (e.g., TLS encryption) to the anonymizing node. This ensures a one-to-one mapping between a descriptor and the corresponding server certificate (c.f. Figure 1). The corresponding TLS public key is stored within the server certificate. By verifying the server certificate with the public key of the descriptor (DHT public key), the client can check if the server is the one referenced within the descriptor.

5 Security Analysis

One of the most serious threats against distributed hash tables are eclipse attacks [14]. These attacks aim to "eclipse" parts of the network or the information within it. In our setup a successful eclipse attack enables an adversary to prevent a node to find honest nodes. This results in a higher percentage of malicious nodes being present within the local lists of the honest nodes than in the whole network. Whereby the attacker increases his probability to identify a user in an anonymization process.

[10] Note that every server stores its own descriptor because the nodeID equals the objectID.

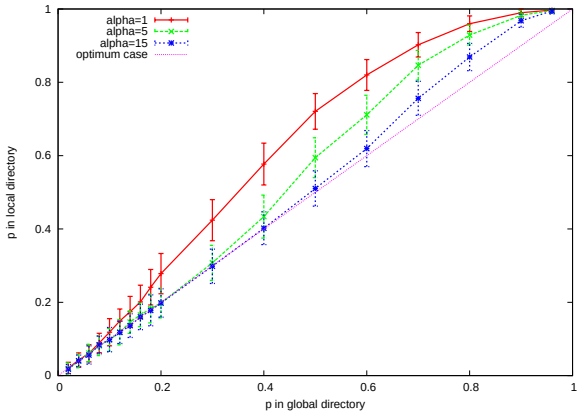

Fig. 2. Fraction of malicious nodes depending on alpha ($n = 5000, k = 20$)

We assume that a good[11] strategy to reach the attacker's objective is, to be uniformly distributed over the whole key space. To this end the attacker generates nodeIDs as a normal honest user would do. We also assume that only the attacker knows the identity of the malicious nodes within the DHT. Moreover the attacker returns only malicious nodes whenever a *findnode* request on one of its nodes is triggered. This increases his chances to be in the client's local list.

In our simulation we use a simplified Kademlia protocol. Therefore our simulation does not cover optimizations or caching functions [12]. We simulated directory lookups as follows: in one run we want to find 100 disjoint descriptors. In order to do this, at least 100 searches are required. For each search we select a random honest node that executes the search for one random ID.

In order to reduce the results' variance, each run (finding 100 disjoint random IDs) was repeated 100 times. Figure 2 presents the influence of α – lookup redundancy factor – in a DHT of $n = 5000$ nodes ($k = 20$). The x-axis shows the fraction of malicious nodes in the DHT. The y-axis plots the fraction of malicious nodes in the client's found list. Figure 3 depicts the influence of an increasing number of users on the result.

Both figures show that the attacker's chances increase if either the total number of nodes grow (and the attackers fraction remains constant) or the attacker increases the amount of malicious nodes. It is also possible to see that α can be used to decrease the influence of the attacker.

A second dangerous attack on clients in a DHT is fingerprinting [13]. One possibility to collect information about the client's view is to observe the queries in the DHT. If this holds, the attacker is able to partially reconstruct the victim's local directory and use this knowledge to reduce the anonymity set of the users. We propose that clients do not execute requests directly on their own, but rather forward encrypted requests to at least one semi-trusted server. Therewith the

[11] We do not claim this is an optimal strategy.

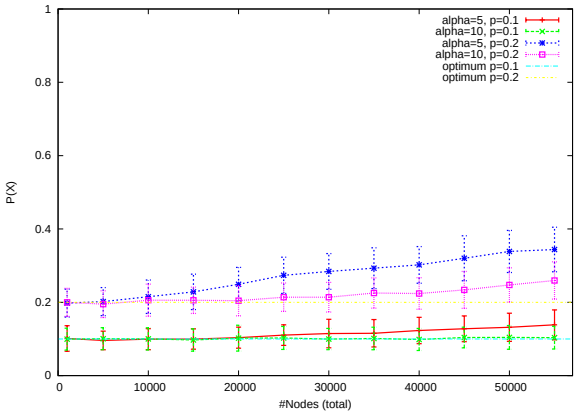

Fig. 3. Fraction of malicious nodes depending on n ($k = 20$)

attacker is not able to gather information about the content of a client's directory by eavesdropping the connection between the client and semi-trusted server. Moreover, since we are not dealing with a global attacker, colluding DHT nodes have no guarantee to observe every request (originating from the semi-trusted hosts). Since several users use the same nodes for their lookups, the uncertainty about client's directory is further increased.

Please note that this procedure neither protects against denial of service attacks nor Sybil attacks. However, to the best of our knowledge, Sybil attacks [15], as well as (distributed) denial of service attacks, constitute unsolved problems for most of the other approaches to network anonymity, too.

6 Scalability Analysis

In Kademlia nodes present leafs in a binary tree. We assume the height of this binary tree to be in $\mathcal{O}(\log(n))$, where n is the number of nodes. This is reasonable due to our assumption of the uniform distribution of the IDs. The number of steps for a nodelookup in Kademlia is $h - \log(k)$[12], where h is the height of the binary tree. Since we call α many nodes per step which each return k nodes, a nodelookup is bound by a function of the order $\mathcal{O}(\alpha \cdot k \cdot \log(n))$.

To find a random descriptor a user needs to generate a random 160-bit binary word and perform a lookup for it. As a result the user gets a set of k nodeIDs (together with the contact addresses) that are closest to the generated word. This set also includes the requested word, if it happens to be an existing nodeID. After the lookup procedure is performed, the user needs to query one of the nodes from the previously returned set to download a descriptor[12]. The cost of downloading a descriptor is bounded by the order $\mathcal{O}(1)$, and, therefore, the overall costs of finding and fetching a single descriptor is in $\mathcal{O}(\alpha \cdot k \cdot \log(n))$.

[12] Every node stores at least its own descriptor.

Further, we assume that a user wants to find c different descriptors ($1 < c \leq n$). The expected number of queries to do that is $\sum_{i=0}^{c} \frac{n}{n-i}$ [16]. For small values of c ($c \ll n$), this roughly equals to c. In case when $c \to n$ (note, that this is the worst case), this problem is equivalent to the classical coupon collector problem [16], and therefore the expected number of searches is $c \cdot H_c$, where H_c is the c-th harmonic number. H_c can be approximated by $(\ln(c) + \gamma)$ where $\gamma \approx 0.577$ is the Euler-Mascheroni constant. Thus, the expected number of search queries is bounded by a function in $\mathcal{O}(c \cdot \log(c))$. If we assume that k,$\alpha \in \mathcal{O}(\log(n))$ and due the fact that a single search is bounded by $\mathcal{O}(\alpha \cdot k \cdot \log(n))$, we can conclude that the expected costs for a user to find c randomly chosen descriptors is bounded by a function in $\mathcal{O}\left(c \cdot \log^3(n) \cdot \log(c)\right)$.

Let m be the number of users in the whole network. Therefore, the expected costs for the total network distribution process in the whole network are bounded by $\mathcal{O}\left(m \cdot c \cdot \log(c) \log^3(n)\right)$. Because of the load-balancing properties of the Kademlia DHT, and also assuming uniform distribution of the nodeIDs, the expected network information distribution costs for a single node is bounded by $\mathcal{O}\left(\frac{m}{n} \cdot c \cdot \log(c) \cdot \log^3(n)\right)$.

If we assume, that the users need to find only a constant number of descriptors (c is constant in this case), the expected costs for a single node is bound by a function in:

$$\mathcal{O}\left(\frac{m}{n} \cdot \log^3(n)\right) \subset \mathcal{O}(m). \tag{1}$$

In case that every user wants to find every descriptor within the DHT ($c = n$), the expected costs are bound by a function in:

$$\mathcal{O}\left(m \cdot \log^4(n)\right). \tag{2}$$

However, since every query returns the k closest nodes to a given ID, possibly there are better strategies for finding every node, e.g., a DHT walk along the whole ID space. Please note that every node returns k closest nodes.

In the following, we compare our network information distribution approach to the one used in Tor. To this end, we analyze the latest version of the Tor directory protocol (version 3) [5].

Due to the nature of Tor's directory approach, the information about the descriptors needs to be refreshed at regular time intervals. In our analysis, we assume that the redistribution procedure needs to be repeated within a constant time interval. This is realistic, considering that every Tor descriptor needs to be updated not later than within 18 hours [5].

In the Tor network there is a differentiation between Directory Authorities (DA) and Directory Caches (DC). DAs are "semi-trusted" servers that must be operated by a Trusted Third Party (TTP). In contrast, a DC can be operated by any volunteering user. As mentioned above, Tor also distinguishes between the ORs (relaying servers) and the OPs (the client application). Let n be the number of ORs and m the number of OPs. For the sake of simplicity we assume that the set of directory servers (DA and DC) is distinct from the set of ORs.

This does not reflect the real world, but the influence of our analysis is neglible, since the number of DAs and DCs is much less than n. Let $a > 1$ be the number of DAs and $b \geq 1$ be the number of DCs within the network.

Briefly, the third version of the directory protocol in Tor works as follows: first, every OR uploads its descriptor to every locally known DA. Second, all DAs participate in a *consensus protocol* to create a consistent view over the network. This view is represented by a so-called *consensus document*, signed by every DA. In the next step every DC downloads this consensus document as well as every missing descriptor from the DAs. Finally, a regular fetching of the network status occurs. After that each OP downloads updated or missing descriptors from one of the DCs[13].

Because every OR uploads its descriptor to all known DAs, the cost of the first step of the distribution process can be estimated with a function in $\mathcal{O}(n)$ (for a single DA). Note, that the size of a descriptor does not depend on the size of the network and therefore does not appear in the $\mathcal{O}$ notation.

In the second protocol step, every DA sends its complete view on the network to every other DA. The size of this information is bound by $\mathcal{O}(n)$. Therefore, for a single DA, the costs of consensus are bound by a function in $\mathcal{O}(n \cdot a)$. The result of the consensus protocol run is saved in a so-called consensus document. This document is periodically downloaded by all DCs. After processing the consensus document, DCs download missing and outdated descriptors from the authorities. For this procedure the costs for a single DA is bound by $\mathcal{O}(\frac{n \cdot b}{a})$, under the assumption of uniformly distributed load over all DAs.

After this step, the consensus document and all descriptors are mirrored at every DC. The clients (OPs) download the consensus document directly from the DCs if they know any DC. If this is not the case, the OPs download the consensus document from one of the DAs. For the benefit of Tor, we assume that an OP knows at least one DC. In this case, the overall costs for a DA can be bound by a function in:

$$\mathcal{O}\left(n \cdot \left(1 + a + \frac{b}{a}\right)\right). \tag{3}$$

Next, we analyze the costs for a single DC. As mentioned above, the function for every DC – to download the consensus document – is bounded in size by $\mathcal{O}(n)$. Further, the consensus document and the missing/updated descriptors are downloaded by every OP from the DCs. We also assume this process to take place uniformly distributed among all DCs. Then, the costs for a single DC are bounded by a function in $\mathcal{O}(\frac{m \cdot n}{b})$, and thus the overall costs for a DC under our assumptions are bound in:

$$\mathcal{O}\left(n + \frac{m \cdot n}{b}\right). \tag{4}$$

If we assume that the number of DAs are in $\mathcal{O}(\log(n))$ and the number of DCs grows linearly with the number of ORs (both assumptions roughly reflect current state of the Tor network), we get the results shown in Table 1.

[13] Please check [5] for more information about the directory protocol

Note that in the case of our approach the numbers in the table reflect the situation where every user wants to find a constant number of descriptors within the DHT. The costs, when every user wants to find every descriptor is bounded according to Formula 2.

The presented analysis shows that the bottlenecks in the network information distribution process in Tor are the DAs. The analysis of our approach shows that even if k and α depend on the network size to harden eclipse attacks, our approach is able to provide a good scalability.

	Tor		Our approach
	DA	DC	
Costs/Server	$\mathcal{O}\left(\frac{n^2}{\log(n)}\right)$	$\mathcal{O}(m+n)$	$\mathcal{O}(m)$

Table 1. Comparison of Distribution Costs

7 Conclusion and Discussion

Although the distribution of network information and the anonymization process are mutually disjoint, the network information distribution has, however, an implicit impact on the anonymization process. Information leakage in the distribution process can be used to reduce or even revoke the anonymity on the network layer.

We find it challenging to make further steps towards the application of distributed methods and provide incentives for the community to advance the research in this topic. We showed that Kademlia can be suitable for the distribution of network information with respect to a local attacker. However, a number of open issues with using a DHT remain, especially regarding protection against several attacks like the *sybil attack* or denial of service attacks. Especially the *Sybil attack* represents a serious challenge. Another question is, if there are more efficient strategies for a local attacker to mount an eclipse attack.

In [4, 17] attacks on the route selection were shown where users posses only a partial knowledge about the network. This is also the case in our approach, since the user's local directory may only contain a subset of a whole directory. However, as all clients' requests to the DHT are done through server over encrypted connections, it is not trivial for an adversary to determine the local view of a client in the network and hence the attacks described in [4, 17] cannot be mounted easily.

Often the security properties of a DHT are considered inferior compared to a central directory approach. However, our eclipse attack analysis indicates that Kademlia can withstand a significant fraction of malicious nodes. In addition, our approach has some major advantages: we showed that it scales significantly better than the centralized directory approach of Tor. Also, by changing α in a nodelookup we can trade-off the probability of a successful eclipse attack with

the costs of doing a nodelookup. This gives us a flexible possibility to adopt our approach to our assumptions about the attacker, and the network.

References

1. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The Second-Generation Onion Router. In: Proceedings of the 13th USENIX Security Symposium. (2004)
2. : Tor Network Status. https://torstatus.kgprog.com/
3. Berthold, O., Federrath, H., Köpsell, S.: Web MIXes: A system for anonymous and unobservable Internet access. In: Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability, Springer-Verlag, LNCS 2009 (July 2000) 115–129
4. Gogolewski, M., Klonowski, M., Kutylowski, M.: Local View Attack on Anonymous Communication. In: Proceedings of ESORICS 2005. (September 2005)
5. Dingledine, R., Mathewson, N.: Tor Directory Protocol Specification. https://www.torproject.org/svn/trunk/doc/spec/dir-spec.txt
6. Freedman, M.J., Morris, R.: 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)
7. Freedman, M.J., Sit, E., Cates, J., Morris, R.: Introducing Tarzan, a Peer-to-Peer Anonymizing Network Layer. In: IPTPS 2002. Volume 2429 of Lecture Notes in Computer Science., Springer (2002) 121–129
8. Rennhard, M., Plattner, B.: Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In: Proceedings of the Workshop on Privacy in the Electronic Society, Washington, DC, USA (November 2002)
9. Tabriz, P., Borisov, N.: Breaking the collusion detection mechanism of morphmix. In: Proceedings of the Sixth Workshop on Privacy Enhancing Technologies (PET 2006), Cambridge, UK, Springer (June 2006) 368–384
10. Nambiar, A., Wright, M.: Salsa: a structured approach to large-scale anonymity. In: CCS '06: Proceedings of the 13th ACM conference on Computer and communications security, New York, NY, USA, ACM (2006) 17–26
11. Mittal, P., Borisov, N.: Information leaks in structured peer-to-peer anonymous communication systems. In: Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS 2008), Alexandria, Virginia, USA, ACM Press (October 2008) 267–278
12. Maymounkov, P., Mazires, D.: Kademlia: A peer-to-peer information system based on the xor metric. In: Peer-to-Peer Systems: First International Workshop, IPTPS 2002 Cambridge, MA, USA, March 7-8, 2002. Revised Papers. Volume 2429/2002 of Lecture Notes in Computer Science., Springer Berlin / Heidelberg (2002) 53–65
13. Danezis, G., Clayton, R.: Route fingerprinting in anonymous communications. In: Peer-to-Peer Computing, IEEE Computer Society (2006) 69–72
14. Singh, A., Ngan, T.W., Druschel, P., Wallach, D.S.: Eclipse attacks on overlay networks: Threats and defenses. In: INFOCOM, IEEE (2006)
15. Douceur, J.: The Sybil Attack. In: Proceedings of the 1st International Peer To Peer Systems Workshop (IPTPS 2002). (March 2002)
16. Motwani, R., Raghaven, P.: Randomized Algorithms. Cambridge University Press (1995)
17. Danezis, G., Syverson, P.: Bridging and fingerprinting: Epistemic attacks on route selection. In: Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008), Leuven, Belgium, Springer (July 2008) 151–166