Anonymity for Trust Holders using k-anonymity Chord

Ahmet Burak Can and Bharat Bhargava
Department of Computer Science, Purdue University
West Lafayette, IN 47907
{acan, bb}@cs.purdue.edu

This research is supported by NSF grants ANI 0219110, IIS 0209059 and IIS 0242840.

Abstract

Anonymity is important in a peer-to-peer system to protect peers that offer/request services. We propose an anonymity scheme on Chord to provide a peer k-anonymity protection against a global passive adversary who can sniff all the communication on a network. For collaborating adversaries, anonymity is protected as long as they perform only passive attacks. An encryption scheme ensures that peers can authenticate the sender of an anonymous response. A trusted third party is not needed. We achieve a performance comparable to Chord. The efficiency and anonymity guarantees are shown theoretically. As a case scenario, anonymous access to trust information is studied on a trust model. Peers query the information stored by trust holders. We show how trust holders anonymously respond to such queries. Attack scenarios are discussed in detail to verify security of the scheme.

Index Terms

Peer-to-peer systems, anonymity, cryptography, trust management, security.

I. INTRODUCTION

Data confidentiality is important in many Internet applications such as e-commerce, online banking, and most remote access applications. Encryption algorithms and public key authentication schemes protect the content of a communication from unauthorized persons. Another dimension in a communication is to protect anonymity of communicating parties. In some applications, parties may want to keep their identity confidential, e.g., negotiations between two incorporating companies and secret messaging among people. Encryption of messages does not protect anonymity if adversaries are capable of sniffing a large portion of the network. Most anonymity related systems rely on mix networks [1] and onion routers [2]. In these systems, all anonymized communication goes through some trusted nodes. These nodes encrypt and shuffle incoming and outgoing traffic so a global passive adversary can not determine the identity of communicating parties.

In a peer-to-peer system, anonymity is needed for censorship resistance[3], publisher/subscriber protection in storage systems[4], [5], witness/trust holder anonymity in a trust model [6], [7], or a general protection against malicious peers [8], [9]. Probabilistic random path building [8], tunnelling [9], limitations on routing information exchange [3], creating multicast groups [10] ,and broadcasting [4], [5], [6] are used as methods of protecting anonymity. These methods are vulnerable to large scale sniffing attacks. An anonymity scheme is needed against global passive adversaries. Mix networks or onion routers can be adapted for peer-to-peer systems to prevent such attacks but they depend on trusted nodes. Due to decentralized nature of peer-to-peer systems, trusted nodes are not preferred in order to anonymize network communication.

Efficiency of an anonymity scheme is important for its usability. An anonymity scheme on a peer-to-peer system should not use network flooding which causes excessive traffic [11]. A distributed hash table (DHT) [12], [13], [14] can be adapted to create an efficient anonymity scheme. Besides efficiency, authenticity of an anonymous reply is important [6]. A malicious peer may forge fake anonymous replies in the name of others. To mitigate such attacks, a peer can add a signature to its anonymous reply. Other peers verify authenticity of a reply by checking the signature. However, this verification should be done without revealing the identity of the signer.

We propose k-anonymity Chord and oblivious replying to provide an anonymous, efficient, and authenticated access method to a peer who offers services or stores information for other peers. Oblivious replying is a cryptographic protocol which guarantee k-anonymity [15] protection against global passive adversaries. The basic idea is that k peers sends k replies to each request and these replies can not be distinguished from each other. An adversary can not track down the sender of a reply. In case collaboration, anonymity is protected if adversaries only perform passive attacks. We do not consider active adversaries who can drop, modify, and forge messages. Countering such an adversary might be expensive in terms of computation and network communication [15]. Instead of a more secure but complex anonymity scheme, we aim a more practical one.

As a case scenario, we adapt our scheme for trust holder anonymity. In a trust model [16], [17], [18], peers establish long-term trust relationships to reduce the risk in future interactions. Each peer becomes a *trust holder* by storing trust information of other peers. A malicious peer may attack its trust holders to avoid dissemination of its bad reputation. Trust holders must be anonymous when answering queries about trust information. Anonymity increases the availability of trust information since trust holders are less vulnerable to denial of service (DOS) attacks. This provides a peer more motivation to perform trust holding duty.

Peers are assumed to register their pseudonyms and some encryption keys to a bootstrap server before joining the network first time. After registration, peers join into two overlay networks: *service* and *trust* networks. The service network can be overlaid on any network substrate. The trust network should be overlaid on k-anonymity Chord. A service request happens as follows. A peer queries the service network to find a particular service, e.g., searching a file in a file sharing network. After finding a service provider, the peer sends a query to the trust network. Trust holders of the service provider anonymously reply the trust information. If the service provider is trustworthy, the peer starts an interaction. Replies of trust holders is authenticated through an encryption scheme. Thus, an adversary can not forge inauthentic trust information. Asymptotic running time of anonymous reply method is $O(\log^2 N/\eta)$ where N is the network size and η is the maximum number of replies that may fit into a network packet.

Section II presents related research. In Section III, communication and encryption architecture has been explained. Section IV explains k-anonymity Chord and three reply methods to protect anonymity. We explain the disadvantages of two previously proposed reply methods and present oblivious reply method as our approach. After discussing future research opportunities in Section V, we conclude in Section VI.

II. Related Work

Chaum [1] first proposed mix networks to protect anonymity of communicating parties for delay tolerant applications. Babel mixes [19] tried to maximize anonymity by introducing variable and large latency. Probabilistic security [20] and cryptographic mix nodes [21], active attacks [22] and practicality [23] of mix networks have been studied. Although different techniques are defined on these papers, the common idea is to use trusted mix nodes for routing messages in an untraceable manner to protect anonymity of communicating parties.

Onion routers [24], [2] form an overlay network to build anonymous, bi-directional virtual circuits for real-time communication. While mix networks are generally designed for delay tolerant applications, e.g., e-mail systems, onion routing is more feasible for real-time applications such as HTTP. Tor [25] expands onion routing with forward secrecy, congestion control, integrity checking and configurable exit policies. Tor uses directory servers to maintain onion router topology and certificates.

Our scheme aims to protect anonymity without relying on a trusted mix network or onion routers. Thus, k-anonymity Chord should be considered as a different category of anonymity systems. However, we use some ideas from the encryption schemes in [1], [24].

Several anonymity systems are proposed on peer-to-peer systems. Crowds[8] forms groups (crowds) of nodes (jondo). A crowd collaborates to protect anonymity of a jondo from outsider. The size of a crowd determines the level of anonymity. However, a local eavesdropper may break anonymity. Sender, receiver, publisher anonymity on peer-to-peer storage systems have been studied in Freenet [4] and Freehaven [5]. An encryption scheme is used while broadcasting requests and anonymously accessing storage providers. Tarzan [9] establishes a random tunnel between two communicating peers to protect their anonymity. Since none of the peers on a tunnel know the whole path, an adversary can not figure out communicating peers. MorhpMix [26] defines a peer-to-peer mix network with a collusion detection mechanism. Like Tarzan, random mix nodes are selected during an anonymous communication. All these peer-to-peer schemes do not provide protection against a global passive adversary.

Anonymity on Chord have been studied by using recursive, randomized, indirect, split, bidirectional routing [27], and virtual nodes [28]. Achord [3] proposes enhancements to provide censorship resistance on Chord. These

TABLE I

PRELIMINARY NOTATIONS

Notation	Description
U_{BS}, R_{BS}	the bootstrap server's public and private keys
P_i	a peer with identifier i
ID_i	P_i's pseudonym in the service network
TID_i	P_i's pseudonym in the trust network
U_i, R_i	P_i's public and private key in the service network
TU_i, TR_i	P_i's public and private key in the trust network
OU_i, OR_i	P_i's public and private key for oblivious replying operations in the trust network
$K(M)$	encryption of M with key K
$H[M]$	hash digest of M
$X\|Y$	concatenation of X and Y

schemes strengthen the responder anonymity in a probabilistic model but do not protect anonymity against global passive adversaries.

A similar study to our scheme, Trustme [6], uses an encryption framework to access trust holders anonymously. Peers flood trust queries to the network. Trust holders can send anonymous, authenticated replies to trust queries. Large number of possible repliers in broadcasting provide a probabilistic anonymity protection for a trust holder. However, flooding causes excessive network traffic and does not protect a trust holder against a global passive adversary.

A stronger form of anonymity scheme, dining cryptographer networks (DC-net) [29], provides unconditional anonymity protection for the sender of a message in a group of participants. Assuming the group size is N, this approach requires $O(N^2)$ message exchange for each message sending. Furthermore, before each message sending, $O(N^2)$ encryption keys should be distributed among N participants using an external secure method. This makes DC-nets impractical for real life scenarios.

III. ARCHITECTURE

Most peer-to-peer networks need a bootstrap server to provide a connection point to the network for new peers. There might be multiple bootstrap servers to provide tolerance to failures and attacks. For simplicity of the notation, the rest of the paper will consider one bootstrap server. Peers register themselves to the bootstrap server (BS) when joining the network for the first time. In our scheme, it is also a basic certification authority for pseudonyms and keys of peers. The bootstrap server has a public/private key pair U_{BS}, R_{BS}. We assume all peers learn the public key, U_{BS}, in a secure way e.g. through a secure web site.

P_i denotes a peer with identifier i. ID_i is the pseudonym of P_i in the service network. ID_i is randomly selected by P_i before registration. Similarly, TID_i is the pseudonym of P_i in the trust network. TID_i is assigned by the bootstrap server during the registration operation. ID_i and TID_i have no relation with each other. P_i has one public/private key pair, $\{U_i, R_i\}$, for the service network operations and two public/private key pairs, $\{TU_i, TR_i\}$ and $\{OU_i, OR_i\}$, for the trust network operations. All key pairs are randomly selected by P_i and have no relation with each other. We assume that peers have good random number generators to prevent brute force guessing attacks on key pairs.

We give simple notations to describe message formats. $K(M)$ stands for the encryption of M with key K which can be a public, private, or symmetric key. $H[M]$ is the hash digest of M. $X|Y$ denotes the concatenation of X and Y. Table I lists the notations for easy reading of the following sections.

A. Adversary Model

An adversary passively observes (sniff) the trust network to reveal pseudonym (TID) or IP number of a trust holder. It[1] does not have active attack capabilities such as dropping, modifying, forging network packets. It has polynomial time computational capabilities and can not break cryptographic algorithms in polynomial time. We also assume searching all pseudonym space in brute force manner is computationally infeasible.

We first assume that an adversary has only local passive observation capabilities. Such an adversary can observe packets destined to itself or its local network and may learn about other peers by generating arbitrary number of query packets. We explain two previously studied methods to counter this adversary (Section IV-A and IV-B).

In Section IV-C, we extend our scheme for global passive adversary model [1], [2] who can observe all traffic on the trust network. Furthermore, if the global adversary can compromise (or collaborate with) some peers, we show that the scheme protects anonymity of a trust holder as long as the compromised peers behave semi-honestly. In semi-honest model[30], an adversary follows the protocols properly but passively observes the network communication. Privacy literature commonly uses this adversary model.

B. Peer Registration

We assume that P_i is joining the network for the first time. It registers itself to the bootstrap server as follows:

1) P_i starts the registration operation by sending a registration request, $U_{BS}(ID_i|U_i|TU_i|OU_i|r_1)$, to the bootstrap server. Due to encryption with U_{BS}, only the server can read the request. The server decrypts the message and stores ID_i, U_i, TU_i, OU_i for future accountability. r_1 is a random value selected by P_i.
2) The server selects another random value, r_2, and sends back $U_i(TS|r_1|r_2)$ to P_i as a challenge. Since the whole message is encrypted with U_i, only P_i can decrypt the message. P_i verifies r_1 value. If r_1 value is correct, the server is authenticated. TS is a time-stamp representing the time of message creation.
3) P_i sends $R_i(TR_i(TS|r_2)|OR_i(TS|r_2))$ to the server. The server decrypts the message and verifies TS, r_2 values. If these values match, the server has verified that P_i has R_i, TR_i, OR_i keys. Otherwise, an adversary could be replying P_i's message or P_i is trying to certify keys which may belong to others.
4) After passing the challenge/response protocol, the server selects a TID_i value representing P_i's pseudonym in the trust network. It sends $U_i(R_{BS}(ID_i|U_i|TS)|R_{BS}(TID_i|OU_i|TS))$ as a reply. $R_{BS}(ID_i|U_i|TS)$ part is a *service certificate*. This certificate proves P_i's registration to other peers. $R_{BS}(TID_i|OU_i|TS)$ part informs P_i about its TID_i and is used as a certificate during oblivious reply operations explained later in Section IV-C. Since these certificates are encrypted with R_{BS}, P_i or another peer can not forge them. P_i decrypts the message using R_i, U_{BS} and verifies TS value. If TS value is same as the previous value, the server is sending the message. P_i stores these certificates for future use.
5) The bootstrap server randomly selects several trust holders for P_i. Let P_j be such a trust holder. The server sends $U_i(R_{BS}(ID_i|TU_j|MTID_j|THN_i|TS))$ to P_i. The inner part $R_{BS}(ID_i|TU_j|MTID_j|THN_i|TS)$ is a *trust certificate*. It tells P_i that a trust holder is charged to store P_i's trust information. P_i can not learn P_j's pseudonym in the trust network since TID_j is not added into the certificate. $MTID_j$ is an anonymized version of TID_j explained in Section IV. Using $MTID_j$ value, P_i or another peer can access the trust information stored by P_j. THN_i is the number of trust holders assigned to P_i.
6) The bootstrap server sends a *trust holder certificate* to each trust holder. For example, P_j's trust holder certificate is $TU_j(R_{BS}(H[ID_i]|TU_j|TS))$. This certificate tells that P_j is charged to hold P_i's trust information. P_i's pseudonym in the service network is not added to protect the anonymity of P_i. However, using $H[ID_i]$

[1]Considering an adversary is a peer, we will use "it" to refer the adversary.

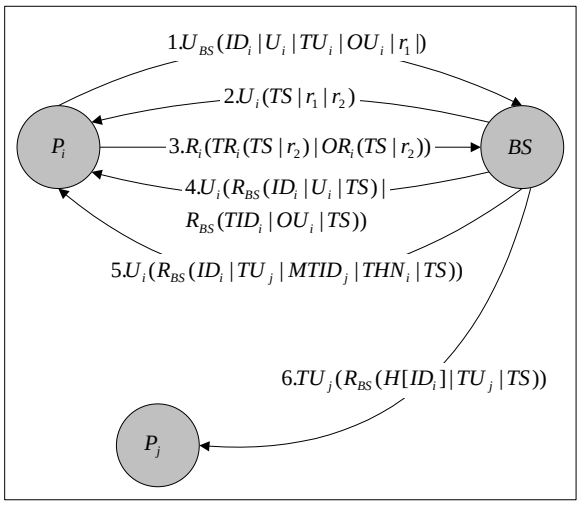

Fig. 1. Registration of P_i to the bootstrap server

TABLE II

NOTATIONS ABOUT THE CERTIFICATES ISSUED BY THE BOOTSTRAP SERVER

Notation	**Description**
$MTID_j$	a masked version of TID_j
THN_i	number of trust holders assigned for P_i
TS	time-stamp
$R_{BS}(ID_i\|U_i\|TS)$	P_i's service certificate
$R_{BS}(ID_i\|TU_j\|MTID_j\|THN_i\|TS)$	P_i's trust certificate about trust holder P_j
$R_{BS}(TID_i\|OU_i\|TS)$	P_i's certificate for oblivious replying operations
$R_{BS}(H[ID_i]\|TU_j\|TS)$	P_j's trust holder certificate about P_i

field, P_j can still answer trust queries about P_i. TU_j is added into the certificate to distinguish the certificates of other trust holders.

Figure 1 briefly explains the peer registration operation. In step 1, P_i sends the necessary information to the bootstrap server. The server sends back a random challenge in step 2. After P_i sends a response to the server's challenge (step 3), the server sends back a service certificate and a certificate for trust network operations in step 4. Then, the server selects P_j as a trust holder of P_i and sends a trust certificate about P_j (step 5). In the last step, the server sends a trust holder certificate to P_j. For the rest of this section, we assume that P_i is a service provider, P_j is a trust holder of P_i, and P_r is requesting a service from P_i.

C. Searching a Service Provider and Sending a Trust Query

P_r sends a query to the service network to find a service, e.g., a particular file. As a service provider, P_i sends its service certificate, $R_{BS}(ID_i|U_i|TS)$, to P_r. P_r decrypts the certificate with U_{BS} and runs a challenge/response protocol to make sure that P_i is a registered service provider and the owner of the certificate. Additionally, an

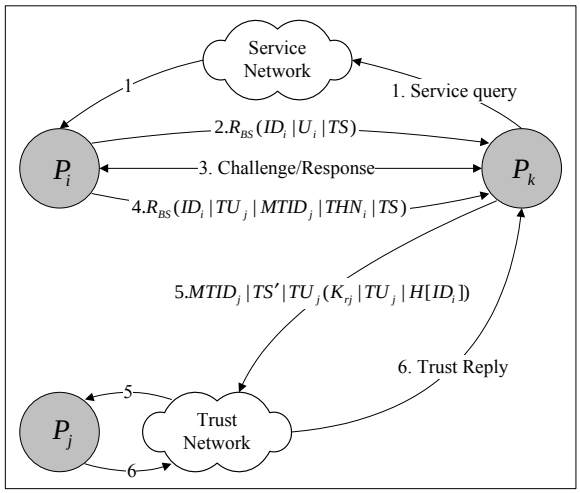

Fig. 2. P_r is searching for a service provider (P_i) and querying its trust information in the trust network

adversary may pretend to be P_i by replying P_i's certificate. Thus, P_r prepares a challenge using U_i and sends it to P_i. Only P_i can respond to the challenge so P_r verifies P_i's identity. This step is very similar to challenge/response steps during the peer registration.

P_i sends its trust certificates to P_r. In our case, P_i sends $R_{BS}(ID_i|TU_j|MTID_j|THN_i|TS)$ to inform P_r about P_j. Using U_{BS}, P_r decrypts the certificate and compares ID_i with the value from the service certificate. If ID_i values match, P_j is a legitimate trust holder. By examining the certificate, P_r verifies the authenticity of P_j without knowing its identity.

P_i may cooperate with some trust holders to elevate its trust level and try to avoid involvement of other trust holders. To achieve this, P_i may not send certificates of other trust holders to P_r. However, THN_i value informs P_r about the existence of other trust holders and forces P_i to send all certificates to the requester.

Then, P_r sends a *trust query*, $MTID_j|TS'|TU_j(K_{rj}|TU_j|H[ID_i])$, to the trust network. TS' is a time-stamp and is unique among all queries of P_r. K_{rj} is a session key randomly created by P_r. Due to encryption with TU_j, only P_j can read $TU_j(K_{rj}|TU_j|H[ID_i])$ part and learn K_{rj} key. TU_j and $H[ID_i]$ fields prevent forgery of the encrypted part. When a query arrives to P_j, it checks TU_j to understand if the query is destined to itself. It looks up $H[ID_i]$ value in its trust holder certificates.

The trust query is routed in the trust network till P_j receives it. When P_j gets the trust query, it sends back with a *trust reply* message. The details of routing and replying operations are explained in Section IV. Figure 2 shows the message exchanges during a service and a trust query.

IV. k-ANONYMITY CHORD

Chord [12] can provide efficient access to trust information. However, Chord does not provide anonymity for the responder of a search request since peers can partially learn the network structure using finger tables. Peers forwarding a search request may guess where the search will end. Additionally, a peer may learn more about a portion of the address space by sending excessive finger requests [3].

We propose k-anonymity Chord to provide k-anonymity protection [31] for peers. k-anonymity Chord performs peer join, leave, and finger table maintenance operations like a normal Chord structure. In our case, trust network is overlaid on a k-anonymity Chord structure. A peer joins the network with its TID value, e.g., P_j joins with TID_j. Thus, k-anonymity Chord can be thought as a network of trust holders that is organized according to TID

values. When a trust holder replies to a trust query, its identity can not be distinguished from k other peers. A trust holder has k-anonymity protection when answering trust queries.

Let P_i be a service provider, P_j be a trust holder of P_i and P_r wants to get a service from P_i. After receiving P_i's service and trust certificates, P_r sends a trust query destined to P_j. Since neither P_i nor P_r know TID_j, the trust query contains $MTID_j$ value. $MTID_j$ is an anonymized version of TID_j where the last m bits are set to zero. In the trust query operations, $MTID_j$ represents a range of pseudonyms between $MTID_j$ and $MTID_j + 2^m$. We call this range as *search range* and the peers in the search range as *target peers*. The bootstrap server decides the value of m so that the expected number of target peers is equal to k. Since the bootstrap server registers all peers, it can compute m precisely. To explain $MTID_j$ selection, we give a numerical example:

Chord peers are located on a 2^n circular address space. We assume the bootstrap server uniformly distributes TID values on the address space. Suppose $n = 32, k = 64$, $TID_j = 0 \times 12345678$ and there are 2^{16} peers in the network. Let X be an indicator random variable that represents if there is a peer on a particular location in the address space (When $X = 1$, there is a peer on a that location). The probability of $X = 1$ is

$$P(x = 1) = \frac{2^{16}}{2^{32}} = \frac{1}{2^{16}}$$

and the expected number of nodes on a particular location is

$$E[X] = \sum_x x \cdot P(x) = 1 \cdot P(x = 1) + 0 \cdot P(x = 0) = \frac{1}{2^{16}}$$

Let Y be a random variable representing the number of peers that fall into a search range. The bootstrap server selects a search range which has $Y \geq k = 64$ expected number of peers. Let S be the number of locations in the search range. Due to the uniformity of distribution, the expected number of peers in the search range is

$$E[Y] = E[X] \cdot S = \frac{1}{2^{16}} \cdot S \geq 64$$

The bootstrap server finds that $S \geq 2^{24}$. This inequality suggests us to select $m \geq \log_2 S = \log_2 2^{24} = 24$. Then, the bootstrap server computes $MTID_j$ as follows:

$$\begin{aligned} MTID_j &= (0 \times 12345678) \wedge (0 \times FF000000) \\ &= 0 \times 12000000 \end{aligned}$$

$MTID_j = 0 \times 12000000$ means that P_j has a TID_j between 0×12000000 and $0 \times 12FFFFFF$. The expected number of peers in this range is 64 due to our selection.

Let $P_0, P_1 \ldots P_{k-1}$ be k target peers located between $MTID_j$ and $MTID_j + 2^m$ range. We define a two-phase routing method for trust queries. First phase is a recursive Chord search operation to find the successor of $MTID_j$ which is P_0. P_r starts this search operation by sending a trust query, $MTID_j|TS'|TU_j(K_{rj}|TU_j|H[ID_i])$, to its closest finger preceding P_0. The receiving peer forwards the query by looking up $MTID_j$ value. Forwarding peers store the query for a period of time. Stored messages will be used later to forward P_j's reply back to P_r. TS' value gives a hint for the expiration time of the query. Forwarding operation continues until P_0 receives the query. After the query reaches to P_0, the second phase starts.

In the following sections, we explain three methods for the second phase. The first two methods are already used in several previous proposals. These methods are vulnerable to global sniffing. In the third method, we describe our oblivious replying scheme to protect anonymity against a global passive adversary. In the attack scenarios, P_r tries to identify P_j. Note that, P_i may pretend to be P_r to learn P_j's identity.

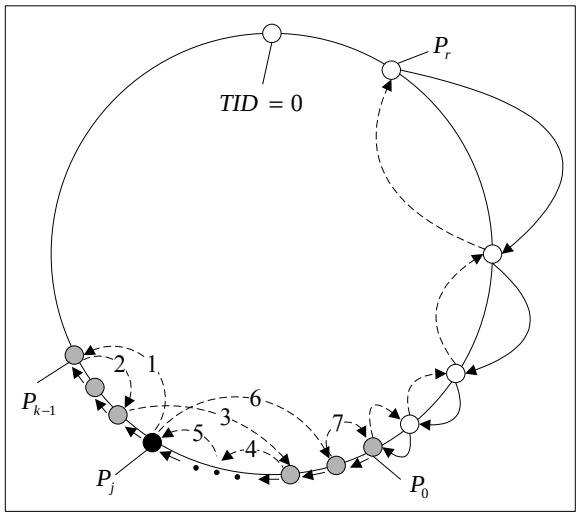

Fig. 3. Two-phase trust query forwarding with random replying

A. Naive Replying

After receiving P_r's query, P_0 tries to decrypt $TU_j(K_{rj}|TU_j|H[ID_i])$ part. If TU_j key does not match, P_0 forwards the query to its successor P_1. If P_1 is not the receiver, it forwards the query to P_2. This process continue until P_j receives the query. If P_j is not online, the query reaches to the last target peer, P_{k-1}. In this case, P_{k-1} sends back a *no reply* message or ignores the query. When P_j receives the query, it searches $H[ID_i]$ value in its certificate database. It sends a *trust reply*, $MTID_j|TS'|K_{rj}(TV_i|H[ID_i]|TS')$, to its predecessor. All target peers forwards the reply to their predecessors until P_0 receives it. P_0 forwards P_j's reply to the peer who forwarded P_r's query. All peers on the query path between P_0 and P_r do the same operation. Finally, P_r receives the reply message and checks $H[ID_i]$ and TS' values. If the values are correct, TV_i value is authentic.

This method protects the authenticity and integrity of the trust reply. A malicious target peer can not obtain K_{rj} key and forge a reply message. If P_r has only local observation capability and out of the search range, P_j can not be distinguished from other $k-1$ target peers. Thus, P_j has k-anonymity protection [31] if P_r has local observation capability.

If P_r falls into the search range by chance, or succeeds to inject some decoy peers into the search range (Sybil attack [32]), or compromises some target peers, it may guess P_j's identity. The number of possible repliers decreases as the trust query advances to P_{k-1}. If P_j is close to P_{k-1}, target peers preceding P_j may guess P_j more accurately. Furthermore, if P_r has global observation capability, it can learn P_j's identity. P_r can observe that P_j neither forwarded the query message nor got a reply message from a successor. Thus, naive replying does not protect anonymity against global passive adversaries and collaborating adversaries.

B. Random Replying

P_0 and all target peers forwards the query to their successors until P_{k-1} receives it. Only P_j can decrypt the content of the query. P_j sends the reply to a random target peer except P_0. [2]. Receiving target peer forwards the reply to another random target peer with a p_f probability. A peer may forward the same reply several times. A forwarder only knows its preceding hop, but can not be sure if the preceding hop is P_j. Finally, a peer decides

[2]We assume target peers already know about each other. Peers may exchange IP addresses during the query forwarding

not to forward and sends the reply message to P_0. As in the naive method, the peers between P_0 and P_r forwards the reply message until P_r receives it. Figure 3 depicts two-phase routing of the trust query with random replying. Points on the circular Chord ring represent the peers involved in the routing of P_r's query. Gray points denote the target peers. The black point represents P_j. Normal and dashed arrows represent the paths of P_r's query and P_j's reply respectively.

As in the naive method, random replying protects anonymity of P_j if P_r is a local passive adversary. In case of a collaboration between P_r and some target peers, P_j has probable innocence [8] if

$$k \geq \frac{p_f}{p_f - 1/2}(c+1) \tag{1}$$

k is the number of target peers and c is the number of collaborators of P_r. Random replying provides a probabilistic anonymity protection for P_j but does not eliminate the chance of being identified. If $c > \frac{k}{2} - 1$, there is no probable innocence for P_j. Thus, random replying does not protect anonymity when P_r compromises half of the target peers. If P_r is a global passive adversary, P_j has no anonymity. P_r can observe all traffic among peers even they randomly forward the reply.

C. Oblivious Replying

Oblivious replying is a secure method to protect anonymity of P_j against a global passive adversary. Moreover, this method is resistant against collaborating passive adversaries in semi-honest adversary model [30]. The basic idea is that each target peer generates a reply message and P_r receives k replies which can not be linked with the senders. P_j's reply is one of these k replies.

We assume that target peers already know each other and exchanged $R_{BS}(TID_i|OU_i|TS)$ certificates. Public key encryption is assumed to ensure semantic security [33]. This implies that encryption of a message depends on the message and a sequence of coin tosses. Encryption of a plaintext with the same public key results in a different ciphertext in each trial. However, the decryptions of these ciphertexts with the private key gives the same plaintext.

As in the random replying method, P_r's query message is forwarded in the search range until P_{k-1} receives it. P_{k-1} tries to decrypt $TU_j(K_{rj}|TU_j|H[ID_i])$ part. If the decryption is successful, it prepares O_{k-2}^{k-1} as follows:

$$O_{k-2}^{k-1} \quad = \quad OU_{k-2}(OU_{k-3}(\ldots OU_1(OU_0(K_{rj}(TV_i|H[ID_i]|TS'))))\ldots))$$

O_{k-2}^{k-1} denotes P_{k-1}'s *oblivious reply* which is to be delivered to P_{k-2}. If the decryption fails, P_{k-1} prepares a similar reply but the innermost layer of O_{k-2}^{k-1} contains $K_{random}(RTV|RHID|TS')$ part. K_{random} is a randomly generated key. RTV and $RHID$ are randomly selected trust and hash values respectively. These random values have the same amount of bits as authentic values.

P_{k-1} sends $MTID_j|TS'|O_{k-2}^{k-1}$ to its predecessor, P_{k-2}. P_{k-2} decrypts the top layer of O_{k-2}^{k-1} which becomes O_{k-3}^{k-1}. Then, P_{k-2} prepares O_{k-3}^{k-2} and sends $MTID_j|TS'|(O_{k-3}^{k-1} \cup O_{k-3}^{k-2})$ to P_{k-3}. The operation $\cup$ denotes the concatenation of O_{k-3}^{k-1} and O_{k-3}^{k-2} in a random order. Since O_{k-3}^{k-1} and O_{k-3}^{k-2} are encrypted and contain the same number of bits, P_{k-3} can not distinguish these replies after the randomization. P_{k-3} peels off the top layer from O_{k-3}^{k-1} and O_{k-3}^{k-2}. It creates O_{k-4}^{k-3} and sends $MTID_j|TS'|(O_{k-4}^{k-1} \cup O_{k-4}^{k-2} \cup O_{k-4}^{k-3})$ to P_{k-4}.

This process is repeated by all target peers until P_0 receives $MTID_j|TS'|(O_0^{k-1} \cup O_0^{k-2} \cup \ldots \cup O_0^2 \cup O_0^1)$. After peeling off the last layers, P_0 adds its own reply message and sends all replies to P_r as described in the previous reply methods. P_r decrypts all replies using K_{rj}. The reply containing the correct $H[ID_i]$ and TS' values is the reply of P_j. Figure 4 shows the flow of oblivious replies among target peers.

If P_r is a global passive adversary, P_j has k-anonymity. P_r can observe communication of a target peer but it can not link any incoming reply of the peer with an outgoing reply. Because, decryption of top layers of incoming replies and randomization of outgoing replies on each target peer and our semantic security assumption do not

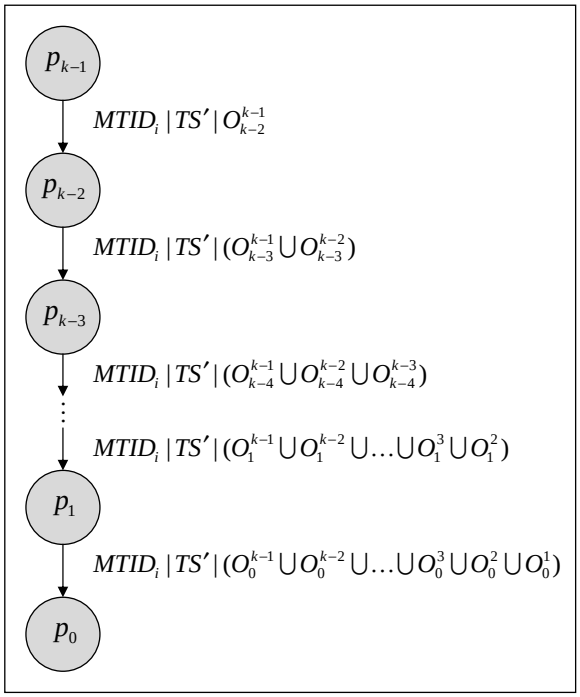

Fig. 4. Message communication among target peers in oblivious replying method

allow P_r to track down the replies. Identical reply sizes make all replies look same. To demonstrate this, we present some lemmas and theorems.

Lemma 1: P_r can not get any information about the sender of an oblivious reply by sniffing incoming and outgoing replies of a target peer.

Proof: Let P_x be a target peer where $0 < x < k-1$. P_x receives $MTID_j|TS'|(O_x^{k-1} \cup O_x^{k-2} \cup \ldots \cup O_x^{x+1})$ from P_{x+1}. P_r can not learn any information about the content of $O_x^{k-1}, O_x^{k-2}, \ldots, O_x^{x+1}$ by sniffing the network. Since P_r does not know $OR_x \ldots OR_0$ keys, it can not decrypt these replies. This situation is same for the outgoing replies.

P_r may try to link P_x's outgoing replies with incoming replies. If P_r can establish such links, it observes all communication among peers and links each reply with a sender. However, P_r can not do this, as explained below.

P_x decrypts top layers of $O_x^{k-1}, O_x^{k-2}, \ldots, O_x^{x+1}$ and sends $MTID_j|TS'|(O_{x-1}^{k-1} \cup O_{x-1}^{k-2} \cup \ldots \cup O_{x-1}^{x+1} \cup O_{x-1}^{x})$ to P_{x-1}. Due to the semantic security, P_r can not get O_x^{k-1} by encrypting one of $O_{x-1}^{k-1}, O_{x-1}^{k-2}, \ldots, O_{x-1}^{x+1}, O_{x-1}^{x}$ with OU_x key[3]. Furthermore, randomized order of outgoing replies and identical reply sizes do not allow P_r to find a link between O_x^{k-1} and any of P_x's outgoing replies. This case is the same for $O_x^{k-2}, O_x^{k-3} \ldots, O_x^{x+1}$. None of the incoming replies can be linked to an outgoing reply without knowing OR_x key. Thus, learning a peer's incoming and outgoing replies does not give any information about sender of a reply. ■

Theorem 1: If P_r has global passive observing capability, oblivious replying provide k-anonymity for P_j.

Proof: P_r can observe all incoming and outgoing messages of $P_0, P_1, \ldots, P_{k-1}$, but can not obtain any information about the senders of replies due to Lemma 1. P_r can decrypt P_0's outgoing replies since last layers of all replies is encrypted with K_{rj}. It can learn P_j's reply. However, it can not find a link between P_j's reply

[3] P_r can learn OU_x key by sniffing the messages during the exchange of $R_{BS}(TID_i|OU_i|TS)$ certificates

and any of P_0's incoming replies due to randomization of outgoing replies and encryption with OR_0. P_r can not distinguish P_j's reply from other $k-1$ replies so P_j has k-anonymity protection. ■

P_r may succeed in injecting decoy peers into the search range (Sybil attack [32]) or compromise some target peers. With the help of collaborators, P_r may track down some replies and identify P_j's reply. We claim that if compromised target peers behave semi-honestly, replies of honest target peers can not be trackable. In this case, P_j's anonymity is proportional to the number of honest target peers. A more precise proof is as follows.

Lemma 1: If all except two target peers are collaborators of P_r, replies of two honest target peers can not be distinguished from each other as long as collaborators behave in semi-honest adversary model.

Proof: Without loss of generality, let P_x and P_y be two honest target peers where $0 < x < k-2$ and $x+1 < y \leq k-1$. Since all other peers are compromised, P_{x+1} and P_{x-1} are two compromised peers. All peers between P_{k-1} and P_{x+1} except P_y are collaborators so P_{x+1} can identify P_y's reply. When P_{x+1} receives incoming replies, it creates O_x^{x+1}, peels off top layers of incoming replies and sends $MTID_j|TS'|(O_x^{k-1} \cup O_x^{k-2} \cup \ldots \cup O_x^{x+2} \cup O_x^{x+1})$ to P_x. Then, P_x repeats the same operations and sends $MTID_j|TS'|(O_{x-1}^{k-1} \cup O_{x-1}^{k-2} \cup \ldots \cup O_{x-1}^{x+2} \cup O_{x-1}^{x+1} \cup O_{x-1}^{x})$ to P_{x-1}. Knowing oblivious replies of all collaborators, P_{x-1} can figure out replies of P_x and P_y. However, it can not distinguish O_{x-1}^y and O_{x-1}^x from each other. Since P_x peels of the top layer of O_x^y and randomizes outgoing replies, P_{x-1} can not find a link between O_x^y and any of O_{x-1}^y or O_{x-1}^x. In P_{x-1}'s view, O_{x-1}^y and O_{x-1}^x are equally likely to be sent by P_y or P_x. Neither P_{x+1} nor P_{x-1} can distinguish the replies of P_x and P_y. Other collaborators can not learn further information than P_{x+1} and P_{x-1}. Hence, P_r and its collaborators can not distinguish the replies of P_x and P_y from each other. ■

Theorem 1: If P_r collaborates with m target peers, oblivious replying provides $k-m$ anonymity protection for P_j as long as compromised peers behave in semi-honest adversary model.

Proof: Immediate from Lemma 2. If there are $k-m$ honest target peers, P_r and its collaborators can not distinguish $k-m$ honest replies from each other and can not track down the sender of each reply. Then, P_j's reply can not be linked to any of $k-m$ honest peers and P_j has $k-m$ anonymity protection. ■

We conclude that oblivious replying provides k-anonymity protection to P_j against global passive adversaries and semi-honest collaborators. For a better understanding of our encryption scheme, similar schemes in [1], [24], [29] can be used.

D. Security Analysis of Oblivious Replying

We identify several active and passive attacks that may break our scheme. Following scenarios can be expanded for single and collaborative adversaries.

Active adversaries. The proposed scheme does not protect anonymity against active adversaries. Identifying vulnerabilities of the scheme is helpful to design more secure schemes in the future research. Thus, we outline several active attacks as follows:

- P_r may forge all incoming replies of an honest target peer, say P_x. After adding its own reply, P_x sends outgoing replies as the normal operation. When P_r receives all replies, it checks for P_j's reply. If P_j's reply is found, P_r understands that P_j is located in $P_x \ldots P_0$ range.
- Assume that P_{x+1} is P_r's collaborator. P_{x+1} can skip P_x and send replies to P_{x-1}. If P_r does not receive P_j's reply, it concludes that $P_x = P_j$. To perform this attack, all peers between P_{k-1} and P_x must be compromised. Otherwise, P_{x+1} and P_r can not decrypt replies of honest peers located between P_{k-1} and P_{x+1} due to the encryption layer with OR_x key.
- P_r injects many peers into the network [32] and tries to fill up a search range with own decoy peers. More decoys make the guessing of P_j easier. To mitigate this attack, the server can limit the number of peers that can be registered from an IP address. Thus, an adversary will need to compromise many computers to create

TABLE III

PERFORMANCE COMPARISON OF REPLY METHODS

	Phase 1	Phase 2	Total
Naive	$O(\log N)$	$O(\log N)$	$O(\log N)$
Random	$O(\log N)$	$O(\log N)$	$O(\log N)$
Oblivious	$O(\log N)$	$O((\log^2 N)/\eta)$	$O((\log^2 N)/\eta)$

decoy peers. Another approach is to ask a puzzle that a computer can not solve[34]. Thus, automated decoy peer creation can be slowed down.

- P_r may intercept all communication to some selected target peers, so only allowed target peers know about the query and replies. If P_r does not get P_j's reply after sending a trust query, one of the intercepted peers is P_j. By repeating this process, P_r can narrow down the candidates for P_j.

Long term tracing attacks. If P_r observes a search range for a long time and sends trust queries periodically, it may catch an instance when P_j is off-line. P_j can not answer a query off-line so P_r may guess P_j's identity. There is no complete solution to this attack since it is independent from security of the reply method. Even a protocol secure against some active adversaries [15] is vulnerable to this attack.

In a probabilistic approach, a larger search range might be selected using results of empirical studies [35], [36] about join and leave behavior of peers and their online periods. Thus, with a high probability, a search range may contain at least $O(k)$ offline and $O(k)$ online peers in any time interval. This can provide off-line/online k-anonymity for P_j.

Arranged query attacks. If the second phase operations are not designed carefully, P_r may figure out P_j using arranged trust queries. P_r sends a trust query directly to a target peer, say P_x. P_x forwards the query to its successor. The target peers located after P_x may think that the search range consists of $P_x \dots P_{k-1}$. They may assume P_x is the start of the search range, and thus, do not add encryption layers for target peers in $P_{x-1} \dots P_0$ range. P_x receives all replies and sends them to P_r. P_r decrypts the replies and looks for a reply from P_j. If there is a reply, P_r sends another query to a target peer between P_{k-1} and P_{x+1}. After sending several queries, P_r can reduce the number of possibilities for P_j.

To counter this attack, every target peer should know the boundaries of the search range very well. If a target peer is the first peer of a range (P_0), it can accept every trust query. Otherwise, a target peer should check the sender of a query by requesting $R_{BS}(TID_r|OU_r|TS)$ certificate and running a challenge/response protocol. If the sender of the query is its predecessor, it accepts the query. Otherwise, the query is rejected.

However, a collaborator in the search range may want to start a search query. For example, P_{x-1} may directly send a query to P_x. To prevent this attack, we have to force that a query should start from P_0. Every target peer adds a signature and its $R_{BS}(TID_r|OU_r|TS)$ certificate to the trust query. A target peer verifies the signatures of previous target peers before forwarding a query. For example, P_x checks the signatures of target peers in $P_{x-1} \dots P_0$ range. If a signature is missing, P_x drops the query.

E. Further Discussion

Performance of k-anonymity Chord. Let N is the number of peers in the network. Phase 1 (recursive Chord search) takes $O(\log N)$ time in all reply methods. Phase 2 (receiving replies from target peers) takes $O(k)$ time in the naive replying. For the random replying, phase 2 takes $O(k)$ time with a high probability [8].

In the oblivious replying method, more than one oblivious reply may be send in the same network packet. Assume that η is the number of replies that fit into a network packet. Efficient implementation of layered encryption and

compression can increase η. In the search range, $k(k-1)/2$ replies are transmitted. Thus, phase 2 takes up to $O(k^2/\eta)$ time in oblivious replying.

If $k = O(\log N)$, we can maintain a performance close to Chord. Asymptotic running times of reply methods are given in Table III when $k = O(\log N)$. By trading off the level of anonymity, our scheme has significant performance advantage comparing to $O(N^2)$ cost of the flooding approach [6].

Multiple trust holders. A trust holder may occasionally go off-line. Redundant trust holders increase availability of the trust information. Moreover, redundancy helps to prevent false replies of malicious trust holders. We assume that majority of a peer's trust holders are not malicious due to random selection. Thus, a false reply can be detected by doing majority selection. In case of detecting a malicious reply, a complaint can be sent to the bootstrap server about the malicious trust holder. Replies of all trust holders can be added to the complaint as an evidence. The server can figure out the malicious peer and select a new trust holder.

Sending trust holder certificates. In Section III-B, the bootstrap server sends a trust holder certificate to each trust holder (it is P_j in our scenario) in step 6. If a global passive adversary observes the bootstrap server during this messaging, it can easily learn the identity of a trust holder. Therefore, this step of registration should be done using k-anonymity Chord operations. In our case, the bootstrap server prepares a special message containing P_j's certificate, such as $MTID_j|TS'|TU_j(R_{BS}(H[ID_i]|TU_j|TS))|"Cert"$. Then, it sends this message to the trust network like sending a normal trust query. The last field in the message indicates that this message is not a regular trust query (The bootstrap server should be a member of the trust network like a normal peer to be able to send this message.). P_j read this message as in the oblivious replying protocol and get its certificate anonymously. Thus, an adversary can not learn P_j's identity.

Certificate renewal. A service certificate and related trust certificates expire according to the TS field. The owner of an expired service certificate requests a new one from the bootstrap server. The server reissues related certificates for the requester and its trust holders.

V. Future Work

Forcing semi-honest behavior. The semi-honest model might be a weaker assumption in case of collaboration. As explained in Section IV-D, a malicious target peer may forge its outgoing replies or skip next target peer without needing a special ability. These attacks decrease the number of candidates and makes guessing the trust holder easier. In order to prevent such attacks, a target peer must be forced to follow oblivious replying protocol. Goldreich [30] shows that semi-honest behavior can be forced by compiling each instruction (message). Thus, a semi-honest protocol can be extended against active adversaries. Compilation process requires commitment schemes and zero-knowledge protocols. This means that more CPU and network resources need to be used for each message sending operation. Ahn et al. [15] presents such an approach by using a secure multiparty sum protocol which relies on a commitment scheme and a zero-knowledge protocol. However, this approach requires sending of up to $O(b^4)$ bits for each b anonymous bits. In a future work, we are planning to force semi-honest behavior in our scheme with a better performance trade off.

Search range management. Peers in a search range should know each other to protect k-anonymity. How they securely learn about each other is a question to be studied. Handling dynamic changes on the trust overlay, determining search range size and managing (merging, splitting) search ranges under high churn are some issues. As explained in Section IV-D, a search range size should be large enough to guarantee online and offline anonymity with a high probability. Additionally, search range size might be adjusted according to the probability of that an adversary can inject a decoy peer in the search range [15].

Growing network. As more peers join the network, search ranges should shrink. The bootstrap server should adjust m to keep $k = O(\log N)$. Otherwise, a reply may create excessive network traffic in the second phase.

The server may recompute m and send new $MTID$ values during the certificate renewal operations. This might reveal information about trust holders. An adversary observing a search range for a long time might guess a trust holder after shrinking of the search range. Therefore, a complete rearrangement of trust holders might be needed. This process requires a study of how certificates are renewed and how new and old trust holders exchange trust information.

Excessive finger requests. If an adversary do not have global observation capability, it can still learn some information about a search range by sending many finger requests on Chord. This information can be used to launch long term observation attacks as explained in Section IV-D. Achord [3] defines some constraints for finger requests on Chord. A peer can not send a finger request to every peer. Achord can be adapted for our scheme to prevent a peer from learning all peers in a search range.

VI. Conclusion

k-anonymity Chord and oblivious replies provide us an efficient and anonymous access scheme. Trust holder anonymity is used as a case scenario to explain our scheme. Trust holders have k-anonymity protection against a global passive adversary. For collaborating adversaries, the scheme protects anonymity in semi-honest adversary model. The replies of trust holders are authenticated through an encryption scheme so fake anonymous replies are prevented. To guide the development of more secure systems, vulnerabilities of the scheme have been explained on several attack scenarios. Although our scheme is vulnerable to active attacks and long term tracing attacks, it provides a good trade off between performance and anonymity.

Our scheme can be extended to support requester anonymity. A group of peers create an anonymous request so identity of the requester is protected. Various anonymity requiring applications may adapt k-anonymity Chord for service provider/requester operations. Additionally, the ideas presented on this research can be used to design anonymity schemes on the other DHT structures such as CAN [13] and Tapestry [14].

VII. Acknowledgements

The authors thank to Mehmet Ercan Nergiz for his insightful comments about cryptography and adversary models.

References

[1] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 4, no. 2, 1981.

[2] 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*, 1997.

[3] S. Hazel and B. Wiley, "Achord: A variant of the chord lookup service for use in censorship resistant peer-to-peer publishing systems," in *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[4] I. Clarke, O. Sandberg, B. Wiley, and T. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *Proceedings of the First Privacy Enhancing Technologies Workshop (PET)*, Lecture Notes in Computer Science, Vol 2009, 2001.

[5] R. Dingledine, M. Freedman, and D. Molnar, "The free haven project: Distributed anonymous storage service," in *Proceedings of the First Privacy Enhancing Technologies Workshop (PET)*, Lecture Notes in Computer Science, Vol 2009, 2001.

[6] A. Singh and L. Liu, "Trustme: Anonymous management of trust relationships in decentralized p2p system," in *Proceedings of the 3rd IEEE Conference on Peer-to-Peer Computing (P2P)*, 2003.

[7] B. Zhu, S. Setia, and S. Jajodia, "Providing witness anonymity in peer-to-peer systems," in *Proceedings of the 13th ACM conference on Computer and Communications Security (CCS)*, 2006.

[8] M. Reiter and A. Rubin, "Crowds: Anonymity for web transactions," *ACM Transactions on Information and System Security*, vol. 1, no. 1, pp. 66–92, 1998.

[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.

[10] R. Sherwood, B. Bhattacharjee, and A. Srinivasan, "P5: A protocol for scalable anonymous communication," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2002.

[11] J. Ritter, "Why gnutella can't scale. no, really," in *http://www.darkridge.com/ jpr5/doc/gnutella.html*, 2001.

[12] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the ACM SIGCOMM*, 2001.

[13] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network," in *Proceedings of the ACM SIGCOMM*, 2001.

[14] B. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 41–53, 2004.

[15] L. von Ahn, A. Bortz, and N. J. Hopper, "k-anonymous message transmission," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS)*, 2003.

[16] K. Aberer and Z. Despotovic, "Managing trust in a peer-2-peer information system," in *Proceedings of the 10th International Conference on Information and knowledge management (CIKM)*, 2001.

[17] F. Cornelli, E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati, "Choosing reputable servents in a p2p network," in *Proceedings of the 11th World Wide Web Conference (WWW)*, 2002.

[18] S. Kamvar, M. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proceedings of the 12th World Wide Web Conference (WWW)*, 2003.

[19] C. Gülcü and G. Tsudik, "Mixing E-mail with Babel," in *Proceedings of the Network and Distributed Security Symposium (NDSS)*, 1996.

[20] D. Kesdogan, J. Egner, and R. Büschkes, "Stop-and-go MIXes: Providing probabilistic anonymity in an open system," in *Proceedings of the 2nd International Workshop on Information Hiding*, 1998.

[21] M. Jakobsson, "Flash Mixing," in *Proceedings of Principles of Distributed Computing (PODC)*, 1999.

[22] A. Serjantov, R. Dingledine, and P. Syverson, "From a trickle to a flood: Active attacks on several mix types," in *Proceedings of the 5th International Workshop on Information Hiding*, 2002.

[23] M. Rennhard and B. Plattner, "Practical anonymity for the masses with mix-networks," in *Proceedings of the 12th International Workshop on Enabling Technologies*, 2003.

[24] D. M. Goldschlag, M. G. Reed, and P. F. Syverson, "Hiding Routing Information," in *Proceedings of First International Workshop on Information Hiding*, 1996.

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

[26] 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.

[27] N. Borisov and J. Waddle, "Anonymity in structured peer-to-peer networks," Tech. Rep. UCB/CSD-05-1390, EECS Department, University of California, Berkeley, 2005.

[28] J. K. Kannan and M. Bansal, "Anonymity in chord," in *http://www.cs.berkeley.edu/ kjk/chord-anon.ps*, 2002.

[29] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, pp. 65–75, 1988.

[30] O. Goldreich, *Foundations of Cryptography*, vol. Basic Tools. Cambridge University Press, 2001.

[31] L. Sweeney, "k-anonymity: a model for protecting privacy," *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[32] J. Douceur, "The sybil attack," in *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[33] S. Goldwasser and S. Micali, "Probabilistic encryption & how to play mental poker keeping secret all partial information," in *Proceedings of the 14th annual ACM symposium on Theory of Computing*, 1982.

[34] T. Aura, P. Nikander, and J. Leiwo, "Dos-resistant authentication with client puzzles," *Lecture Notes in Computer Science*, vol. 2133, pp. 170+, 2001.

[35] S. Saroiu, P. Gummadi, and S. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proceedings of the Multimedia Computing and Networking*, 2002.

[36] S. Saroiu, K. Gummadi, R. Dunn, S. D. Gribble, and H. M. Levy, "An analysis of internet content delivery systems," in *Proceedings of the 5th USENIX Symposium on Operating Systems Design & Implementation (OSDI)*, 2002.