k-anonymity Chord for Anonymous Query Response

AHMET BURAK CAN
Hacettepe University
and
BHARAT BHARGAVA
Purdue University

Peer-to-peer systems provide a distributed way of sharing and storing information. Each peer stores some information and responds to queries. In some applications, protecting anonymity of a responding peer is important to increase availability of information. This paper presents a cryptographic protocol on Chord to protect anonymity of peers when responding to queries. In this protocol, peers form anonymity groups and generate responses inside groups. Responder of a query has k-anonymity protection against an adversary who can sniff all communication on the network. Validity of an anonymous reply can be verified so fake replies of malicious peers are prevented. The proposed approach can be adapted to other DHT structures to protect responder anonymity.

Keywords: Peer-to-peer systems, Responder Anonymity, Cryptography

1. INTRODUCTION

Peer-to-peer systems rely on collaboration of peers while performing system tasks, e.g., organizing, sharing, and searching resources. Responder anonymity is an important problem especially in publisher anonymity, and censor resistance applications [Waldman et al. 2000]. Peers sharing important resources are vulnerable to attacks of malicious peers. Protecting anonymity of a responder during a search query may mitigate some attacks. Hence anonymity can motivate a peer to perform information sharing task without worrying about identity reveal and can make information more available.

Probabilistic random path building, tunneling [Reiter and Rubin 1998; Freedman and Morris 2002; Mislove et al. 2004], flooding [Clarke et al. 2001; Dingledine et al. 2001; Singh and Liu 2003], limitations on routing information exchange [Hazel and Wiley 2002], and redundant queries [Nambiar and Wright 2006; Panchenko et al. 2009] are the most common methods to protect anonymity on peer-to-peer systems. Although these methods provides a protection against local adversaries, they are generally vulnerable to global passive adversaries who can sniff all communication on the network. Another common approach, mix networks [Chaum 1981] might be adapted to peer-to-peer systems. Trusted mix nodes encrypt and shuffle the network traffic so a global passive adversary can not easily determine communicating parties. However, in an ideal solution on peer-to-peer systems, trusted nodes should not involve in anonymity protocols, and peers should organize themselves to protect anonymity.

Our previous work on anonymity [Can and Bhargava 2010] introduced the oblivious reply protocol on Chord [Stoica et al. 2001] to protect anonymity of trust holders in a reputation-based trust model. This paper presents a general solution on Chord to protect anonymity of responders applicable in various peer-to-peer applications and discusses security properties of the proposed approach in cryptography perspective. We modify the search operation on Chord and call this modified distributed hash table (DHT) structure as *k-anonymity Chord*. Responders on this DHT structure have k-anonymity protection even adversaries sniff all network communication. During the search operation, the k-anonymity Chord runs the oblivious reply protocol to protect

Author's address: A.B. Can, Department of Computer Engineering, Hacettepe University, 06800 Ankara/Turkey
B. Bhargava, Department of Computer Science, Purdue University, West Lafayette, IN 47907 US

anonymity. In this protocol, peers form anonymity groups of size k. A query is sent to anonymity group instead of individual peers. After receiving a query, each peer in an anonymity group sends back a reply using the oblivious reply protocol. Even adversaries sniff all communication on the network, these replies can not distinguished from each other so the real responder has k-anonymity protection. Furthermore, initiator of a query can check authenticity of replies and identify fake replies of malicious peers.

Section 2 discusses related research. Section 3 introduces the general architecture and peer registration operations. Section 4 presents the k-anonymity Chord, analyzes security properties of the oblivious reply protocol, and provides a discussion about vulnerabilities and performance considerations. Section 5 outlines the results of this work.

2. RELATED WORK

Anonymity in communication networks has been studied by many researchers in different perspectives. Chaum [Chaum 1981] first proposed mix networks to protect anonymity of communicating parties for delay tolerant applications, e.g., e-mail systems. Trusted mix nodes use cover traffic to shuffle messages so an adversary can not determine who is communicating with whom. Chaum also proposed dining cryptographer networks [Chaum 1988], which provide unconditional sender anonymity in a group of participants. If the group size is N, this approach requires $O(N^2)$ message exchange for each message sending operation. Furthermore, before sending a message, $O(N^2)$ encryption keys should be distributed among N participants using a secure external method. This makes Chaum's DC-net impractical for real life scenarios.

Onion routers [Syverson et al. 1997; Goldschlag et al. 1996], form an overlay network to build anonymous, bi-directional virtual circuits for real-time communication such as HTTP. While mix networks are designed for delay tolerant applications, onion routing is more feasible for real-time applications. A widespread deployment of onion routing, Tor [Dingledine et al. 2004], extends onion routing with forward secrecy, congestion control, integrity checking, and configurable exit policies. Although it's high traffic overhead issues [Dingledine and Murdoch 2009; Wacek et al. 2013; AlSabah et al. 2013], Tor is widely used by research community. Tor uses directory servers to maintain onion router topology. Nodes learn whole topology from directory servers to select random relays while building a circuit (anonymization path). NISAN [Panchenko et al. 2009] and Torsk [McLachlan et al. 2009] use DHTs to solve scalability issues of Tor networks due to topology maintenance costs. In both approaches, random relays to build a Tor circuit are selected using DHTs. NISAN uses redundant independent lookups and bounds checking to mitigate some active attacks to reveal relays. In Torsk, random walks are used to select secret buddies of lookup initiators so secret buddies can hide the identity of initiators. However, Wang et al. [Wang et al. 2010] present that both NISAN and Torsk are vulnerable some passive and active attacks. To protect relay nodes from attacks, Mittal et al. [Mittal et al. 2011] proposed a privacy preserving information retrieval method. Resilience of anonymity networks against denial of service attacks is also studied by researchers [Danner et al. 2012; Elahi et al. 2012; Barbera et al. 2013; Das and Borisov 2013]. Since selection of relay nodes is an important issue to mitigate attacks, some researchers [Wang et al. 2013; Das et al. 2014; Akavipat et al. 2014] proposed reputation-based approaches to select reliable nodes and collaboratively filter malicious peers. Shirazi et al. [Shirazi et al. 2013] proposed a metric to measure resilience of anonymity networks.

Besides mix networks and onion routing, another interesting anonymity approach, the buses [Beimel and Dolev 2003; Ren et al. 2008], traverse synchronous message tokens in the network continuously. When a sender receives a bus, it fills some seats with encrypted messages. If the sender does not have any real message, it puts encrypted dummy messages. When a bus arrives to a receiver, all or related seats are decrypted to understand if there is a message. Although this approach can protect sender and receiver anonymity, a bus should travel in a network forever and nodes should produce dummy messages when they don't have any real message.

In peer-to-peer systems, random path building, tunneling, flooding, and limitations on routing information exchange are the most studied approaches to protect anonymity. In Crowds [Reiter and Rubin 1998], nodes form anonymity groups (crowds) to protect requester anonymity. A request started in a crowd is randomly forwarded in the crowd several times and finally sent to the outside world by a node, which also receives the response from the outside world. In peer-to-peer storage systems, Freenet [Clarke et al. 2001] and Freehaven [Dingledine et al. 2001] flood storage requests to protect requester and responder anonymity. Since no peer on the path between the requester and responder knows the whole query path, it is hard to determine communicating parties. Tarzan [Freedman and Morris 2002] establishes a random tunnel in a peer-to-peer network between a peer and an Internet server. Since no peer on a tunnel knows the whole path, a query initiating peer can have anonymity. Like Tarzan, MorhpMix [Rennhard and Plattner 2002] defines a peer-to-peer mix network with a collusion detection mechanism. Randomly selected peers behave as mix nodes and build a circuit to protect anonymity. AP3 [Mislove et al. 2004] also use a random tunnel building approach and a multicast group approach to protect publisher anonymity. Some approaches use secret sharing schemes and random path building together to protect anonymity. Publius [Waldman et al. 2000] uses a secret sharing scheme to protect publisher anonymity. Shares of a master key are distributed among several nodes so nodes sharing a file encrypted with the master key are not accountable for what they stored. Han and Liu [Han and Liu 2008] split a query into n shares and send them to neighbors in a mobile peer-to-peer network. Peers who take t shares can decrypt and flood the query. The responder builds an onion path to the requester and sends a response on this path. Another study, Salsa [Nambiar and Wright 2006], uses a DHT to create anonymous circuits. This DHT defines a secure lookup mechanism with redundancy and bounds checking, which allows to select random relays on a circuit anonymously. Although each node knows a part of the network, relays are selected from all peers. Shadowwalker [Mittal and Borisov 2009] introduces shadow nodes, which enable construction of secure virtual circuits to protect anonymity.

Although most approaches in peer-to-peer systems protect anonymity for unstructured networks, there are few approaches to protect anonymity on structured overlay networks, a.k.a. distributed has tables(DHTs). Since DHTs provide efficient access to information, anonymity on DHT structures can find many applications [Jahid et al. 2012; Fabian and Feldhaus 2014]. However, ensuring anonymity on DHTs is a difficult problem [Mittal and Borisov 2008; Tran et al. 2009]. In one of the first approaches, Hazel and Wiley [Hazel and Wiley 2002] use routing limitations as a way of protecting anonymity on Chord [Stoica et al. 2001]. Borisov and Waddle [Borisov and Waddle 2005] use recursive, randomized, indirect, split, bidirectional routing on Chord. Anonymity on these approaches is not dependent to cryptographic properties and thus vulnerable against adversaries who can observe the whole anonymity path or network. Kondo et al. [Kondo et al. 2009] introduce node management and anonymous communication layer to protect anonymity on Chord. Anonymity layer implements a protocol similar to onion routing and enables sender/receiver anonymity by using encrypted traffic. Wang and Borisov [Wang and Borisov 2012] also proposed an approach by splitting queries, launching dummy queries, and removing malicious peers with an attacker identification mechanism.

Most of the above approaches in peer-to-peer systems try to protect anonymity on a large scale network model and assume that an adversary can not observe the whole path on a circuit or flooding path. Methods like random relay selection, redundant lookups, limiting knowledge about the network, and changing routing methods can mitigate attacks of an adversary who can observe only a part of the path. If an adversary can observe the whole path, these methods may not protect anonymity. An adversary who can sniff the whole query and response path can determine the communicating parties. Additionally, excessive network traffic caused by flooding or maintaining information about relay nodes may reduce scalability of these systems.

In our approach, k-anonymity of a responder is protected even an adversary sniffs all network. To achieve this, search operation of Chord is modified. However, anonymity is protected through cryptographic properties, rather than changing network structure or routing operation. Our

Table I. Preliminary notations

Notation	Description
U_{bp}, R_{bp}	the bootstrap peer's public/private key pair
P_i	the i^{th} peer
AID_i	P_i's pseudonym
AU_i, AR_i OU_i, OR_i	P_i's public/private key pairs used in anonymity operations
$K(M)$	encryption of M if K is a public/symmetric key
$K\{M\}$	signing of M if K is a private key
$H[M]$	hash digest of M
$X\|Y$	concatenation of X and Y

approach tries to protect anonymity among a small group of peers instead of a large network. Therefore, it should not be directly compared with approaches like Tor, AP3, Salsa, NISAN, Torsk, etc. Our approach should be considered as a way of enhancing anonymity of responders on DHT structures and can be adapted on various DHT structures.

3. ARCHITECTURE

Every peer-to-peer network needs a connection point to the network. Even in a complete peer-to-peer architecture like Gnutella [Gnutella], some bootstrap peers are needed to connect new peers to the network. As in other peer-to-peer approaches, we assume that a *bootstrap peer* (*bp*) provides a connection point to the network for new peers. The bootstrap peer is a basic certification authority for pseudonyms and encryption keys. U_{bp} and R_{bp} are the public and private keys of the bootstrap peer. We assume all peers learn U_{bp} in a secure way, such as using a secure web site or digital certificates. There might be multiple bootstrap peers to provide tolerance to failures. For simplicity, the paper considers one bootstrap peer.

Let P_i be the i^{th} peer. In peer registration, P_i is assigned to a pseudonym, AID_i, which is P_i's pseudonym. AID_i is selected by the bootstrap peer so an adversary can not easily set up multiple peers within a specific range of pseudonyms and launch coordinated attacks in the network [Douceur 2002]. For anonymity operations, P_i randomly selects $\{AU_i, AR_i\}$ and $\{OU_i, OR_i\}$ public key pairs. These key pairs have no relation with each other. We assume that peers have good pseudo-random number generators to prevent brute force guessing attacks.

We give some notations to describe message formats. Assuming K is a public key or a symmetric encryption key, $K(M)$ denotes encryption of M with key K for message confidentiality. If K is a private key, $K\{M\}$ denotes signing of M. Assuming H is a hash function, $H[M]$ denotes hash digest of M. If X and Y are two messages, $X|Y$ denotes concatenation of X and Y. Table I lists these notations for easy reading of the following sections.

Adversary model. An adversary tries to learn the responder of a query. It[1] has polynomial time computational capabilities and can not break cryptographic algorithms in polynomial time. It has global passive attack capabilities, i.e., it can sniff all network communication. Note that, even an adversary may not sniff the whole network by itself, it may collaborate with some peers and launch passive attacks with them to obtain global passive attack capabilities. If the adversary involves in a anonymous query, it behaves in semi-honest adversary model [Goldreich 2001]. In this adversary model, an adversary obeys the rules of the running protocol but passively observes network communication to obtain information.

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

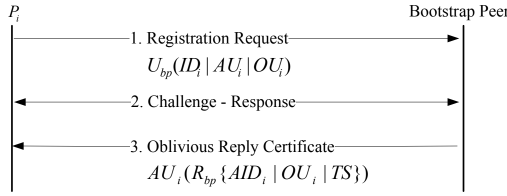


Figure 1. Registration of P_i to the bootstrap peer

3.1 Peer registration

Each peer must register itself to the bootstrap peer when joining the network for the first time. During the registration, the bootstrap peer selects an AID value and issues a certificate for the new peer. Assuming P_i is registering itself, the following registration steps are done as shown in Figure 1:

(1) P_i sends the bootstrap peer a registration request containing AU_i, OU_i values. The request is encrypted with U_{bp} so only the bootstrap peer can read its content.

(2) The bootstrap peer runs a challenge-response protocol to ensure that P_i has the corresponding private keys, AR_i, OR_i. After ensuring that P_i is the owner of AR_i, OR_i keys, the bootstrap peer saves AU_i, OU_i public keys for future accountability. We do not limit our design to any specific challenge-response protocol. A well-known public-key based protocol can be used [Boyd and Mathuria 2003] for this step.

(3) The bootstrap peer selects an AID_i value either randomly or in a way to maintain uniform distribution in the network. The bootstrap peer sends P_i an *oblivious reply certificate*, $R_{bp}\{AID_i|OU_i|TS\}$. The certificate is encrypted with AU_i for confidentiality. P_i decrypts this message and stores the certificate.

The certificate informs P_i about its AID_i and is used in the oblivious reply protocol. P_i or another peer can not forge fake certificates without having R_{bp} key. The certificate expires based on a timestamp field, TS. In case of expiration, P_i can request a new certificate from the bootstrap peer, in a similar way to the registration operation.

Figure 1 briefly explains the peer registration. At first, P_i sends a registration request to the bootstrap peer (step 1). Then, the boostrap peer runs a challenge-response protocol to validate that P_i has the private keys, AR_i, OR_i (step 2). After passing this validation step, the bootstrap peer sends the oblivious reply certificate to P_i (step 3). The registration operation may contain more implementation details. We omit such details to focus on our problem.

4. K-ANONYMITY CHORD

Chord [Stoica et al. 2001] is a distributed hash table for peer-to-peer networks. It assigns each resource to a particular peer. Search operations (*queries*) on Chord locate resources in $O(\lg N)$ time, where N is the number of peers in a network. Chord can provide an efficient way of storing and accessing information on peer-to-peer networks but can not protect anonymity of a peer when responding to a query. Peers on a Chord ring partially know the network structure and may guess the responder. A malicious peer may learn more about Chord's address space by sending excessive finger requests [Hazel and Wiley 2002], which makes guessing a responder easier without having global sniffing capabilities.

We propose the oblivious reply protocol to provide k-anonymity protection on Chord [Sweeney 2002] against global passive attacks. A responder's identity can not be distinguished from k other peers when responding to queries. We call this DHT structure *k-anonymity Chord*, which performs peer join, leave, and finger table maintenance operations like a normal Chord ring. However, search operation is modified to protect anonymity of responders.

4.1 Formation of anonymity groups

A peer joins the k-anonymity Chord with its pseudonym, i.e., P_j joins with AID_j value. The pseudonym determines the location of P_j on the Chord ring. Peers form anonymity groups on this ring based on their $MAID$ values. For P_j, $MAID_j$ is a masked value of AID_j where the last m bits are set to zero. Peers in $MAID_j$ and $MAID_j + 2^m - 1$ range form the *anonymity group* of P_j. The peers in an anonymity group are called as *target peers*. The target peers in an anonymity group have the same $MAID$ value, which can be considered as an identity number of the group. The bootstrap peer should decide an m value so that the expected number of target peers in an anonymity group is equal to or greater than k. Since the bootstrap peer registers all peers and determine their positions in the network, it can compute a precise m value. Assuming the bootstrap peer uniformly distributes peers on the Chord ring, $MAID_j$ can be computed as follows:

Chord allocates peers on a 2^n circular address space where n is the length of AID values in bits. Suppose that $n = 32, k = 64, AID_j = 12345678H$ and there are $N = 2^{16}$ peers in the network. Let X_i be an indicator random variable, which represents if there is a peer on a particular location i (When $X_i = 1$, there is a peer on the location i). The probability of $X_i = 1$ is

$$P(X_i = 1) = \frac{N}{2^n} = \frac{2^{16}}{2^{32}} = \frac{1}{2^{16}}$$

and the expected number of nodes on a particular location is

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

Let S be the number of total locations (addresses) in an anonymity group and Y be a random variable representing the number of peers in the group. Assuming uniform distribution of peers, the expected number of peers in an anonymity group is

$$E[Y] = \sum_{i=MAID_j}^{MAID_j+S} E[X_i] = S \cdot \frac{1}{2^{16}}$$

In a peer-to-peer network, peers frequently go offline/online. Therefore, number of online peers in an anonymity group changes with time. Let Z be a random variable representing the number of online peers in an anonymity group. Assuming at least 25% of all peers are online in any time period, the expected number of online peers in an anonymity group is

$$E[Z] \geq S \cdot \frac{1}{2^{16}} \cdot \frac{1}{4} = \frac{S}{2^{18}}$$

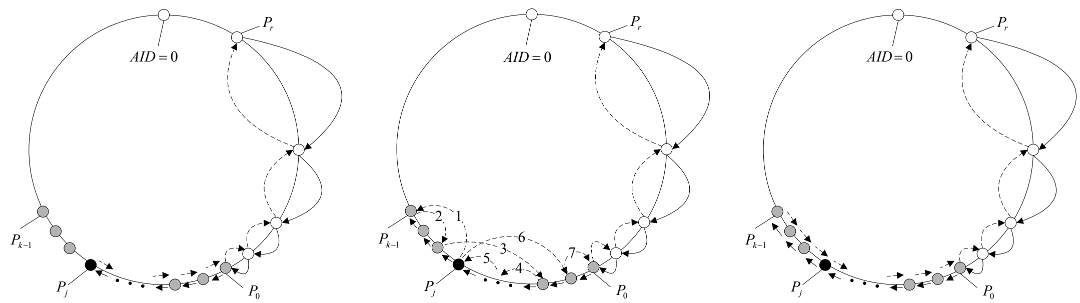
$E[Z]$ should be greater than or equal to $k = 64$. Thus, the bootstrap peer finds that $S \geq 2^{24}$. This inequality suggests us that $m = \log_2 S \geq 24$. Then, the bootstrap peer sets at least the last 24 bits of AID_j to zero and computes $MAID_j$ as follows:

$$\begin{aligned} MAID_j &= 12345678H \wedge 0FF000000H \\ &= 12000000H \end{aligned}$$

$MAID_j = 12000000H$ means that P_j has an AID_j value between $12000000H$ and $12FFFFFFH$. The expected number of online peers in this range is at least 64 due to our selection.

4.2 Query operation on k-anonymity Chord

Let $P_0, P_1, \ldots, P_{k-1}$ be the target peers in $MAID_j$ and $MAID_j + 2^m$ range, in other words, the anonymity group of P_j. By the design of k-anonymity Chord, we know that P_0 is the owner of $MAID_j$ value. Assume that P_r starts a query to get an anonymous response from P_j. To start this query, P_r needs to know $MAID_j$ value, ID of the requested resource, and AU_j key



(a) With naive reply (b) With random-route reply (c) With oblivious reply

Figure 2. Routing of a query on k-anonymity Chord and various reply protocol options

of P_j. How to obtain this information is dependent to the application. If the aim is to protect anonymity of a responder which provides a handle for a file in a peer-to-peer storage system, an index peer [Napster] or a web site [Bittorent] can provide the necessary information for a query. ID value should be a descriptor value (such as name) that uniquely identifies the requested file. If the aim is to protect anonymity of trust holders in a reputation-based trust model [Aberer and Despotovic 2001; Kamvar et al. 2003], P_r can obtain the necessary information as a signed certificate from another peer[Can and Bhargava 2010].

After getting $MAID_j, AU_j, ID$ values, P_r can send an anonymous query to P_j's anonymity group. As an anonymous query, P_r sends $MAID_j|TS'|AU_j(K_{rj}|TS'|ID)$ to the network. $MAID_j$ represents the destination of this query, P_j's anonymity group. TS' is a time-stamp to guarantee uniqueness and freshness of the query. K_{rj} is a random session key created by P_r. Due to the encryption with AU_j, only P_j can read the content of $AU_j(K_{rj}|TS'|ID)$ part.

4.3 Routing a query

k-anonymity Chord defines a two-phase routing method for a query. The first phase is a recursive Chord search to find P_0, who is the owner of $MAID_j$ value according to Chord's algorithm. P_r starts the first phase by sending a query, $MAID_j|TS'|AU_j(K_{rj}|TS'|ID)$, to the closest peer preceding P_0 according to its Chord finger table. The receiving peer forwards the query to another one by looking up $MAID_j$ value in its finger table. Forwarding operation continues until P_0 receives the query. Each forwarding peer caches the query for a period of time so P_j's reply can be send back to P_r later.

After the query reaches to P_0, the second phase is started to get P_j's reply among target peers. In the following sections, we present three methods for the second phase of the query. The first two methods are vulnerable to global sniffing attacks. These methods are discussed to clarify our contribution. Our method, the oblivious reply protocol, protects anonymity against global passive adversaries. In the attack scenarios, we assume that P_r or another peer tries to identify the responder, P_j.

4.4 Naive reply

After receiving a query, P_0 tries to decrypt $AU_j(K_{rj}|TS'|ID)$ part. If AU_j key does not match with P_0's key, P_0 can not identify TS', ID values. Since the query is for another target peer, 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 operation continues until P_j receives the query[2]. P_j sends a reply, $MAID_j|TS'|K_{rj}(M|ID|TS')$, back to its predecessor. M is the response message of P_j to the query. Each target peer sends the reply back to its predecessors until P_0 receives it. P_0 sends the reply back to the previous peer on the query path. All peers on the path repeat the same operation until P_r receives the reply.

[2]If P_j is not online, the query reaches to the last target peer, P_{k-1}, and the query is dropped.

P_r checks encrypted ID and TS' values in the reply. If values are correct, M is an authentic response from P_j. Figure 2(a) shows two-phase routing of a query with the naive reply. Points on the Chord ring represent the peers involved in routing of P_r's query. Gray points represent the target peers. The black point is P_j. Solid arrows represent the path of P_r's query. Dashed arrows represent the path of P_j's reply.

The encryption scheme protects authenticity of a reply. A malicious target peer can not obtain K_{rj} key and forge a fake reply. However, if P_r can sniff all network links or just P_j's links, it can observe that P_j neither forwarded the query nor received a reply from its successor. Thus, P_r can identify that P_j is the responder.

4.5 Random-route reply

After P_0 receives a query, all target peers forward the query to their successors until P_{k-1} receives it. P_j decrypts the query and waits for while to ensure that P_{k-1} receives the query. Then, P_j prepares a reply as in the naive reply method and sends it to a random target peer (assuming all target peers in the anonymity group know each other). Receiving peer randomly forwards the reply to another one with a probability of p_f. A peer may receive and forward the same reply several times. Finally, a peer decides to send the reply to P_0. Peers on the query path between P_0 and P_r forward the reply until P_r receives it. Figure 2(b) shows two-phase routing of a query with random-route reply.

If P_r only has local observation capability, random-route reply may provide a probabilistic anonymity protection for P_j but does not eliminate chance of being identified. A forwarding peer only knows its preceding hop, but can not ensure if the preceding hop is P_j. In case of a collaboration between P_r and some target peers, P_j has probable innocence [Reiter and Rubin 1998] if

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

k is the number of target peers and c is the number of P_i's collaborators. The probable innocence means that a peer is no more likely to be the responder than not to be the responder. If $c > \frac{k}{2} - 1$, there is no probable innocence for P_j. In other words, random-route reply does not protect anonymity when half of the target peers are compromised. In this analysis, adversaries are assumed to have local sniffing capabilities. If P_r has global sniffing capabilities, anonymity of P_j can not be protected.

4.6 The oblivious reply protocol

Naive and random reply protocols are vulnerable global passive adversaries. The oblivious reply protocol protects anonymity of a responder against global passive adversaries. The basic idea is that each target peer generates a separate reply for a query. However, a reply can not be linked with its sender. Thus P_j's reply can not be traced while replies are being collected from target peers. We have several assumptions for the protocol:

(1) Peers have good pseudo-random number generators.

(2) Each target peer knows other target peers in its anonymity group and its exact location in the group, i.e, the number of hops from P_0 and P_{k-1}.

(3) All target peers exchange their oblivious reply certificates, i.e., P_j exchanges $R_{bp}\{AID_j|OU_j|TS\}$ with others. Once certificates are exchanged, they can be used in many queries.

(4) The public key encryption scheme is not commutative, i.e., $A(B(M)) \neq B(A(M))$. Additionally, the public key encryption scheme satisfies semantic security [Goldwasser and Micali 1982]. This implies that result of an encryption depends on the message and a sequence of coin tosses. Thus, encryption of a plaintext with the same public key results in a different ciphertext in each trial. However, decryptions of these ciphertexts give the same plaintext.

As shown in Figure 2(c), when P_r's query reaches to P_0, each target peer forwards the query until P_{k-1} receives it. P_{k-1} tries to decrypt contents of the query. If the decryption is successful, it prepares O_{k-2}^{k-1} as follows:

$$
\begin{aligned}
O_{k-2}^{k-1} &= OU_{k-2}(O_{k-3}^{k-1}) \\
O_{k-3}^{k-1} &= OU_{k-3}(O_{k-4}^{k-1}) \\
&\dots \\
O_{1}^{k-1} &= OU_{1}(O_{0}^{k-1}) \\
O_{0}^{k-1} &= OU_{0}(K_{rj}(M|ID|TS')|AB)
\end{aligned}
$$

O_{k-2}^{k-1} denotes P_{k-1}'s *oblivious reply* to be delivered to P_{k-2}. The innermost encryption layer contains P_{k-1}'s response, M. The authenticity bit, AB, is set to 1 in order to show that the reply is authentic.

If the decryption of the query fails, P_{k-1} generates a false oblivious reply and sets $AB = 0$ to indicate that the reply is inauthentic. Then, the innermost layer of O_{k-2}^{k-1} contains $K_{rnd}(RM|RID|TS')|AB$ as the content. K_{rnd} is a randomly generated key. RM and RID are random response message and ID values respectively. These random values should have the same amount of bits as the authentic values. Due to the layered encryption, only P_0 can read AB field. Therefore, regardless of its content, P_{k-1}'s oblivious reply looks same for other peers. For the rest of paper, we will use "reply" and "oblivious reply" terms interchangeably. The protocol runs as follows:

(1) P_{k-1} sends $MAID_j|TS'|O_{k-2}^{k-1}$ to its predecessor, P_{k-2}.

(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 $MAID_j|TS'|(O_{k-3}^{k-1} \cup O_{k-3}^{k-2})$ to P_{k-3}. The operation $\cup$ denotes the concatenation in 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 randomization of their order.

(3) P_{k-3} decrypts the top layers of O_{k-3}^{k-1} and O_{k-3}^{k-2}. It creates O_{k-4}^{k-3} and sends $MAID_j|TS'|(O_{k-4}^{k-1} \cup O_{k-4}^{k-2} \cup O_{k-4}^{k-3})$ to P_{k-4}.

(4) All target peers repeat this operation until P_0 receives $MAID_j|TS'|(O_0^{k-1} \cup O_0^{k-2} \cup \ldots \cup O_0^2 \cup O_0^1)$. After decrypting the last layers of all replies, P_0 checks AB fields and determines the authentic reply. P_0 sends this reply to the previous peer on P_r's query path. All peers on the path repeat the same operation until P_r receives the reply. If there are multiple replies with $AB = 1$, all of them are sent to P_r since only P_r can determine the authentic reply.

(5) P_r decrypts all incoming replies using K_{rj}. The reply with the correct ID and TS' values is the authentic one. A malicious peer can not forge an authentic reply since it can not obtain K_{rj}.

The protocol uses layered encryption concept of mix networks and onion routers in a different way. For a better understanding of the encryption scheme, the reader may refer to [Chaum 1981; Goldschlag et al. 1996]. Figure 3 shows operation of the protocol among target peers. All replies are accumulated from P_{k-1} to P_0. At the end, P_0 receives $k-1$ replies and sends some replies to P_r.

4.7 Security analysis of the oblivious reply protocol

If P_r is a global passive adversary, it can observe all communication among target peers but layered encryption of replies, identical reply sizes, randomization of replies on each target peer, and semantic security assumption do not allow P_r to identify P_j's reply. The oblivious reply protocol provides k-anonymity protection for responders as long as adversaries perform passive attacks. We demonstrate this as follows:

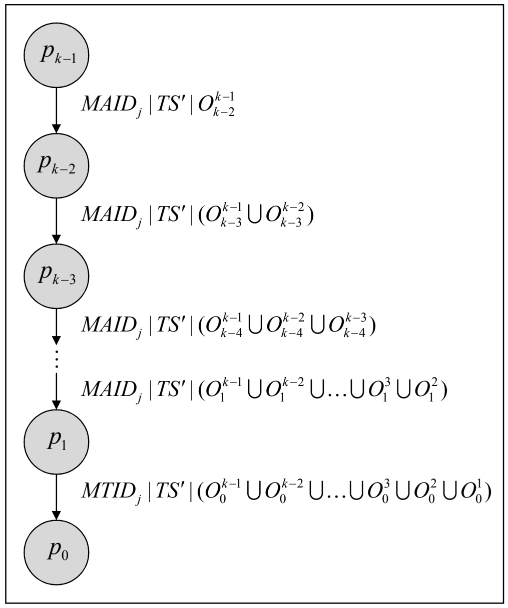


Figure 3. Message communication among target peers in the oblivious reply protocol

(1) *P_r can not learn any information about the sender of an oblivious reply by sniffing incoming and outgoing replies of a target peer.*

Let $P_x \in \{P_0, \ldots, P_{k-1}\}$ be an honest target peer and P_r be capable of sniffing P_x's communication. When P_{x+1} sends $MTID_j|TS'|(O_x^{k-1} \cup O_x^{k-2} \cup \ldots \cup O_x^{x+1})$ to P_x, P_r can sniff these replies. However, P_r can not learn contents of $O_x^{k-1}, O_x^{k-2}, \ldots, O_x^{x+1}$ due to layered encryption with $OR_x, \ldots, OR_0$ keys. This condition is also true for P_x's outgoing replies.

P_r may try to link P_x's incoming and outgoing replies. If it can succeed on this, P_r may try trace a reply by sniffing all target peers' incoming and outgoing replies. However, this is not possible for two reasons:

(a) Layered encryption of oblivious replies, randomization of their order by P_x, and identical reply sizes do not allow P_r to distinguish a particular outgoing reply from others. All replies look same for P_r.

(b) When P_x exchanges its oblivious reply certificate, $R_{bp}\{AID_x|OU_x|TS\}$, with other target peers, P_r can learn OU_x by sniffing messages. It can encrypt P_x's outgoing replies with OU_x and try to obtain an incoming reply. However, P_r can not do this due to our semantic security assumption. None of the outgoing replies can be linked to an incoming reply without knowing OR_x key.

Let P_r be capable of sniffing P_0's communication. P_r can decrypt P_0's outgoing oblivious replies since it is encrypted with K_{rj}. However, P_r can not find a link between P_0's outgoing and incoming replies due to the reasons explained above. P_r must have OR_0 key to establish such a link.

Therefore, sniffing a peer's incoming and outgoing replies does not provide any information to adversaries.

(2) *If P_r has global observation capability, the oblivious reply protocol provides k-anonymity protection for P_j.*

P_r can observe all messages among $P_0, P_1, \ldots, P_{k-1}$ but it can not obtain any information due to above arguments. P_r can not distinguish P_j's reply from other $k-1$ replies so P_j has k-anonymity protection.

P_r may obtain some collaborators among target peers by compromising them or injecting decoy peers into them (Sybil attack [Douceur 2002]). With the help of these collaborators, P_r may try to track replies and identify P_j's reply. We claim that if collaborators behave in semi-honest adversary model [Goldreich 2001], the protocol protects P_j's anonymity. Assuming P_r is a global passive adversary, we demonstrate this as follows:

(3) *If all except two target peers are P_r's collaborators, replies of two honest target peers can not be distinguished from each other as long as collaborators behave in semi-honest adversary model.*
Without loss of generality, let P_x and P_y be honest target peers where $0 < x < k-2$ and $x+1 < y \leq k-1$. All other peers are collaborators of P_r based on our assumption. As a collaborator, P_{x+1} can identify P_y's reply, O^y_{x+1}, since all peers between P_{k-1} and P_{x+1} except P_y are collaborators. P_{x+1} decrypts top layers of its incoming replies, adds its own reply, and sends $MAID_j|TS'|(O^{k-1}_x \cup \ldots \cup O^{x+2}_x \cup O^{x+1}_x)$ to P_x.
P_x repeats the same operations and sends $MAID_j|TS'|(O^{k-1}_{x-1} \cup O^{k-2}_{x-1} \cup \ldots \cup O^{x+2}_{x-1} \cup O^{x+1}_{x-1} \cup O^{x}_{x-1})$ to P_{x-1}. Knowing oblivious replies of all collaborators, P_{x-1} can figure out O^x_{x-1} and O^y_{x-1}. It can also learn O^y_x from P_{x+1}. To identify senders of O^x_{x-1} and O^y_{x-1}, P_{x-1} has to find a link between one of these replies and O^y_x. However, P_{x-1} can not do this as discussed in listing 1. In P_{x-1}'s view, O^x_{x-1} and O^y_{x-1} are equally likely to be decryption of O^y_x. Without having OR_x key, neither P_r nor other collaborators can learn any further information than P_{x+1} and P_{x-1}. Therefore, P_r and its collaborators can not distinguish replies of P_x and P_y. If P_x, P_y are two consecutive peers $(x+1=y)$, collaborators still can not distinguish replies of two honest peers. When P_y sends its reply to P_x, collaborators may identify O^y_x. However, without having OR_x key, they can not establish a link between O^y_x and P_x's outgoing replies, O^x_{x-1}, O^y_{x-1}.

(4) *If m target peers are collaborators of P_r, the oblivious reply protocol provides $k-m$ anonymity protection for P_j as long as collaborators behave in semi-honest adversary model.*
With the above arguments, if there are $k-m$ honest target peers, adversaries can not distinguish $k-m$ honest replies from each other in the semi-honest adversary model. P_j has $k-m$ anonymity protection since it can not be linked with any of $k-m$ replies.

We conclude that the oblivious reply protocol provides k-anonymity protection for responders as long as adversaries perform only passive attacks.

4.8 Vulnerabilities of the oblivious reply protocol

A passive attack to the oblivious reply protocol is possible due to open nature of peer-to-peer systems. A global passive adversary may observe an anonymity group for a long time to catch a responder's offline period. It can periodically send queries. If no reply comes back, the adversary understands that responder is offline and may guess its identity. High churn nature of peer-to-peer systems makes this attack possible. Even a perfect anonymity providing system is vulnerable to this attack since peers intermittently join and leave the network. A possible approach to mitigate this attack is making anonymity groups large enough to provide responders offline anonymity. When calculating $MAID$ values in Section 4.1, we consider this situation and form anonymity groups by assuming that nearly 25% of peers in an anonymity group are online in any time period. If P_j is offline during a query, 75% of target peers in its group will be offline on average. Thus, determining P_j's identity will not be easy. Statistical results of empirical analysis [Ripeanu et al. 2002; Saroiu et al. 2002] can be used to guess churn rate and determine appropriate group sizes more accurately.

The oblivious reply protocol can not protect anonymity if adversaries perform active attacks. We identify several active attacks that can be launched by P_r or its collaborators. Assuming P_x is an honest target peer, the following attacks may break P_j's anonymity:

(1) *Forging replies*: Let P_{x+1} be a collaborator and one of target peers, where $0 < x+1 < k-1$.

P_{x+1} can ignore its incoming replies and send fake replies to P_x. Since peers in $P_x, \ldots, P_0$ range can not understand this situation, they perform normal operation. At the end of protocol, if P_r receives P_j's reply, P_j is located in $P_x, \ldots, P_0$ range. Otherwise, P_j is probably in $P_{k-1}, \ldots, P_{x+2}$ range[3]. If P_r has several collaborators, the attack can be repeated to reduce the number of possible target peers for P_j.

If P_r has global active attack capability, it can perform the same attack by forging P_x's incoming replies. If P_r receives P_j's reply, P_j is located in $P_x, \ldots, P_0$ range. According to the result, P_r can repeat the attack on various target peers and can reduce the number of possibilities.

(2) *Dropping selected replies*: Assume that P_{x+1} is a collaborator and one of target peers, where $0 < x + 1 < k - 1$ and there are some other collaborators in $P_{k-1}, \ldots, P_{x+2}$ range. P_{x+1} can identify replies of honest peers in its incoming replies by communicating other collaborators. If P_{x+1} drops all honest replies, P_x can understand this situation (By our assumption, P_x knows its exact location in the anonymity group and the number of replies that it should receive from P_{x+1}.). Therefore, P_{x+1} sends P_x forged fake replies instead of dropped ones. If P_r does not receive P_j's reply, P_j is probably one of the honest peers in $P_{k-1}, \ldots, P_{x+2}$ range. Otherwise, P_j is in $P_x \ldots P_0$ range.

(3) *Skipping a peer*: Let $P_{k-1}, \ldots, P_{x+1}$ be collaborators. P_{x+1} forges fake replies and sends them directly to P_{x-1} so it skips P_x. If P_r does not receive P_j's reply, P_x is probably P_j. This attack can succeed only if $P_{k-1}, \ldots, P_{x+1}$ are all collaborators.

(4) *Isolating a peer*: If P_r has global active attack capability, it may intercept all communication to P_x. Other target peers think that P_x is offline and do not send P_x any message. P_r sends a query and waits for P_j's reply. If P_r does not receive an authentic reply, P_x is probably P_j. By repeating this process for other target peers, P_r can narrow down candidates for P_j.

In addition to above attacks, a collaborator may also drop all queries and replies passing through it. Although such an attack does not give any information about P_j's identity, it can be considered as a denial of service attack. Such adversaries can be reported to the bootstrap peer and excluded from the query and reply operations. We are considering such attacks since it is out of our discussion.

If a target peer is forced to stay complaint with the rules of oblivious reply protocol, some active attacks can be prevented. Goldreich [Goldreich 2001] explains that semi-honest behavior can be forced by compiling each instruction (message), which might be the next step of this study.

4.9 Performance Considerations

We consider message complexity and computational complexity to evaluate performance of our protocol. In the oblivious reply protocol, a reply is forwarded up to $\Theta(k)$ times. For k replies, $\Theta(k^2)$ network packets are forwarded in phase 2. However, more than one reply can be sent in the same network packet for efficiency. While oblivious replies are getting closer to P_0, size of an oblivious reply decreases and number of replies that fit into a network packet increases. Therefore, number of network packets does not increase as quick as the number of replies while replies are forwarded from P_{k-1} to P_0. Efficient implementation of layered encryption and compression may also decrease reply sizes and improve the performance.

During an anonymous query, each peer makes $\Theta(k)$ public-key decryptions while decrypting top layers of its incoming replies and $\Theta(k)$ public-key encryptions while creating its own reply. For each run of the protocol, $\Theta(k^2)$ public-key encryptions and decryptions are performed in total. This can be acceptable comparing to multiparty computation approaches [Goldreich 2001].

[3] P_j might be offline at the time of query.

5. CONCLUSION

In a peer-to-peer system, anonymity protection against local passive attacks results in a weak anonymity protection. An adversary with global passive attack capabilities may identify anonymous peers in most anonymity solutions. Furthermore, an adversary with some collaborators may reveal identity of anonymous peers by launching collaborative attacks even it does not have global attack capabilities. The oblivious reply protocol provides k-anonymity protection for responders against global passive adversaries. If k is the group size, the oblivious reply protocol requires $\Theta(k^2)$ message exchanges for each anonymous reply, which is good comparing to complex secure multi-party computation approaches. The protocol allows to send multiple replies in the same network packet so message complexity can be reduced depending on the implementation.

As a future work, the proposed approach can be adapted to other DHT structures such as CAN [Ratnasamy et al. 2001] and Tapestry [Zhao et al. 2004] to create alternative anonymity solutions. Reducing complexity of the oblivious reply protocol and protecting anonymity against active attacks are some other future work directions. Anonymity against active attacks requires compilation of each protocol message, which means more message and computation complexity. Therefore, reducing complexity and increasing security against active attacks are conflicting tasks. Finding a trade-off between these tasks might be an interesting future work study.

REFERENCES

Aberer, K. and Despotovic, Z. 2001. Managing trust in a peer-2-peer information system. In *Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM)*.

Akavipat, R., Al-Ameen, M., Kapadia, A., Rahman, Z., Schlegel, R., and Wright, M. 2014. Reds: A framework for reputation-enhanced dhts. *Parallel and Distributed Systems, IEEE Transactions on 25,* 2 (Feb), 321–331.

AlSabah, M., Bauer, K., Elahi, T., and Goldberg, I. 2013. The path less travelled: Overcoming tors bottlenecks with traffic splitting. In *Privacy Enhancing Technologies*. Springer, 143–163.

Barbera, M. V., Kemerlis, V. P., Pappas, V., and Keromytis, A. D. 2013. Cellflood: Attacking tor onion routers on the cheap. In *Computer Security–ESORICS 2013*. Springer, 664–681.

Beimel, A. and Dolev, S. 2003. Buses for anonymous message delivery. *Journal of Cryptology 16,* 1, 25–39.

Bittorent. Bittorent web site. `http://bittorrent.org`. Accessed Nov 2014.

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

Boyd, C. and Mathuria, A. 2003. *Protocols for Authentication and Key Establishment*. Springer.

Can, A. B. and Bhargava, B. 2010. Anonymous access to trust information using k-anonymity chord. In *Proceedings of the Second International Conference on Advances in P2P Systems (AP2PS)*.

Chaum, D. 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM 4,* 2.

Chaum, D. 1988. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology 1*, 65–75.

Clarke, I., Sandberg, O., Wiley, B., and Hong, T. 2001. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of the First Privacy Enhancing Technologies Workshop (PET)*. LNCS, vol. 2009.

Danner, N., DeFabbia-Kane, S., Krizanc, D., and Liberatore, M. 2012. Effectiveness and detection of denial-of-service attacks in tor. *ACM Transactions on Information and System Security (TISSEC) 15,* 3, 11.

Das, A. and Borisov, N. 2013. Securing anonymous communication channels under the selective dos attack. In *Financial Cryptography and Data Security*. Springer, 362–370.

Das, A., Borisov, N., Mittal, P., and Caesar, M. 2014. Re 3: relay reliability reputation for anonymity systems. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*. ACM, 63–74.

Dingledine, R., Freedman, M., and Molnar, D. 2001. The Free Haven project: Distributed anonymous storage service. In *Proceedings of the First Privacy Enhancing Technologies Workshop (PET)*. LNCS, vol. 2009.

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

Dingledine, R. and Murdoch, S. J. 2009. Performance improvements on tor or, why tor is slow and what were going to do about it. *Online: http://www. torproject. org/press/presskit/2009-03-11-performance. pdf* .

DOUCEUR, J. 2002. The sybil attack. In *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS)*.

ELAHI, T., BAUER, K., ALSABAH, M., DINGLEDINE, R., AND GOLDBERG, I. 2012. Changing of the guards: A framework for understanding and improving entry guard selection in tor. In *Proceedings of the 2012 ACM workshop on Privacy in the electronic society*. ACM, 43–54.

FABIAN, B. AND FELDHAUS, T. 2014. Privacy-preserving data infrastructure for smart home appliances based on the octopus dht. *Computers in Industry*.

FREEDMAN, M. J. AND MORRIS, R. 2002. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*.

GNUTELLA. Wikipedia entry for Gnutella. `http://en.wikipedia.org/wiki/Gnutella`. Accessed Nov 2014.

GOLDREICH, O. 2001. *Foundations of Cryptography*. Vol. 1. Cambridge University Press.

GOLDSCHLAG, D. M., REED, M. G., AND SYVERSON, P. F. 1996. Hiding Routing Information. In *Proceedings of the First International Workshop on Information Hiding*.

GOLDWASSER, S. AND MICALI, S. 1982. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*.

HAN, J. AND LIU, Y. 2008. Mutual anonymity for mobile p2p systems. *IEEE Transactions on Parallel and Distributed Systems 19,* 8, 1009–1019.

HAZEL, S. AND WILEY, B. 2002. 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)*.

JAHID, S., NILIZADEH, S., MITTAL, P., BORISOV, N., AND KAPADIA, A. 2012. Decent: A decentralized architecture for enforcing privacy in online social networks. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*. 326–332.

KAMVAR, S., SCHLOSSER, M., AND GARCIA-MOLINA, H. 2003. The (eigentrust) algorithm for reputation management in P2P networks. In *Proceedings of the 12th World Wide Web Conference (WWW)*.

KONDO, M., SAITO, S., ISHIGURO, K., TANAKA, H., AND MATSUO, H. 2009. Bifrost: A novel anonymous communication system with dht. In *Parallel and Distributed Computing, Applications and Technologies, 2009 International Conference on*. IEEE, 324–329.

MCLACHLAN, J., TRAN, A., HOPPER, N., AND KIM, Y. 2009. Scalable onion routing with torsk. In *Proceedings of the 16th ACM conference on Computer and communications security*.

MISLOVE, A., OBEROI, G., POST, A., REIS, C., DRUSCHEL, P., AND WALLACH, D. S. 2004. Ap3: Cooperative, decentralized anonymous communication. In *Proceedings of the 11th ACM SIGOPS European Workshop*.

MITTAL, P. AND BORISOV, N. 2008. Information leaks in structured peer-to-peer anonymous communication systems. In *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 267–278.

MITTAL, P. AND BORISOV, N. 2009. Shadowwalker: peer-to-peer anonymous communication using redundant structured topologies. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*.

MITTAL, P., OLUMOFIN, F. G., TRONCOSO, C., BORISOV, N., AND GOLDBERG, I. 2011. Pir-tor: Scalable anonymous communication using private information retrieval. In *USENIX Security Symposium*.

NAMBIAR, A. AND WRIGHT, M. 2006. Salsa: a structured approach to large-scale anonymity. In *Proceedings of the 13th ACM conference on Computer and communications security*.

NAPSTER. Wikipedia entry for Napster. `http://en.wikipedia.org/wiki/Napster`. Accessed Nov 2014.

PANCHENKO, A., RICHTER, S., AND RACHE, A. 2009. Nisan: network information service for anonymization networks. In *Proceedings of the 16th ACM conference on Computer and communications security*.

RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R., AND SHENKER, S. 2001. A scalable content-addressable network. *SIGCOMM Comput. Commun. Rev. 31,* 4, 161–172.

REITER, M. AND RUBIN, A. 1998. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security 1,* 1, 66–92.

REN, J., LI, T., AND LI, Y. 2008. Anonymous communications in overlay networks. In *Proceedings of IEEE International Conference on Communications (ICC)*.

RENNHARD, M. AND PLATTNER, B. 2002. Introducing morphmix: Peer-to-peer based anonymous internet usage with collusion detection. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES)*.

RIPEANU, M., FOSTER, I., AND IAMNITCHI, A. 2002. Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing 6,* 1, 50–57.

SAROIU, S., GUMMADI, P., AND GRIBBLE, S. 2002. A measurement study of peer-to-peer file sharing systems. In *Proceedings of the Multimedia Computing and Networking*.

SHIRAZI, F., DIAZ, C., MULLAN, C., WRIGHT, J., AND BUCHMANN, J. 2013. Towards measuring resilience in anonymous communication networks. In *6th Workshop on Hot Topics in Privacy Enhancing Technologies, Bloomington, USA*. Vol. 12.

SINGH, A. AND LIU, L. 2003. Trustme: Anonymous management of trust relationships in decentralized P2P system. In *Proceedings of the 3rd IEEE Conference on Peer-to-Peer Computing (P2P)*.

STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., AND BALAKRISHNAN, H. 2001. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM Comput. Commun. Rev. 31,* 4, 149–160.

SWEENEY, L. 2002. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10,* 5, 557–570.

SYVERSON, P. F., GOLDSCHLAG, D. M., AND REED, M. G. 1997. Anonymous connections and onion routing. In *Proceedings of the IEEE Symposium on Security and Privacy*.

TRAN, A., HOPPER, N., AND KIM, Y. 2009. Hashing it out in public: common failure modes of dht-based anonymity schemes. In *Proceedings of the 8th ACM workshop on Privacy in the electronic society*. ACM, 71–80.

WACEK, C., TAN, H., BAUER, K. S., AND SHERR, M. 2013. An empirical evaluation of relay selection in tor. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

WALDMAN, M., RUBIN, A. D., AND CRANOR, L. F. 2000. Publius: A robust, tamper-evident, censorship-resistant web publishing system. In *Proceedings of the 9th Conference on USENIX Security Symposium*.

WANG, Q. AND BORISOV, N. 2012. Octopus: A secure and anonymous dht lookup. In *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*. IEEE, 325–334.

WANG, Q., LIN, Z., BORISOV, N., AND HOPPER, N. 2013. rbridge: User reputation based tor bridge distribution with privacy preservation. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

WANG, Q., MITTAL, P., AND BORISOV, N. 2010. In search of an anonymous and secure lookup: attacks on structured peer-to-peer anonymous communication systems. In *Proceedings of the 17th ACM conference on Computer and communications security*.

ZHAO, B., HUANG, L., STRIBLING, J., RHEA, S. C., JOSEPH, A. D., AND KUBIATOWICZ, J. 2004. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications 22,* 1, 41–53.

Ahmet Burak Can is currently affiliated with Department of Computer Science & Engineering at Hacettepe University, Turkey. He received the Ph.D. degree in Computer Science from Purdue University, West Lafayette. He has BS and MS degrees in Computer Science and Engineering from Hacettepe University. He is a member of the IEEE. His main research areas are computer networks, distributed systems, network security, and computer vision. His current research activities focus on trust and reputation management, anonymity protection, and incentive mechanisms in peer-to-peer systems.

Bharat Bhargava is a professor of the Department of Computer Science at Purdue University. Professor Bhargava is conducting research in security and privacy issues in distributed systems. Professor Bhargava is a Fellow of the IEEE and of the IETE. He has been awarded the charter Gold Core Member distinction by the IEEE Computer Society for his distinguished service. He serves on seven editorial boards of international journals. He also serves the IEEE Computer Society on Technical Achievement award and Fellow committees. Professor Bhargava is the founder of the IEEE Symposium on Reliable and Distributed Systems, IEEE conference on Digital Library, and the ACM Conference on Information and Knowledge Management.