Anonymous Access to Trust Information Using k-anonymity Chord

Ahmet Burak Can
Department of Computer Engineering
Hacettepe University
06800 Ankara/Turkey
abc@hacettepe.edu.tr

Bharat Bhargava
Department of Computer Science
Purdue University
West Lafayette, IN 47907 USA
bb@cs.purdue.edu

***Abstract*—In a reputation based trust network, each peer stores trust information of others and answers the trust queries, in addition to providing services to others. We present a cryptographic protocol on Chord, which provides anonymous access to trust information. Peers form anonymity groups and generate responses inside the group. Responder of a trust query has k-anonymity protection against an adversary who can sniff all communication on the network. Moreover, our encryption scheme ensures that the initiator of a trust query can check the validity of an anonymous reply.**

***Keywords*-anonymity, trust management, peer-to-peer**

I. Introduction

Trust is fundamental to achieve collaborative tasks in a peer-to-peer system. When organizing, sharing, and searching resources, trustworthy peers are identified based on the trust information. Due to the large scale and distributed nature of peer-to-peer systems, a centralized entity cannot track a massive number of peer interactions and manage all trust information. Consequently, the burden should be shared among the peers, whereby each peer can become a *trust holder* [1], [2]. A trust holder needs anonymity to protect itself against malicious peers. Such protection motivates a peer to perform the trust holding duty and may prevent denial of service attacks, making information more available.

In peer-to-peer systems, several methods are studied to protect anonymity: limitations on routing information exchange [3], probabilistic random path building [4], [5], and flooding [6], [7], [8]. These methods are vulnerable to global passive adversaries who can sniff all the communication on the network. Mix networks [9] or onion routers [10] might be adapted for peer-to-peer systems. Trusted mix nodes encrypt and shuffle the network traffic so a global passive adversary can not determine who is communicating with whom. In an ideal solution, peers should organize themselves to protect anonymity and should not depend on trusted nodes. This is more adequate for the decentralized nature of peer-to-peer systems.

We propose k-anonymity Chord [11] to protect the anonymity of a trust holder when responding to trust queries. As in most peer-to-peer systems [12], we assume the existence of a bootstrap peer, which is a connection point to the network. Peers register their pseudonyms and encryption keys to the bootstrap peer when joining the network for the first time. A new peer obtains some certificates during the registration and then, joins two overlay networks: *service* and *trust* networks. The service network can be any network substrate, e.g., Gnutella[12], Freenet [6]. In a service request, a peer queries the service network to find a particular service such as a file. Several service providers respond to the query and send back their certificates to the requester. For each service provider, the requester sends trust queries to the trust network. This network must overlay on k-anonymity Chord, which runs the oblivious reply protocol to protect anonymity of trust holders. Peers form anonymity groups of size k. Each peer in an anonymity group sends back a trust reply after receiving a trust query. A peer's reply can not be distinguished from the replies of others. Thus, the real responder has k-anonymity protection against global passive adversaries. The requester can check the authenticity of trust replies to identify fake replies of malicious peers.

Section II explains the related research. Section III presents the encryption architecture, peer registration, and communication during service and trust queries. Section IV introduces k-anonymity Chord and the oblivious reply protocol. Section V gives a discussion about performance considerations and other issues. Section VI outlines the conclusions and results of our work.

II. Related Work

Various methods have been studied to protect anonymity in computer networks. We outline some of the prominent methods as follows.

Mix networks and onion routers. Mix networks are first proposed by Chaum [9] to protect anonymity of communicating parties for delay tolerant applications. Trusted mix nodes use cover traffic to shuffle messages so an adversary can not determine who is communicating with whom. Onion routers [10] form an overlay network to build anonymous, bi-directional virtual circuits for real-time communication. While mix networks are designed for delay tolerant applications, e.g., e-mail systems, onion routing is more feasible for real-time applications such as HTTP. Tor [13] extends onion routing with forward secrecy, congestion control, integrity checking and configurable exit policies. Our approach aims

 ISBN: 978-1-61208-102-1

to protect anonymity without relying on a trusted mix nodes or onion routers so it should be considered in a different category of anonymity systems. In our approach, peers register themselves to a trusted peer but this peer does not participate in the anonymity protocol.

DC-Nets. Chaums dining cryptographer networks (DC-net) [14], 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 Chaums DC-net impractical for real life scenarios.

Buses. In the bus [15] approach, synchronous message tokens traverse in the network forever. When a peer receives a bus, it fills some seats with encrypted messages or dummy messages if it does not have any real message. When a bus arrives to a receiver, all or some related seats are decrypted to get the message. Ren et. al [16] applied this approach on peer-to-peer networks and circulated bus tokens on overlay rings. Although the bus approach can protect sender/receiver anonymity, the bus must traverse the network forever even the nodes do not have any real message.

Flooding. Freenet [6] and Freehaven [7] flood the storage requests in peer-to-peer storage systems to protect the requester and responder anonymity. Since no peer on the flooding path knows the whole path, it is hard to determine the requester and responder. Trustme [8], floods encrypted trust queries to the network and trust holders send back authenticated replies. Han and Liu [17] split a query into n shares and send the shares to neighbors in a peer-to-peer network. The peers who take t shares can decrypt and flood the query. The responder builds an onion path to the requester and sends the response on this path. MuON [18] uses a gossip protocol to reduce the traffic caused by query flooding. In all of these approaches, flooding protects anonymity if the adversary can not sniff the whole network. Additionally, excessive network traffic caused by flooding reduces the scalability of these systems.

Random path building. In Crowds [4], nodes form anonymity groups (crowds) and randomly forwards the requests in the crowd to protect the requester anonymity. Tarzan [5] establishes a random tunnel between a peer and an Internet server to protect the peers anonymity. Since none of the peers on a tunnel know the whole path, the initiator of a request can not be determined. MorhpMix [19] defines a peer-to-peer mix network where random mix nodes are selected during an anonymous communication. These approaches protect anonymity if the adversary can not sniff the whole communication path.

Changing the routing method. Anonymity has been studied on Chord by using recursive, randomized, indirect, split, bidirectional routing [20]. Achord [3] defines routing limitations on Chord to provide censorship resistance. These schemes offer anonymity protection in a local adversary model and can not protect anonymity against a global passive adversary.

III. ARCHITECTURE

We assume the existence of a bootstrap peer (*bp*), which provides a connection point to the network for new peers. There might be multiple bootstrap peers to provide tolerance to failures and attacks. For simplicity of the notation, the rest of the paper considers one bootstrap peer, which is a basic certification authority for pseudonyms and encryption keys. It has a public/private key pair $\{U_{bp}, R_{bp}\}$. We assume all peers learn U_{bp} in a secure way, e.g., through a secure web site. A peer registers itself to the bootstrap peer when joining the network for the first time. During the registration, the bootstrap peer issues some certificates for the new peer.

P_i denotes the i^{th} peer. ID_i and TID_i are the pseudonyms of P_i in the service and trust networks respectively. While ID_i is selected by P_i before registration, TID_i is assigned by the bootstrap peer during the registration. ID_i and TID_i have no relation with each other. $\{U_i, R_i\}$ is P_i's public/private key pair for the service network operations. For the trust network, it has $\{TU_i, TR_i\}$ and $\{OU_i, OR_i\}$ key pairs. 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.

If K is a public key or a symmetric encryption key, $K(M)$ stands for the encryption of M with key K for message confidentiality. When K is a private key, the operation is considered as signing of M. $H[M]$ is the hash digest of M. $X|Y$ denotes the concatenation of X and Y.

A. Adversary Model

An adversary tries to learn the pseudonym (TID) or IP number of a trust holder. It (we assume that an adversary is a peer so we will use "it") might have passive attack capability, e.g., sniffing the network communication. It may collaborate with some peers and launch attacks by coordinating with them. A local passive adversary can perform passive attacks only in a limited number of networks links. A global passive adversary can perform passive attacks on all links of a network. It has polynomial time computational capabilities and can not break cryptographic algorithms in polynomial time. Semi-honest adversary model [21] means that an adversary stays complaint with the protocols but may observe the network communication to obtain information.

B. Peer Registration

To demonstrate registration operation, we assume that P_i is joining the network for the first time and registering itself to the bootstrap peer as follows:

1) P_i sends $U_{bp}(ID_i|U_i|TU_i|OU_i|r_1)$ to the bootstrap peer as a registration request. r_1 is a random value

 ISBN: 978-1-61208-102-1

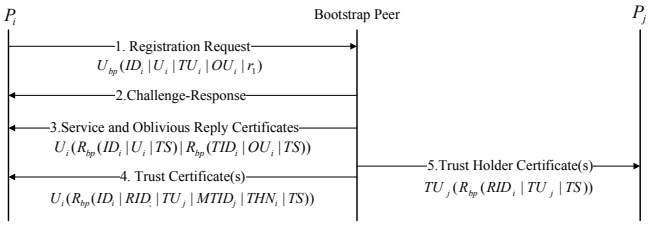

Figure 1. Registration of P_i to the bootstrap peer

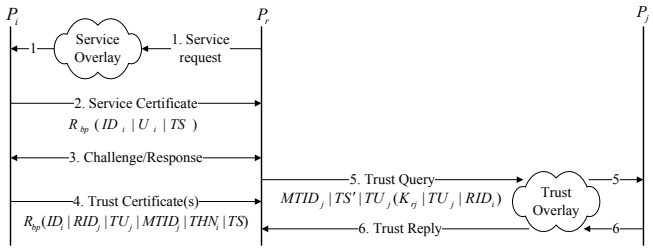

Figure 2. P_r is searching for a service provider (P_i) and then, querying its trust information

selected by P_i. Only the bootstrap peer can read the contents due to the encryption. It decrypts the request and stores ID_i, U_i, TU_i, OU_i for future accountability.

2) The bootstrap peer runs a challenge-response protocol to verify that P_i has U_i, TU_i, OU_i keys.
3) Assuming P_i passed step 2, the bootstrap peer selects TID_i value and sends back $U_i(R_{bp}(ID_i|U_i|TS)|R_{bp}(TID_i|OU_i|TS))$ to P_i. Only P_i can decrypt the message and check if TS value is same. $R_{bp}(ID_i|U_i|TS)$ is a *service certificate*. P_i sends this certificate to peers who request P_i's services. It proves P_i's registration to the service requester. $R_{bp}(TID_i|OU_i|TS)$ is an *oblivious reply certificate*, which is used during the oblivious reply operations explained in Section IV-C. It also informs P_i about its TID_i.
4) The bootstrap peer randomly selects P_i's trust holders. Let P_j be such a trust holder. The bootstrap peer sends $U_i(R_{bp}(ID_i|RID_i|TU_j|MTID_j|THN_i|TS))$ to P_i. The inner part, $R_{bp}(ID_i|RID_i|TU_j|MTID_j|THN_i|TS)$, is the *trust certificate*, which means that a peer associated with $MTID_j$ value and TU_j key will store P_i's trust information. $MTID_j$, explained in Section IV, is an anonymized value of TID_j and represents a range of peers instead of a particular peer. Using this certificate, P_i or another peer can send trust queries destined to P_j, but can not learn P_j's identity. RID_i is a random value to hide P_i's real identity from its trust holder, P_j. THN_i is the number of P_i's trust holders determined by the bootstrap peer.
5) The bootstrap peer sends a *trust holder certificate* to each trust holder. For example, P_j's trust holder certificate is $R_{bp}(RID_i|TU_j|TS)$. This certificate informs P_j about its trust holding duty on P_i's trust information. To protect P_i's anonymity, ID_i is not added to the certificate. P_j does not know P_i's identity but it can answer trust queries by using RID_i value.

A service certificate and related trust certificates expire according to TS field. The owner of an expired service certificate requests a new one from the bootstrap peer. Figure 1 briefly explains the peer registration operation.

C. Searching a Service Provider and Sending a Trust Query

Assume that P_i is a service provider, P_j is a trust holder of P_i, and P_r requests a service from P_i. Figure 2 shows the message exchanges during a service request and a trust query. In Step 1, P_r sends a query to the service network to find a service, e.g., a particular file. Assuming P_i has the service, it sends back a reply message containing its service certificate, $R_{bp}(ID_i|U_i|TS)$ (Step 2). P_r verifies the certificate using U_{bp} and runs a challenge/response protocol to authenticate P_i (Step 3). Then, P_i sends its trust certificates to P_r (Step 4). In our case, P_r receives only $R_{bp}(ID_i|RID_i|TU_j|MTID_j|THN_i|TS)$, which is the trust certificate for trust holder P_j. If ID_i, TS values match with the values from the service certificate, P_r ensures that P_j is a legitimate trust holder. However, P_r can not learn P_j's identity. If there are other trust holders, THN_i value informs P_r about the existence of other trust holders and forces P_i to send all certificates.

After verifying service and trust certificates, P_r sends a *trust query*, $MTID_j|TS'|TU_j(K_{rj}|TU_j|RID_i)$, to the trust network (Step 5). TS' is a time-stamp and unique among P_r's queries. K_{rj} is a random session key, which can only be learned by P_j due to encryption with TU_j. The encrypted part, $TU_j(K_{rj}|TU_j|RID_i)$, includes TU_j and RID_i fields to prevent forgery of the content. P_j checks these values and understands if the query is destined to itself. In Step 6, P_j sends back a *trust reply* message. The details of Step 5 and 6 will be explained in the next section.

IV. k-ANONYMITY CHORD

Chord [11] is a distributed hash table (DHT) designed for peer-to-peer networks. Chord's algorithm assigns each resource to a particular peer. We use Chord to access trust information efficiently. However, anonymity of a trust holder can not be protected on Chord when responding to a trust query. A responder can be identified since peers partially learn the network structure using Chord's finger tables. Additionally, a peer may learn more about an arbitrary part of the address space by sending excessive finger requests [3]. This makes guessing a responder easier without having global sniffing ability.

 ISBN: 978-1-61208-102-1

We propose the oblivious reply protocol on Chord to provide k-anonymity protection for trust holders. This means that a trust holder's identity can not be distinguished from k other peers when responding to trust 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, the search operation is modified to protect anonymity of the responder. In our case, the trust network overlays on a k-anonymity Chord ring. A peer joins the trust network with its TID value, e.g., P_j joins with TID_j. For the rest of the section, we assume that P_i is a service provider, P_j is a trust holder of P_i and P_r wants to get a service from P_i.

A. Formation of Anonymity Groups

As explained in Section III-C, P_r obtains P_i's service and trust certificates during its service request and prepares a trust query destined to P_j. However, P_r can not send the query directly to P_j since it does not know TID_j. It sends the query on the k-anonymity Chord by putting $MTID_j$ value to the query message. $MTID_j$ is an anonymized version of TID_j where the last m bits are set to zero. In a trust query, $MTID_j$ represents the range of pseudonyms between $MTID_j$ and $MTID_j + 2^m$. We call this range *search range* and the peers in the search range *target peers*. The bootstrap peer decides the value of m so that the expected number of target peers in a search range is equal to k. Since the bootstrap peer registers all peers, it can compute a precise m value. To explain $MTID_j$ selection, we give a numerical example.

Chord peers are located on a 2^n circular address space where n is the length of a TID. We assume that the bootstrap peer uniformly distributes peers on this address space. Suppose that $n = 32, k = 64$, $TID_j = 12345678H$ 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 (When $X = 1$, there is a peer on 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 peer selects a search range that has $Y \geq k = 64$ expected number of peers. Let S be the number of locations in a search range. Due to the uniformity of peer 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 peer finds that $S \geq 2^{24}$. This inequality suggests us that $m \geq \log_2 S = \log_2 2^{24} = 24$.

Then, the bootstrap peer computes $MTID_j$ as $MTID_j = 12345678H \wedge FF000000H = 12000000H$. This means that P_j has a TID_j value between $12000000H$ and $12FFFFFFH$. The expected number of peers in this range is 64 due to our selection.

B. Routing a Trust Query

Let $P_0, P_1 \ldots P_{k-1}$ be k target peers in $MTID_j$ and $MTID_j + 2^m$ range and P_j be one of the target peers. By this definition, P_0 is the owner of $MTID_j$ value. We define a two-phase routing method for trust queries. The first phase is a recursive Chord search to find P_0, the owner of $MTID_j$ value. P_r starts the first phase by preparing a trust query, $MTID_j|TS'|TU_j(K_{rj}|TU_j|RID_i)$, for P_j. It looks up its finger table, sends the query to the closest peer preceding P_0. The receiving peer forwards the query to another one by looking up $MTID_j$ value in its finger table. Forwarding operation continues until P_0 receives the query. TS' value gives a hint for the expiration time. Each forwarding peer caches the query to send the trust reply back to P_r.

After the query reaches to P_0, the second phase starts and the oblivious reply protocol runs to send the query to P_j and get its reply anonymously. The following section explains this protocol. In the attack scenarios, P_r tries to identify P_j by sniffing the network or obtaining collaborators. Note that, P_i may pretend to be P_r to learn P_j's identity.

C. Oblivious Reply

Oblivious reply is a cryptographic protocol to protect anonymity of a trust holder against a global passive adversary. This protocol is secure against collaborating passive adversaries in semi-honest adversary model [21]. The basic idea is that each target peer generates a separate trust reply. These replies can not be linked with the senders and P_j's reply can not be tracked during the operation of protocol. The protocol has several assumptions:

- Each target peer knows its search range and the other target peers in the search range. Additionally, each target peer knows its exact location in the range, i.e, the number of hops from P_0 and P_{k-1}.
- All target peers exchange their $R_{bp}(TID_i|OU_i|TS)$ certificates. Once the certificates are exchanged, they can be used in many trust queries.
- The public key encryption scheme ensures semantic security [22]. This implies that the result of an encryption depends on the message, key, and a sequence of coin tosses. Thus, encryption of a message with the same public key results in a different cipher text in each trial. However, the decryptions of these cipher texts give the same plain text.
- The public key encryption scheme is not commutative, which means that $A(B(M)) \neq B(A(M))$.

 ISBN: 978-1-61208-102-1

After P_0 receives P_r's query, each target peer forwards the query until P_{k-1} receives it. P_{k-1} tries to decrypt the contents of the query. If the decryption is successful, it prepares O_{k-2}^{k-1} as follows:

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

O_{k-2}^{k-1} denotes P_{k-1}'s *oblivious reply*, which is destined to P_{k-2}. The last field, AB, is the authenticity bit. It is set to 1 if the reply is authentic.

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

1) P_{k-1} sends $MTID_j|TS'|O_{k-2}^{k-1}$ 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 $MTID_j|TS'|(O_{k-3}^{k-1} \cup O_{k-3}^{k-2})$ 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 the 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 $MTID_j|TS'|(O_{k-4}^{k-1} \cup O_{k-4}^{k-2} \cup O_{k-4}^{k-3})$ to P_{k-4}.
4) This operation is repeated by all target peers until P_0 receives $MTID_j|TS'|(O_0^{k-1} \cup \ldots \cup O_0^2 \cup O_0^1)$. After decrypting the last layers, it 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 one. If $P_0 = P_j$, it ignores all replies and generates its own reply and sends it to P_r.
5) P_r decrypts the reply using K_{rj}. If the reply is containing the correct RID_i and TS' values, it is authentic. A malicious peer can not forge an authentic reply since it can not obtain K_{rj}.

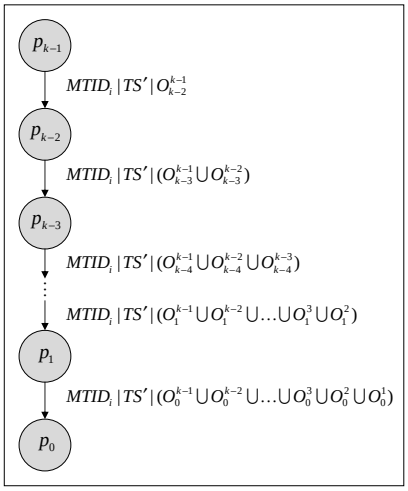

Figure 3. Message communication among target peers in the oblivious reply protocol

Figure 3 shows the flow of oblivious replies among target peers. If P_r is a global passive adversary, it can observe all the communication among target peers but it can not identify the sender of any reply. Identical reply sizes, semantic security assumption, layered encryption of replies, and randomization of their order on each target peer do not allow P_r to trace the replies. The oblivious reply protocol provides k-anonymity protection for trust holders as long as adversaries perform passive attacks. Due to space limitations, we can not give the proofs of our claim in this paper. Interested readers may refer to [23]. For a better understanding of our encryption scheme, similar ideas in [9], [14], [10] can be referred.

The oblivious reply protocol can not protect anonymity if adversaries perform active attacks, e.g. forging replies, dropping selected replies, skipping a target peer. If a target peer can be forced to stay complaint with the rules of oblivious reply protocol, these attacks can be prevented. Goldreich [21] shows that semi-honest behavior can be forced by compiling each instruction (message).

V. Discussion

Performance Considerations. We consider the message complexity to evaluate the performance of oblivious reply protocol. A reply is forwarded up to $O(k)$ times. For k replies, $O(k^2)$ network packets are forwarded in phase 2. More than one reply can be sent in the same network packet for efficiency. Assuming η is the number of replies in a network packet, phase 2 can be performed with up to $O(k^2/\eta)$ network packets. Note that the size of a reply decreases and η increases as replies are getting closer to P_0.

Sending trust holder certificates. In Section III-B, the bootstrap peer sends a separate certificate to each trust holder in step 6. If a global passive adversary observes the bootstrap peer during this step, it

can learn P_j's identity. Therefore, the bootstrap peer sends a special message containing P_j's certificate, $MTID_j|TS'|TU_j(R_{BS}(H[ID_i]|TU_j|TS))|"Cert"$ to the trust network like a normal trust query. The last field in the message indicates that this message is a certificate, not a trust query. All target peers forward the message till the last peer in the search range receives it. Due to the encryption with TU_j, only P_j can read the content of the message. No peer can understand who is the receiver of the certificate.

VI. Conclusion

In a peer-to-peer system, defending anonymity against only local attacks results in a weak anonymity protection. An adversary with global passive attack capabilities or with some collaborators may learn about the anonymous peer by launching collaborative attacks. The oblivious reply protocol provides k-anonymity protection for a trust holder against global passive adversaries. The protocol requires $O(k/\eta)$ message exchanges where k is the group size and η is the number of reply messages that can fit into a network packet.

The oblivious reply protocol can be adapted to other DHT structures or applications that need responder anonymity. Moreover, our ideas can be used to support requester anonymity. A group of peers may generate an anonymous request so the identity of the requester is protected.

References

[1] 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.

[2] 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.

[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] 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.

[5] 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.

[6] 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)*, ser. LNCS, vol. 2009, 2001.

[7] 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)*, ser. LNCS, vol. 2009, 2001.

[8] 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.

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

[10] 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.

[11] 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.

[12] Gnutella. http://en.wikipedia.org/wiki/Gnutella.

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

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

[15] A. Beimel and S. Dolev, "Buses for anonymous message delivery," *Journal of Cryptology*, vol. 16, no. 1, pp. 25–39, 2003.

[16] J. Ren, T. Li, and Y. Li, "Anonymous communications in overlay networks," in *Proceedings of IEEE International Conference on Communications (ICC)*, 2008.

[17] J. Han and Y. Liu, "Mutual anonymity for mobile p2p systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 8, pp. 1009–1019, 2008.

[18] N. Bansod, A. Malgi, B. K. Choi, and J. Mayo, "Muon: Epidemic based mutual anonymity in unstructured p2p networks," *Computer Networks*, vol. 52, no. 5, pp. 915 – 934, 2008.

[19] 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.

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

[21] O. Goldreich, *Foundations of Cryptography*. Cambridge University Press, 2001, vol. 1.

[22] 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.

[23] A. B. Can, "Trust and anonymity in peer-to-peer systems," Ph.D. dissertation, Department of Computer Science, Purdue University, 2007.

 ISBN: 978-1-61208-102-1