Rumor Riding: Anonymizing Unstructured Peer-to-Peer Systems

Jinsong Han and Yunhao Liu

Department of Computer Science and Engineering
Hong Kong University of Science and Technology
jasonhan@cse.ust.hk liu@cse.ust.hk

***Abstract*—Although anonymizing Peer-to-Peer (P2P) systems often incurs extra costs in terms of transfer efficiency, many systems try to mask the identities of their users for privacy considerations. Existing anonymity approaches are mainly path-based: peers have to pre-construct an anonymous path before transmission. The overhead of maintaining and updating such paths is significantly high. In this paper, we propose Rumor Riding (RR), a lightweight mutual anonymity protocol for decentralized P2P systems. RR employs a random walk scheme which frees initiating peers from the heavy load of path construction. Compared with previous RSA-based anonymity approaches, RR also takes advantage of lower cryptographic overhead by mainly utilizing a symmetric cryptographic algorithm to achieve anonymity. We demonstrate the effectiveness of this design through trace-driven simulations. The analytical and experimental results show that RR is more efficient than existing protocols. We also discuss our early implementation experiences with the RR prototype.**

***Index Terms*—Mutual Anonymity, Non Path-based, P2P**

I. INTRODUCTION

Peer-to-peer (P2P) networks, such as Napster, Gnutella, and BitTorrent, have become essential media for information dissemination and sharing over the Internet. Concerns about privacy, however, have grown with the rapid development of P2P systems. Recently, a number of P2P users have encountered problems caused by being traced on non-anonymous P2P systems due to their plain-text query messages and direct-downloading behaviors. Hence, the requirement for *anonymity* has become increasingly essential in current P2P applications for both content requesters and providers.

A number of methods [1, 2, 10, 13, 20] have been proposed to provide anonymity. Most, if not all of them, deliver messages via non-traceable paths comprised of several anonymous proxies or middle agent peers. In these approaches, known as path-based approaches, users usually need to construct anonymous paths before transmissions. All nodes in the path cooperate to forward data to a receiver. Data is pre-wrapped by the initiator in a layered-encryption packet (usually using asymmetric cryptographic algorithms, such as RSA), which will be peeled off along the path to the receiver. Although path-based protocols provide strong anonymity, they have the following problems:

1. Pre-construction of paths requires users to obtain a large number of IP addresses and public keys from other peers in advance. Furthermore, initiators must perform large amounts of cryptographic processing in preparation for layer-encrypted packets. Both the collection of information and the preparation of packets incur high costs.

2. Initiators have to periodically update middle nodes along the anonymous paths. An invariable path might otherwise become increasingly vulnerable under the analysis of attackers. In addition, users often expect to extend the length of anonymous paths, as a longer path entails a higher degree of anonymity. Both of these requirements increase the maintenance and update overhead.

3. In highly dynamic P2P systems, peers randomly join and leave. If a chosen node goes offline, the whole path fails, and such a failure is often undetected by the initiator. Therefore, a "blindly-assigned" path is very unreliable, and users have to frequently probe the path and retransmit messages.

To address the above problems, we propose a *non path-based* anonymous P2P protocol called **R**umor **R**iding (RR). The design goal is twofold: first, to eliminate the huge overhead of path construction and maintenance; second, to use a symmetric cryptographic algorithm to replace the asymmetric one so as to reduce the cryptographic overhead and make the protocol more practical.

In the RR design, we first let an initiator encrypt the query message with a symmetric key, and then send the key and the cipher text to different neighbors. The key and the cipher text take random walks separately in the system, where each walk is called a *rumor*. Once a key rumor and a cipher rumor meet at some peer, the peer is able to recover the original query message and act as an agent to issue the query for the initiator. A similar idea is also employed during the query response, confirm, and file delivery processes. Thus, the rumors serve as the primitives of this protocol to achieve the mutual anonymity and meet the design objectives. The highlights of our contributions are as follows.

- We design a lightweight mutual anonymous P2P protocol, RR, in which anonymous paths are automatically constructed via the rumors' random walks. Neither the initiator nor the responder needs to be concerned with anonymous path construction and maintenance. Extending the scope of anonymous servants from a small clique of

1-4244-0593-9/06/$20.00 ©2006 IEEE

nodes to the entire P2P network, RR significantly increases the degree of anonymity of the systems.

- RR mainly employs a symmetric cryptographic algorithm to achieve anonymity. Without involving the asymmetric cryptographic algorithm in the transmission of messages, it significantly cuts down the cryptographic overhead for the initiator, the responder, and the middle nodes.
- We conducted trace-driven simulations. We show that RR outperforms existing protocols in terms of efficiency. In previous anonymous systems, users have to communicate with other peers or bootstrapping servers to obtain enough proxies' IP addresses. With RR, each initiating peer has no requirement of extra information to construct paths, thus eliminating the risk of information leakage caused by links that are used for peers to request the IP addresses of anonymous proxies.

The rest of this paper is structured as follows. In Section II, we describe the related work on anonymous communications. In Section III, we present the design of the RR protocol. Section IV analyzes and discusses the key issues and the degree of anonymity of RR. Section V presents simulation results and performance evaluations. We introduce our early implementation experience with RR in Section VI, and conclude the work in Section VII.

II. Related Work

Since Chaum [1] pioneered the concept of anonymity, many approaches have been proposed to acquire anonymous communication. These approaches fall into two categories: path-based anonymous delivery and anonymous multicast. Onion Routing[4] and Mix[1] are the classical approaches of the first category, and P^5 [10] is representative of the latter.

Onion Routing[4], as well as its second generation, Tor [14], is the most popular path-based protocol providing anonymous connections over the Internet based on a layered-encryption method. They mainly focus on the IP layer rather than the application level. APFS [6] proposes to provide the responder with anonymity based on the Onion design. Shortcut-responding Protocol [13] provides P2P mutual anonymity with a reduced response delay. Crowds [2] enables the intermediate nodes to randomly choose a successor to forward the request to, and encrypts it with a symmetric cipher algorithm. Due to the inherent problem of these path-based approaches, users suffer the path construction and the high cryptographic overhead from RSA. Even worse, they need to periodically probe and update the anonymous paths. All of these incur nontrivial overhead, particularly for fresh nodes.

P^5 [10] is based on the concept of broadcasting to achieve mutual anonymity: all participants in the same channel send fixed-length encrypted packets at a fixed rate, making it look like all the participants form a ring. To make the broadcasting scalable, P^5 proposes a novel broadcast hierarchy architecture to construct the anonymous broadcasting channels. All nodes are organized into different broadcasting groups, and each group is mapped into a node in the virtual hierarchical spinning tree. A high level in the hierarchy guarantees a high level of anonymity at a high cost of communication bandwidth, and vice versa. Users can make a balanced selection based on the tradeoff between the level of anonymity and the cost of communication efficiency. Due to well designed channels and hierarchy overlaid spanning trees, P^5 also improved the communication efficiency. To defend against traffic analysis, P^5 introduces noise packets to maintain a fixed transmitting rate for each user. However, the flooding-based message delivery inevitably incurs large traffic overhead on the network, and it is not practical to employ P^5 in a P2P system.

The proposed protocol in this work, Rumor Riding, combines the positive features of the path-based anonymous delivery and the anonymous multicast approaches. Instead of using a single path, RR allows messages to be sent via multiple anonymous paths (at least two) without considering path construction, while 'shrinking' the scope of anonymous multicast.

III. Rumor Riding Anonymous Protocol

In this section, we present our work, the Rumor Riding (RR) protocol, which includes five major components: *Rumor Generation and Recovery*, *Query Issuance*, *Query Response*, *Query Confirm*, and *File Delivery*. Although RR is designed for unstructured P2P systems, it can be easily extended into other distributed systems.

A. Rumor Generation and Recovery

RR employs the AES algorithm to encrypt original messages, as illustrated in Fig. 1. The key size is 128-bit, which is secure enough for encryption. To determine whether a pair of cipher and key rumors hit, we employ a Cyclic Redundancy Check (*CRC*) function to attach a *CRC* value, *CRC*(*M*), to the message *M*.

In Fig. 1, we also illustrate the rumor recovery procedure in the agent node *S*. For received key rumors and cipher rumors, *S* uses AES to recover a message *M'* and the checksum *CRC*(*M'*). It then performs the *CRC* function to the recovered *M'* and compares the result with *CRC*(*M'*). If they match, the sower *S* is aware that it has successfully recovered a message *M*. The purpose of the *CRC* function is to avoid using a complex text understanding technique to distinguish a meaningful *M*.

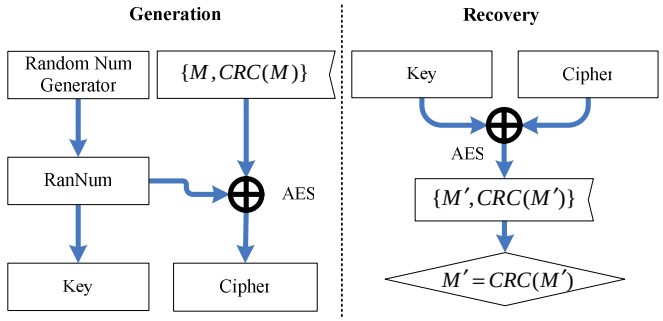

Figure 1 Rumor generation and recovery.

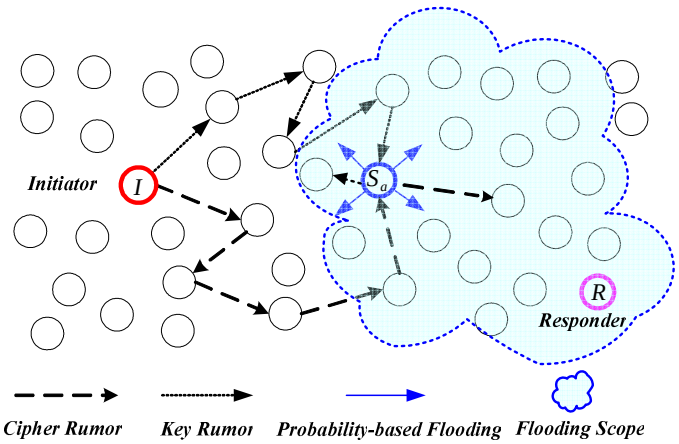

Figure 2 Query issuance.

B. Query Issuance

When an initiator I wishes to start an anonymous query, it first generates the query content q, and inserts its own public key K_I^+ into q, which will be used by the responder in the query response and file delivery phase. I then employs an AES cryptographic algorithm to encrypt q into a cipher text C with a symmetric key K. It organizes the key K and the cipher text C into two query rumors, r_K and r_C. In Gnutella, each packet is labeled with a Descriptor ID, a string that uniquely identifies the packet. RR also uses descriptors to identify rumors. Hence, two random number strings, IDK_I and IDC_I, are used to label the two rumors. After generation, I forwards the rumor messages to two randomly chosen neighbors, as shown with the dashed and dotted lines in Fig. 2. The query cipher rumor and the query key rumor then start their random walks in the P2P network.

The strategy of processing rumors is different from that of processing normal queries. RR requires every node to temporarily keep a local cache to store the received rumors. When a node receives a query key rumor, it performs the rumor recovery procedure to check all cached cipher rumors. If a decrypted rumor holds a plain text matching the *CRC* value, q will be successfully recovered. Whether or not there is a match, this intermediate node reduces the TTL value of the received key rumor by one, keeps a temporary record containing the ID and route information of this rumor in the local cache, and forwards it to a randomly chosen neighbor. This procedure continues until the TTL value of this rumor is reduced to zero.

For the received query cipher rumor, the process is similar. Therefore, if a pair of query rumors reach a certain node, no matter what the sequence is, this node will eventually recover the original q. The key issue with this procedure is that the number of rumors and their initial TTL values need to be carefully selected so that at least one pair of rumors, including a key and a cipher, will meet. A detailed discussion on the parameter settings are presented in Section IV.

If one intermediate node that recovers a q is willing to act as an agent peer, it conducts a search on behalf of the unknown I. We call this node a *sower*. When a peer identifies itself as a sower, it proceeds as follows. First, it checks the TTL values defined in the rumors; if they are not zero, the sower forwards the rumors out, so that if there are attackers who can overhear some of the messages sent to the sower, it is still not trivial to determine whether or not the peer is a sower. Second, the sower, S_a, as illustrated in Fig. 2, attaches the original query message q with its IP address, and then issues the query marked with a label ID_q in a plain text (ID_q is also used for S_a to locate the correlated r_K and r_C). In this operation, we avoid a blind flooding. Instead, we employ a probability-based-flooding search, in which the sower selects a subset of its neighbors and issues the query. Note that the sower does not send the query to the nodes which sent or have been sent the two rumors of this query. Such a selective flooding is effective on defending against collaborating attacks, on which we discuss further in Section IV.B. In later discussions, we use the l_K and l_C to denote rumor paths from I to S_a.

C. Query Response

When a node receiving the query has a copy of the desired file, it becomes a responder R. To respond to the query, R encrypts the plain text of the response message re using the initiator's public key K_I^+. It encrypts $<(re)_{K_I^+}, ID_q, \text{IP}_{S_a}, K_R^+>$ using AES, where K_R^+ is the public key of R, and encloses the cipher text and the key into two response rumors, re_K and re_C. They are then assigned with IDK_R and IDC_R, respectively.

After being sent out from R, two rumors start their random walks in the system. We illustrate this procedure in Fig. 3. RR guarantees that at least one pair of rumors meet at a certain peer S_b. We use l'_K and l'_C to denote their paths from R to S_b. S_b decrypts the cipher text in re_C with the key in re_K, and recovers the IP address of sower S_a.

If S_b volunteers to forward the response for R, it contacts S_a via a TCP connection, and forwards these two response rumors to S_a. Note that S_b also attaches its IP address, ID_q, IDK_R, and IDC_R to the two rumors. When S_a receives the responses re_K and re_C, it delivers them to the originating peers of r_K and r_C. Two response rumors are marked with IDK_I and IDC_I, to help them walk along the reversed paths of l_K and l_C. The successor nodes continue this procedure. Thus, two response rumors make use of l_K and l_C to reach I.

Upon two response rumors, I recovers $(re)_{K_I^+}$ from re_K and re_C, and then decrypts $(re)_{K_I^+}$ to recover the original response messages re using its private key K_I^-. A flooding search

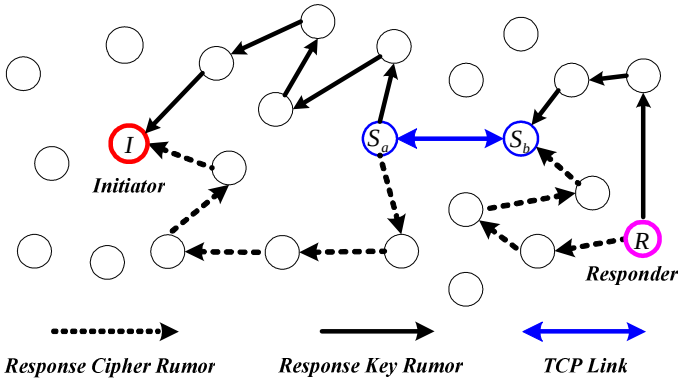

Figure 3 Query response.

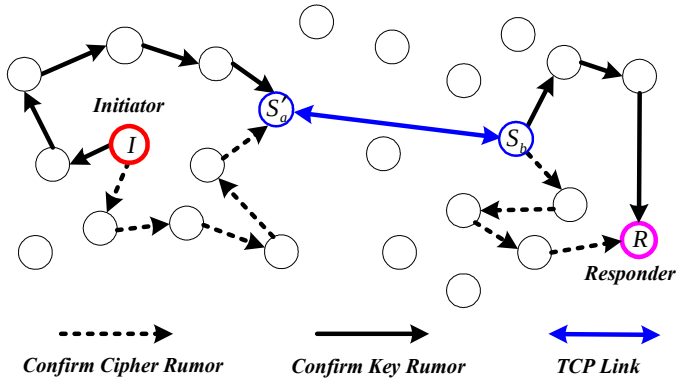

Figure 4 Query confirm.

procedure may raise multiple responses. To simplify the demonstration, we assume that *I* only selects one candidate as the file provider for multiple responses. Without loss of generality, we continue using *R* to denote this desired provider.

D. Query Confirm

In the query confirm phase, *I* uses the responder's public key to encrypt the confirm message *rc*. *I* then encrypts $<(rc)_{K_R^+}, IDK_R, IDC_R, IP_{S_b}>$ and obtains two confirm rumors, rc_K and rc_C, which take random walks in the system. Note that two confirm rumors are marked with new descriptors: IDK'_I and IDC'_I. We assume that rc_K and rc_C collide in a new sower S'_a. We denote their paths from *I* to S'_a by l_{rc_K} and l_{rc_C}. When S'_a recovers the IP address of S_b from rc_K and rc_C, it directly contacts S_b to forward rc_K and rc_C attached with IDK'_I and IDC'_I via a TCP link, as shown in Fig. 4. The rc_K and rc_C are then delivered along the reversed paths of l'_K and l'_C until they reach *R*.

E. File Delivery

After recovering the confirm message from $(rc)_{K_R^+}$ using its private key K_R^-, *R* implements a digital envelope technique to encrypt the file into cipher C_f. The purpose of this technique is to reduce the cryptographic overhead. Instead of including C_f into the rumor generation, *R* encrypts $<IDK'_I, IDC'_I, IP_{S'_a}>$ to generate the data cipher rumor and the data key rumor, and attaches the digital envelop payload to the data cipher rumor. The large data cipher rumor and the small data key rumor first take random walks to meet each other at a sower S'_b, then traverse the path from S'_b to S'_a via a TCP connection, and eventually reach *I* along the reversed paths of l_{rc_K} and l_{rc_C}. Upon receiving the digital envelop, *I* recovers the desired file using its private key. For large-size files, responders can split them into multiple segments.

F. Multiple Rumor Riding

Previous works employ multiple walkers, say *k*-walkers, to shorten the query delay time. After *L* hops, *k*-walkers should cover approximately the same amount of peers as a one-walker covers after $k \times L$ hops, while the response time can be significantly reduced. To accelerate the query cycle, in RR, an initiator can issue multiple rumors in the query cycle. We denote this scheme as (*i*, *j*)-RR, which issues *i* cipher rumors and *j* key rumors.

G. Rumor TTL

The selection of rumor TTL, together with the number of cipher and key rumors, determines 1) how many sowers a query would have, and 2) how the sowers are distributed. The tradeoff with this is that for each query, RR requires a least number of sowers randomly distributed in the entire system, but too many sowers will lead to a greater overhead.

To assign a proper TTL, RR employs simple schemes similar to the one used in [24]. Peers periodically insert several pairs of 'testing' rumors into the system, including the initiator's IP. Any sower recovers this testing query sends the retained TTL of the two rumors directly to the initiator through a TCP connection. During this process, we do not require anonymity. The initiator observes the ID (IP address) of the responding sower and the distribution of the reported TTL. If the diversity of sowers is poor or the median of TTLs is low, the initiator enlarges the TTL or adjusts the number of rumors. It decreases the TTL value or the number of rumors when there are too many sowers.

H. Rumor Cache

Each peer needs to cache a number of received rumors before the rumors are matched. Due to space limitations, it is unlikely to provide infinite space. Therefore, rumor removal policies are necessary for each peer. Since key rumors and cipher rumors are in different sizes, and key rumors are typically shorter than cipher rumors (especially during content deliveries), RR allocates three caches in each peer for 1) key rumors, 2) all cipher rumors except the data cipher rumors, and 3) the data cipher rumors. RR assigns a time-duration and starts a timer for each rumor. An FIFO method is employed in the three caches; RR always drops the 'oldest' rumor in the queue until there is enough free space to store the newly arrived rumor. In addition, a peer can drop cipher rumors 'older' than 2 minutes, as most of the queries have the response time less than 1 minute, according to many observations.

IV. Discussion

In this section, we focus on several key issues in the RR design. We first present a theoretical analysis on how to ensure that each query has at the least one sower and that the sowers are evenly distributed over the system. We then discuss the attack models and analyze the anonymity degree of RR.

A. Sower Distribution and Collision Rate

RR rumors are sent in random directions, and each peer forwards a rumor to one of its neighbors without any bias. According to the observations in [19], random walk achieves statistical properties similar to independent sampling for every reasonable network. Studies in [11, 21] show that if the walk length is sufficiently large, the final receivers of a random walk query are randomly distributed. We define *collision distance*, *T*, as the hop count of the shorter path that rumors walk from the initiator to the sower, as illustrated in Fig. 5. When taking the *T* to be $O(\log n)$, the sowers are evenly distributed guaranteed by

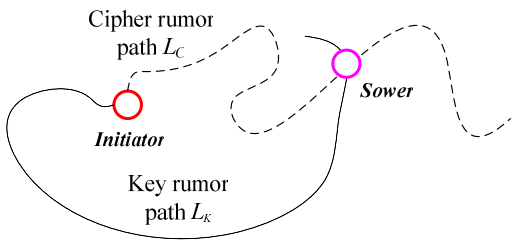

Figure 5 Collision distance $T = \text{Min}\{L_C, L_K\}$.

the observations in [21]. Our simulation results in Section V show that carefully selecting parameters will lead to desired collision distances. We assume that each peer accessed by a rumor is an independent sample from a space of uniform distribution.

A peer becomes a sower if it receives a pair of rumors. For a (i, j)-RR scheme, there are i cipher rumors and j key rumors. Without loss of generality, we assume that each rumor has a fixed TTL value of L. After rumor spreading, the popularity of cipher rumors is $i \times L$, where popularity means the total number of peers receiving the cipher rumor. We further assume that those nodes are distinct with each other and the distribution of them is uniformly random. On the key rumor path, the probability of a peer only being visited by this key rumor and not having the cipher rumor is $(1 - i \times L / n)$. The probability of a key rumor terminating its walk without hitting a cipher rumor is given by $(1 - i \times L / n)^L$. Thus, the probability of a successful collision in a (i, j)-RR is given by:

$$p_h = 1 - (1 - i \times L / n)^{j \times L} \quad (1)$$

We also introduce a parameter τ to indicate the minimum acceptable collision rate. The expected collision rate is formulated subject to the constraint: $p_h \geq 1 - \tau$, $0 < \tau << 1$. Combining this with (1) we have

$$j \times L \times \log(1 - i \times L / n) \geq \log(\tau) \quad (2)$$

To keep the collision rate p_h at a high level, say 99%, peers can choose the proper i, j, and L according to (1) and (2). We calculate three typical distributions in (1, 1), (1, k), and (k, k)-RR schemes to examine the optimal setting of i and j. Based on (2), we show a theoretical distribution of p_h in Fig. 6, Fig. 7, and Fig. 8. The network size is one million nodes. Indeed, the (k, k)-RR scheme achieves a higher collision rate than others in most cases. Thus, we assert that the (k, k)-RR scheme is a proper choice for a high collision rate. From the results, it is clear that larger TTL values of rumors corresponds with a higher collision rate, while increasing the number of rumors also leads to a higher collision rate. We use the above results to guide the implementation of the RR protocol.

It must be admitted that the above result merely considers the ideal case. In practice, there might not be $i \times L$ distinct nodes on the cipher rumor routes. Consequently, fewer sowers exist than estimated. Therefore, the values of i, j, and L obtained from (1) and (2) are the lower bounds in uniformly random topologies. A node can increase the values to guarantee enough sowers. On the other hand, due to the small world property of P2P networks, the collision rate in real P2P overlays is significantly higher than that in uniformly random topologies, as demonstrated in simulations in Section V.

As we mentioned, the collision distance is another important factor balancing the tradeoff between user anonymity and the query delay. Normally, initiators wish the sower peers to be as far away as possible since the sowers recover query messages and might help adversaries to locate the initiator if they are compromised. Thus, the number of rumors should be limited. We also show our simulation results for the upper bound of rumor numbers in a practical network in Section V.

B. Anonymity Analysis

In this subsection, we first discuss the degree of anonymity that RR achieves, and then analyze the protocol effectiveness under various attack scenarios.

1) Anonymity Degree

We first define the *Anonymity Degree* (AD) as the probability of making an incorrect guess to identify a participant. A higher degree signifies that better anonymity has been achieved.

In RR, when an intermediate node receives a query or confirm rumor, it forwards the rumor to a randomly chosen neighbor. From an observer's perspective, each node sending the rumor could be the actual initiator of the rumor. Similarly, when an intermediate node receives a response or data rumor, any intermediate node delivering the rumor could be a potential receiver. Therefore, an observer (initiator, responder or an intermediate node) cannot distinguish the initiator and responder from the other peers. Thus, if the number of nodes in the P2P system is n, the initiator's or responder's AD is $(n - 2) / (n - 1)$ from the viewpoint of a normal observer (the number of potential initiators/responders is n - 1).

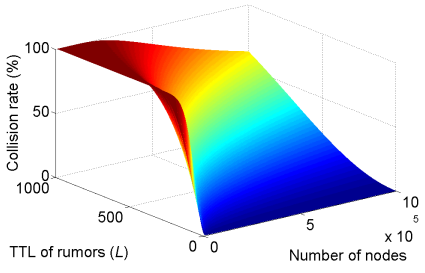

Figure 6 Collision rate in (1, 1)-RR.

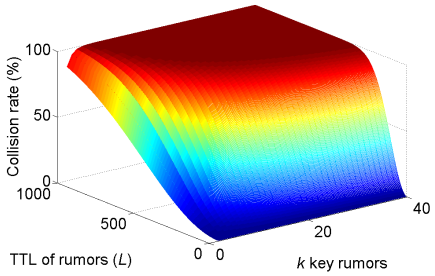

Figure 7 Collision rate in (1, k)-RR.

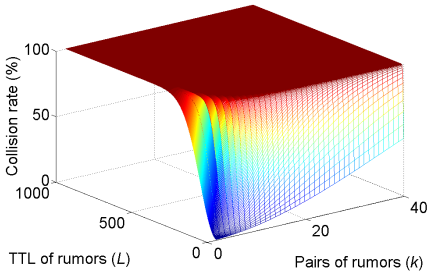

Figure 8 Collision rate in (k, k)-RR.

2) Attacks

We assume that the number of adversary nodes is m. Thus, the probability of a peer being an adversary is m / n. In some cases, adversaries may merely observe the fact that a sender is sending information, without any knowledge about the transmitted data. We claim that the protocol achieves unlinkability to the initiator and responder if they cannot be identified when communicating with each other. In our attack model, we assume that based on the records, the adversary nodes are able to observe and store the communication traversing them and guess the identity of nodes that initiated those transmissions. Adversary nodes also have the capability to perform active attacks which include dropping, hijacking, and forging packets, controlling flows and connections of the network, etc.

We categorize the major attacks a P2P anonymity protocol should be able to defend against, and discuss why RR is invulnerable.

Message coding attack: a passive observer can trace a message in the network if the message does not change its coding. This attack is also the main motivation to perform encryptions in anonymous designs. For example, a naive anonymous system can simply let queries randomly walk in the system before reach a node which chooses to act as an agent to flood the query. Such a plaintext random forwarding design can achieve initiator anonymity to some extent. The fresh node, however, would lose its anonymity immediately if it sends its first query to an observer. Similarly, if an attacker (statistically) tracks signal messages from a sender to a receiver [10], the anonymity of such a plaintext random forwarding protocol vanishes. RR provides unlinkability to fresh nodes, such that when observers obtain a rumor, it cannot link a query to the received rumor.

Local collaborating attack: neighboring adversaries may collaborate to monitor the traffic passing through and share the information in order to identify the possible neighboring initiators. When two adversaries neighboring the initiator receive a pair of rumors of a message, one of them may forward the key rumor to another. The latter will recover the message, and guess that the node sending the rumor is the initiator.

In RR, a sower selects a subset of its neighbors to send the plaintext query, and the two collaborating nodes will not receive the query. In this way, adversaries only bet that the monitored node is an initiator or a responder. Thus, the AD of the initiator or responder becomes $1 - 1 / (1 + s)$, where s is the number of sowers of this pair of rumors. Suppose an initiator is neighboring c local collaborating nodes. If c exceeds 2, then the AD becomes $1 - 1 / (1 + s \times (1 - p)^{(c-2)})$, where p is the probability of a sower choosing a neighbor to send the query. Hence, RR is not subject to the local collaborating attack if the adversaries cannot compromise more than three neighbors of the monitored node.

Timing attack: in a timing attack [15], the adversary deduces the correlation between the timings of packets, such as the response time of a query, the time difference of a query, a time interval between two sequential packets, etc., to locate a transmission. Timing attacks pose a serious threat to path-based approaches. RR is invulnerable in that (1) rumors are delivered over the overlay network in a random walk manner and RTT measurements do not reveal the real distance to the responder; (2) if adversaries want to trace the rumor via the time difference to locate the responder, they need to trace one query rumor from initiator to a sower, then trace the plaintext query message from the sower to the responder, which is too cumbersome to be practical; and (3) a sower issues a request only after it obtains a pair of query rumors, so the response time is mainly dependent on the random walks of rumors, which are unpredictable. All of these factors make it difficult to launch a timing attack.

Predecessor attack: in some anonymous systems, an initiator repeatedly communicates to a specific responder in many rounds. Adversaries are able to identify the path pointed to the responder in each round, and log any node that sends a message to this path. In this case, the initiator is most likely the one which appears more [17]. The fundamental assumption in such an attack is that an initiator always communicates with a specific responder in the long run. A variation is *passive logging attacks*[12]. In RR, rumors correlating to a message walk randomly and interact with random sowers unpredictably. Based on our earlier discussion, the sowers of a given initiator or responder are not unique and randomly distributed over the system. Hence, adversaries are not able to perform such an attack to identify the initiator or responder via sowers, and RR is not subject to this type of attack.

Traffic analysis attack: an adversary can extract traffic flow information such as packet count, message volume, and communication pattern, etc., and build correlations between the initiator, responder, and their communication. Similar to timing attacks, traffic analysis attacks can compromise the initiator's or responder's anonymity if adversaries control a large fraction of the network. For example, based on traffic shaping [22] adversaries clog traffic in the suspected nodes and observe the traffic change when they slightly mitigate the clogging traffic. Thus, the real traffic can be deduced. Performing this attack consequentially along the reversed path of the traffic, adversaries can easily determine the initiator. RR is much less vulnerable to this attack since subsequent messages do not belong to the same traffic, and there are not any continuous paths in RR.

Traceback attack: adversaries start from a known sower to trace back to the initiator along the rumor paths. The adversary examines the stored routing state of the peers to identify the paths between the initiator and responder. We consider the users' anonymity in two attack scenarios: (1) One-way back tracking: adversaries that are on the rumor path back-track and collaborate with each other to detect the source node of this rumor; (2) Multiple-ways back tracking: at least one adversary intercepts both the cipher rumor and the key rumor.

Theorem: The probability that collaborating attackers correctly guess the initiator of one captured rumor is less than $(m + 1) / n$, $1 \le m < n$, where n is the number of total peers and m

is the number of adversaries in the system.

Proof: Let E_k ($k \geq 1$) denote the event that the first adversary occupies the kth position on the path, where the initiator occupies the 0th position. We let I denote the event that an adversary is on the rumor path immediately after the initiator. We also define $E_{k+} = E_k \vee E_{k+1} \vee E_{k+2} \vee \ldots \vee E_L$, where $1 \leq k \leq L$. Clearly, $\Pr[I \mid E_1] = 1$. Therefore, we calculate $\Pr[I \mid E_{1+}]$, the probability that adversaries successfully determine that a cipher rumor is coming from the initiator as follows.

For adversaries:

$$\Pr[E_i] = \left(\frac{n-m}{n}\right)^{i-1} \frac{m}{n},$$

$$\Pr[E_{2+}] = \frac{m}{n}\sum_{k=1}^{\infty}\left(\frac{n-m}{n}\right)^{k} = \frac{m}{n}\left(\frac{\frac{n-m}{n}}{1-\frac{n-m}{n}}\right) = \frac{n-m}{n},$$

$$\textit{and} \qquad \Pr[E_{1+}] = \frac{m}{n}\sum_{k=0}^{\infty}\left(\frac{n-m}{n}\right)^{k} = 1. \tag{3}$$

Note that we extend the path length L of rumors to infinity in order to maximize the impact of attacks. Obviously, we also get $\Pr[E_1] = m / n$, $\Pr[I \mid E_1] = 1$, $\Pr[I \mid E_{2+}] = 1 / (n - m)$. The last equation is derived from the observation that if the first adversary does not immediately follow the initiator, it can only guess the initiator of the message with a probability of $1 / (n - m)$. We have

$$\Pr[I] = \Pr[E_1] \Pr[I \mid E_1] + \Pr[E_{2+}] \Pr[I \mid E_{2+}]. \tag{4}$$

Substituting the above equations with those of (3), we calculate $\Pr[I \mid E_{1+}]$ by:

$$\Pr[I \mid E_{1+}] = \frac{\Pr[I \wedge E_{1+}]}{\Pr[E_{1+}]} \leq \frac{\Pr[I]}{\Pr[E_{1+}]} = \frac{m+1}{n} \blacksquare \tag{5}$$

When the number of adversaries, m, approaches n, the probability of an adversary correctly guessing the sender's identity approaches 1. The AD of an initiator or responder is $1 - (m + 1) / n$, $1 \leq m < n$.

Analogously, the probability that the multiple-ways back tracking attacks can correctly guess that a peer is the initiator of suspected rumors is given by:

$$p_m = \frac{(m+1)^2 \cdot L^2 \cdot i \cdot j \cdot m}{n^5}, 1 \leq i, j \leq \frac{n}{L}, 1 \leq m \leq n-1 \tag{6}$$

Thus the AD of the initiator and the responder is $1 - p_m$ in this scenario.

In summary, when there are no global adversaries or active tracebacks, RR achieves a high degree of anonymity. RR is vulnerable if the attackers have a global knowledge or full control of the network. In this scenario, most anonymous approaches fail. It is noticeable that path-based approaches have to depend on the anonymous proxies or routers to achieve the promised anonymity. With the increase of the network size, it is very difficult for individual peers to maintain a large number of anonymous agents in path-based approaches. In contrast, RR takes advantage of using the entire P2P network to provide anonymity. Thus, the anonymity set, which including all equivalent probability to be a certain initiator or responder, is maximized into the size of $(n - m)$.

V. Simulation

Additional latency of data delivery, bandwidth consumed by anonymous traffic, and crypto processing, if they exist, are necessary in order to provide anonymity. In this section, we evaluate the RR design through trace-driven simulations.

A. Metrics

We use the following metrics for evaluating RR.

Collision rate. To verify the theoretical results in Section IV, we examine the distribution of collision rate with real traces. Besides the verification, we also use the results to guide the selection of rumor parameters.

Collision distance. A longer collision distance means a higher anonymity level, but also increases the delay of a query as well as the traffic overhead. On the other hand, the collision distance must be sufficiently large to guarantee sower diversity, as we discussed in Section IV.

Number of sowers. We are also concerned with the number of sowers in a query cycle. Since each sower implements a flooding search for an initiator, too many sowers will incur a large number of replicated query messages, and too few sowers will result in failure to provide enough redundancy and reliability.

Traffic overhead. The amount of traffic overhead represents the comprehensive latency in data delivery and bandwidth. Specifically, we are more interested in the extra traffic overhead caused by anonymous components. We assume that a query cycle involves e edges in the P2P overlay. For each edge in the P2P overlay, there is a unique path mapped into the physical internet layer with the length l. For each message enrolled in one query cycle, we calculate the sum of the distances that this message passes through. Therefore, the traffic overhead of a query cycle is defined as $C = M \times L = \sum |m_i| \times l_i$, $1 \leq i \leq e$, where $|m_i|$ is the size of the traversed messages.

Crypto latency. It is the overhead incurred by the main cryptographic algorithms, 128-bit AES and 1024-bit RSA, in this protocol. We investigate the cryptographic overhead compared with other anonymity protocols. We use the processing overhead in one AES operation as the basic unit to make conversions between RSA and AES. Thus we can investigate the comprehensive cryptographic overhead incurred by different algorithms.

Response time. In P2P systems, it is defined as the time elapsed from when a query is issued to when the first response arrives. In our simulation, the response time is defined as the time from the start of rumor spreading to the time when the initiator receives the first response message.

B. Methodology

The P2P topologies come from two sources. One is based on

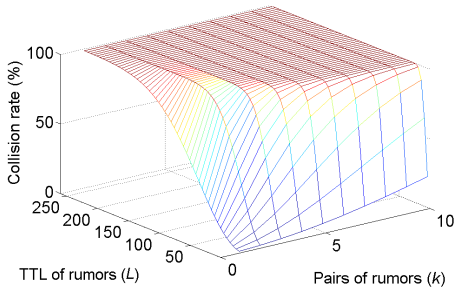

Figure 9 Theoretical collision rate.

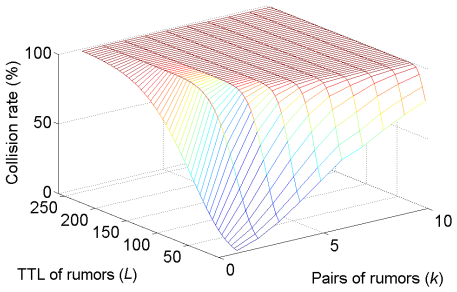

Figure 10 Collision rate of simulation.

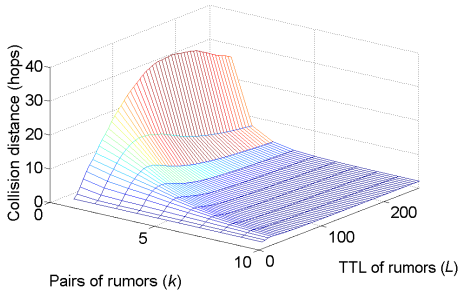

Figure 11 Collision distance.

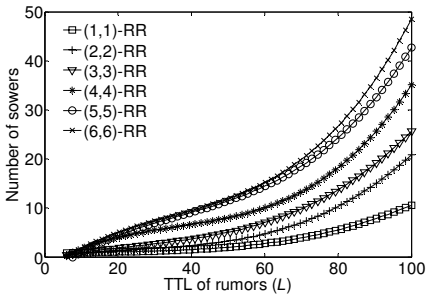

Figure 12 Number of sowers.

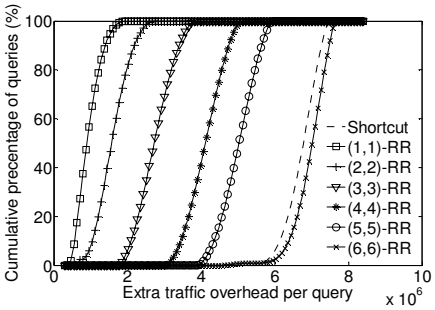

Figure 13 Cumulative distribution of traffic overhead.

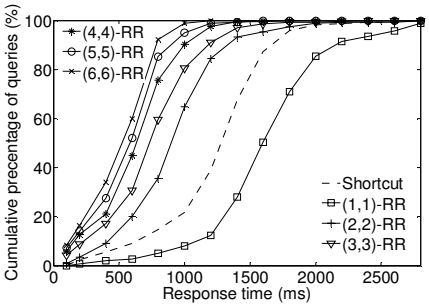

Figure 14 Cumulative distribution of response time.

the DSS Clip2 trace, which collected log data from Jan 2001 to Jun 2001. The other one is a more recent snapshot kit of Ion P2P [23], which logged data from Sep 2004 to Feb 2005, including topologies with high degree nodes (*i.e.*, maintaining more than 30 neighbors). When adopting Ion's traces into the simulated topologies, we only use the ultrapeers of its snapshots, which perform the flooding search in a hybrid Gnutella. Our simulations ran over different traces, ranging from 10^3 to 10^5 nodes. To simulate the physical internet layer below the P2P overlay [18], we used BRITE [5] to generate 30,000 – 100,000 nodes in the internet-like topologies. Content popularity of a publisher follows a Zipf-like distribution (aka Power Law) [3], where the relative probability of a request for the ith most popular page is proportional to $1/i^{\alpha}$, with α typically taking on some value less than one.

To perform the security algorithms used in RR protocol, we employ Crypto++, a software kit which provides standard cryptographic functions. Our experiments for simulation and implementation are both conducted on several desktop PCs, typically with Pentium M 3.2G CPU, 1GBytes memory, 40G hard disk, and 10/100M Ethernet card. We also simulate the dynamic properties of the P2P overlay network by assigning a lifecycle to each peer. The lifetime is generated according to the distribution observed in [8]. The mean of the distribution is chosen to be 600 seconds [9, 16]. The value of each peer's lifecycle is decreased by one with each passing second. When peers use up their lifetimes at the end of each second, they leave the system the following second, and other fresh peers selected from the physical internet layer join in as replacements.

C. *Results*

We first consider the collision rate of a single rumor spreading. To verify the theoretical result discussed in subsection IV.A, we simulate rumor spreading procedures in the traces with a (k, k)-RR scheme. We experiment in the sample space of rumor numbers $k \in [1..10]$ and path length $L \in [1..256]$ (the default TTL value in Gnutella is 7). The average results of the collision rates are presented in Fig. 10. It is observed that the collision rates are usually higher than they are in the theoretical results, which are shown in Fig. 9. Since the topology in Gnutella networks follows small-world properties, a random path in the P2P topology often traverses high-degree nodes, causing the collision rates to be higher than they are in homogeneous networks, in which the node degree follows a uniform distribution. This phenomenon is particularly obvious in the dense topologies of the Ion's traces. Combining the results of theoretical and real experiments, we obtain that the proper lower bound of the number of rumors k and the TTL value of each rumor L is $k \times L \geq 100$. We use this result to guide the setting of rumors in our protocol.

As discussed in Section IV, the collision distance is important in balancing the tradeoff between user anonymity and the query delay in the RR design. In the results of the (k, k)-RR scheme plotted in Fig. 11, it is shown that if L is larger than 25 ($1 \leq k \leq 10$), the average collision distance is no less than 5. On the other hand, a small k, say less than 6, guarantees that the most collision distance will be larger than 5. While k exceeds 6, the collision distance tends to be constrained within 2~5 hops. Considering the fact that anonymity is more important than latency, we suggest that the number of rumors k should be kept to a maximum

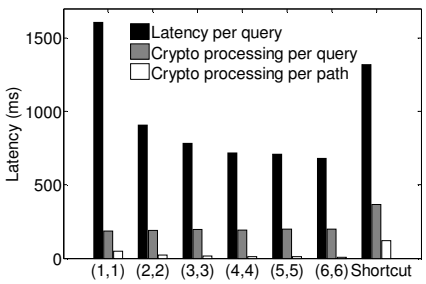

Figure 15 Components of latency.

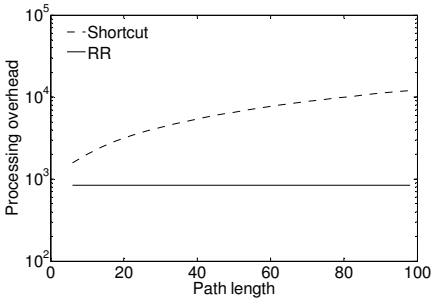

Figure 16 Average cryptographic overhead of the initiator and responder for one query.

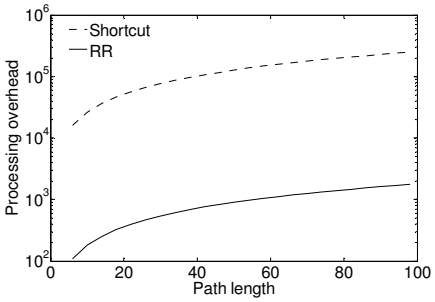

Figure 17 Cryptographic overhead of intermediate nodes for one query.

of 6. Besides the collision distance, we are also concerned with sower diversity. Ideally, sowers should be uniformly distributed over the entire system so that for a given peer, distinct sowers are generated in different RR executions. We use the *distinct sower ratio* (*D*) to evaluate the diversity. If a peer performs *g* rounds of RR, and generates *d* distinct sowers ($d \leq g$), the distinct sower ratio of this peer is given by $D = (d / g) \times 100\%$. By repeatedly running RR for each node in its lifecycle, we see that when *L* is larger than 30 and $k \in [1..6]$, the ratio *D* is larger than 92%. Therefore, selecting $L > 30$ and $1 \leq k \leq 6$ can effectively guarantee a safe collision distance as well as the random distribution of sowers.

In the meantime, RR needs to limit the number of sowers in order to avoid a large number of replicated query messages. As shown in Fig. 12, when we select $k \times L \leq 200$, each (*k*, *k*)-RR scheme has no more than 10 sowers ($1 \leq k \leq 6$). Therefore, $k \times L$ should be in the range [100, 200] in order to meet both the reliability and the scalability requirements.

We then consider the traffic overhead. We compare RR with the most recent work, Shortcut protocol [13]. We insert 10,000 queries into the system, and Fig. 13 plots the cumulative distribution of the extra traffic overhead of (*k*, *k*)-RR schemes. In our experiments, the TTL value of rumors is constrained by $k \times L = 150$. Note that a larger *L* means more traffic overhead. We observe that the average traffic overhead incurred by the Shortcut protocol is little lower than that of the (6, 6)-RR scheme, which is the maximum value of our suggested settings. Except for this case, the traffic overhead of RR is much smaller than that of the Shortcut protocol.

Users of current P2P systems often have rigid requirements for the response time for requesting resources. We show the cumulative distribution curves of response times in different (*k*, *k*)-RR schemes in Fig. 14, comparing them with those of the Shortcut protocol. Clearly, the average response latency is decreased when we increase the number of rumors, *k*. However, more rumors incur more traffic overhead and message replications. Careful selection of the RR protocol settings will lead to the reduction of both the traffic overhead and response time compared with those of the Shortcut protocol.

Compared to the previous approaches, such as Shortcut and APFS, the most significant advantage of RR is that the cryptographic processing overhead has been cut down tremendously. This feature also results in a low latency. Figure 15 contrasts the average latency per query and the part caused by cryptographic processing. In addition, we also compare the average latency of cryptographic processing along the rumor paths. We observe that in most cases, the time spent on cryptographic processing in an onion path is over 10 times higher than the time spent in a rumor spreading path. This is due to the fact that Shortcut's peers must perform a large number of RSA operations for both anonymous path construction and anonymous relaying, which incur significantly longer responses than those of RR.

We also examine the cryptographic overhead of the RR protocol. Figures 16 and 17 show the average cryptographic overhead of RR and the Shortcut protocol in a query cycle. We can see that RR significantly reduces the cryptographic overhead for initiators, responders, and intermediate nodes.

The total cryptographic overhead of the intermediate nodes is linearly proportional to the length of the onion path when using the Shortcut protocol. Also, the onion routing technique leads to an overtly high cryptographic overhead for Shortcut users. With paths of similar lengths, RR gains a large decrease of cryptographic overhead compared with path-based approaches due to the usage of symmetric encryption/decryption instead of asymmetric key-based algorithms. Hence, the light cryptographic overhead strongly supports RR's implementation in large-scale P2P systems.

VI. Implementation Experience

We implemented a RR prototype on the Window XP platform. We used the Crypto++ Library to implement all built-in cryptographic algorithms. The modification to the Gnutella prototype protocol comprises the following components:

(1) Sower peers require direct TCP links to forward the rumors. For TCP forwarding, we took advantage of Windows Sockets.

(2) RR uses AES for cryptographic operations. In each rumor packet, the payload includes a cipher text generated by using the AES (CBC mode). The AES key size is 128-bit.

(3) To guarantee the quality of AES keys, RR generates the keys using the random number generator function in the Crypto++ Library.

(4) Previous works [7] show that each node generates 0.3 queries per minute on average, most queries' lengths are below 100 bytes, and the average number of neighbors of the traces are below 60. After calculation, we set the size of the local key rumor, cipher rumor, and data cipher rumor caches as 1MB, 2MB, and 10MB, respectively. The time-duration for cipher rumors is 2 minutes, and the time-duration is 10 minutes for key rumors. The size of the file fragment is 512K bytes.

We examined the throughput and the latency of RR. Table I presents the latency and throughput of a peer to perform cryptographic operations, including AES key generation, AES operation, *CRC* calculation, and RSA operation. In RR, the throughput of an initiator query depends on the rumor generation speed, which is determined by the AES key generation, AES encryption, and *CRC* calculation. Among them, the slowest algorithm is the AES key generation, which can provide 14,221 keys per second on average.

TABLE I LATENCY AND THROUGHPUT OF ALGORITHMS

ALGORITHMS	THROUGHPUT (Mbytes/s)
128-bit AES key generation	0.217±0.00443
128-bit AES Encryption	8.155±0.256
CRC-32 calculation	137.48±4.79
1024-bit RSA Encryption	0.148±0.00280
1024-bit RSA Decryption	0.00670±0.000126

VII. CONCLUSION

Existing anonymity approaches are mainly path-based. Peers have to recruit middle nodes and construct paths before transmissions. The overhead of maintaining and updating the paths is also significantly high. In this paper, we propose a lightweight and non path-based mutual anonymity protocol for unstructured P2P systems, Rumor Riding (RR). Employing a random walk concept, RR issues key rumors and cipher rumors separately, and expects that they meet in several random peers. The results of extensive trace-driven simulations show that RR provides a high degree of anonymity and outperforms existing approaches in traffic overhead and processing latency. We also discuss how RR can effectively defend against popular attacks. The early experience of our prototype implementation shows its practicality.

Future and ongoing work includes accelerating the query speed, introducing mimic traffic to confuse attackers, and optimizing the *k* and *L* combination to further reduce the traffic overhead. We will also investigate other security properties of RR, such as the unlinkability, information leakage, and failure tolerance when facing different attacks. It would also be interesting to explore the possibility of implementing this lightweight protocol in other distributed systems, such as grid systems and ad-hoc networks.

ACKNOWLEDGMENT

The authors would like to thank the shepherd, Bobby Bhattacharjee, for his constructive feedback and valuable input. This work is supported in part by the Hong Kong RGC grants HKUST 6264/04E and HKUST 6152/06E.

REFERENCES

[1] D. Chaum, "Untraceable electronic mail return addresses, and digital pseudonyms", Communications of the ACM, 1981.

[2] M. K. Reiter and A. D. Rubin, "Crowds: anonymity for web transactions", ACM Transactions on Information and System Security, 1998.

[3] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: evidence and implications", In Proceedings of IEEE INFOCOM, 1999.

[4] D. Goldschlag, M. Reed, and P. Syverson, "Onion routing", Communications of the ACM, 1999.

[5] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: an approach to universal topology generation", In Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS), 2001.

[6] V. Scarlata, B. N. Levine, and C. Shields, "Responder anonymity and anonymous Peer-to-Peer file sharing", In Proceedings of IEEE ICNP, 2001.

[7] K. Sripanidkulchai, "The popularity of Gnutella queries and its implications on scalability", In Proceedings of The O'Reilly Peer-to-Peer and Web Services Conference, 2001.

[8] S. Saroiu, P. Gummadi, and S. Gribble, "A measurement study of Peer-to-Peer file sharing systems", In Proceedings of Multimedia Computing and Networking (MMCN), 2002.

[9] S. Sen and J. Wang, "Analyzing Peer-to-Peer traffic across large networks", In Proceedings of ACM SIGCOMM Internet Measurement Workshop, 2002.

[10] R. Sherwood, B. Bhattacharjee, and A. Srinivasan, "P^5: A protocol for scalable anonymous communication", In Proceedings of IEEE Symposium on Security and Privacy, 2002.

[11] I. Abraham and D. Malkhi, "Probabilistic quorums for dynamic systems", In Proceedings of DISC, 2003.

[12] M. Wright, M. Adler, B. N. Levine, and C. Shields, "Defending anonymous communications against passive logging attacks", In Proceedings of IEEE Symposium on Security and Privacy, 2003.

[13] L. Xiao, Z. Xu, and X. Zhang, "Low-cost and reliable mutual anonymity protocols in Peer-to-Peer networks", IEEE Transactions on Parallel and Distributed Systems, 2003.

[14] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the second-generation onion router", In Proceedings of the USENIX Security Symposium, 2004.

[15] B. N. Levine, M. K. Reiter, C. Wang, and M. Wright, "Timing attacks in low-latency Mix systems", In Proceedings of the International Conference on Financial Cryptography, 2004.

[16] Y. Liu, X. Liu, L. Xiao, L. M. Ni, and X. Zhang, "Location-aware topology matching in P2P systems", In Proceedings of IEEE INFOCOM, 2004.

[17] M. K. Wright, M. Adler, B. N. Levine, and C. Shields, "The predecessor attack: an analysis of a threat to anonymous communications systems", ACM Transactions on Information and System Security (TISSEC), 2004.

[18] X. Zhang, Q. Zhang, Z. Zhang, G. Song, and W. Zhu, "A construction of locality-aware overlay network: mOverlay and its performance", IEEE JSAC Special Issue on Recent Advances on Service Overlay Networks, 2004.

[19] N. Bisnik and A. Abouzeid, "Modeling and analysis of random walk search algorithms in P2P networks", In Proceedings of HOT-P2P, 2005.

[20] D. Figueiredo, J. Shapiro, and D. Towsley, "Incentives to promote availability in Peer-to-Peer anonymity systems", In Proceedings of IEEE ICNP, 2005.

[21] R. Morselli, B. Bhattacharjee, A. Srinivasan, and M. A. Marsh, "Efficient lookup on unstructured topologies", In Proceedings of ACM PODC, 2005.

[22] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor", In Proceedings of IEEE Symposium on Security and Privacy, 2005.

[23] D. Stutzbach and R. Rejaie, "Characterizing the two-tier Gnutella topology", In Proceedings of ACM SIGMETRICS, 2005.

[24] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "SybilGuard: defending against Sybil Attacks via social networks", In Proceedings of ACM SIGCOMM, 2006.