A *k*-Anonymous Communication Protocol for Overlay Networks *

Pan Wang
Dept. of Electrical and Computer Engineering
NC State University
Raleigh, NC 27695
pwang3@ncsu.edu

Peng Ning
Dept. of Computer Science
NC State University
Raleigh, NC 27695
pning@ncsu.edu

Douglas S. Reeves
Depts. of Computer Science & Electrical and Computer Engineering
NC State University
Raleigh, NC 27695
reeves@ncsu.edu

ABSTRACT

Anonymity is increasingly important for network applications concerning about censorship and privacy. The existing anonymous communication protocols generally stem from mixnet and DC-net. They either cannot provide provable anonymity or suffer from transmission collision. In this paper, we introduce a novel approach which takes advantage of hierarchical ring structure and mix technique. This proposed protocol is collision free and provides provable k-anonymity for both the sender and the recipient, even if a polynomial time adversary can eavesdrop all network traffic and control a fraction of participants. Furthermore, it can hide the sender and the recipient from each other and thus can be used for anonymous file sharing. The analysis shows the proposed protocol is secure against various attacks. Measurements further demonstrate it is practical.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Network communications*

General Terms

Security

Keywords

Overlay Networks, Security, Anonymity

1. INTRODUCTION

The privacy of communication has become a critical issue in the Internet. Encryption protects the content of communication, but does not conceal the fact that two users are communicating. In many situations, users may wish to make their communication anonymous. For instance, a customer placing an online order may not want his/her transactions to be traced. As another example, if the item ordered by this person can be delivered electronically (e.g., an electronic book or a digital movie), he/she may not want his/her destination address (e.g., the email account, the IP address of his/her computer) to be identified.

In the past two decades, a number of anonymous communication protocols (e.g., [5, 7, 8, 11, 14, 17, 23, 26, 32, 33]) have been proposed. Most of them originate from Chaum's two seminal approaches: mixnet [11] and DC-net [12]. The mixnet family protocols (e.g., [14, 18, 27, 32, 33]) use a set of "mix" servers that shuffle the received packets to make the communication path (including the sender and the recipient) ambiguous. They rely on the statistical properties of background traffic that is also referred to as the *cover traffic* to achieve the desired anonymity, and cannot provide provable anonymity. The DC-net family protocols (e.g., [5, 12, 16, 17]) utilize secure multi-party computation techniques. They provide provable anonymity without relying on trusted third parties. However, they suffer from the transmission collision problem that does not have a practical solution [17].

In this paper, we develop a simple and scalable anonymous communication protocol. It provides provable $k-$anonymity to both the sender and the recipient without transmission collision. That is the sender and the recipient are indistinguishable from the other $k-1$ participants, where k is a predetermined parameter that can be any number between 1 and $\mathcal{N}$ (the number of participants in the network).

In the proposed protocol, the participants, which are referred to as the *nodes*, are organized into a set of logical rings and form an overlay network over the Internet. Within each ring, an anonymous transmission mechanism, which is the cornerstone of the proposed protocol, uses message batching and one-way key chain to make a node's message indistinguishable. It ensures (i) a node can send messages to any other node (in its ring) without disclosing identity to all nodes in the network, and (ii) a node is prevented from maliciously modifying or replaying any transmitted message in a ring.

A sender utilizes the above anonymous transmission mechanism to anonymously communicate with a recipient that

*This material is based upon work partially supported through the U.S. Army Research Office under the Cyber-TA Research Grant No. W911NF-06-1-0316.

ASIACCS'07, March 20-22, 2007, Singapore.

may be located in a different ring. That is the sender conceals the ID of destination ring, in which the recipient resides, into its outgoing message. It sends the message to a randomly chosen (agent) node in its local ring, following the above anonymous transmission mechanism. This agent node extracts the message and forwards it to the corresponding destination ring without knowing the sender's identity. The forwarded message is locally broadcasted in the destination ring, i.e., sent to all member nodes. The recipient thus receives the message without disclosing its identity. If each ring has at least k honest nodes, the proposed protocol, therefore, provides provable $k-$anonymity for both the sender and the recipient.

The analysis shows the proposed protocol is secure under a strong adversary model, in which the adversary controls a fraction of nodes, is able to eavesdrop all network traffic and maliciously modify/replay the transmitted messages. We have completed a proof-of-concept implementation of the basic communication module and tested it on PlanetLab [2]. Our results demonstrate the proposed protocol is practical.

The rest of the paper is organized as follows. Section 2 describes the nomenclature and cryptographic tools used in this paper, and provides an overview of the related work. Section 3 introduces the system and threat models and some major notation used in the paper. Section 4 presents the proposed $k-$anonymous communication protocol in detail. Section 5 provides the anonymity, security and performance analysis. Finally, section 6 concludes this paper and points out some future research directions.

2. BACKGROUND

In this section, we first briefly describe the nomenclature of anonymity defined by previous work, then introduce some cryptographic tools used in the paper, and finally give a brief overview of related work.

2.1 Nomenclature of Anonymity

The concept of *anonymity* in information management has been discussed in previous work [28, 30, 33, 37]. Three types of anonymity or anonymous communication properties were defined: *sender anonymity*, *recipient anonymity* and *relationship anonymity*. We recall the definitions in [28]. *Sender anonymity* means that a particular message is not linkable to any sender and no message is linkable to a particular sender. *Recipient anonymity* similarly means that a message cannot be linked to any recipient and that no message is linkable to a recipient. *Relationship anonymity* is a weak property; it means that the sender and the recipient cannot be identified as communicating with each other, though it may be clear they are participating in some communication. The above anonymities are also referred to as the *full anonymities*, since they guarantee that an adversary cannot infer anything about the sender, the recipient, or the communication relationship from a transmitted message.

k-Anonymity is a weaker guarantee of anonymity, if compared with full anonymity. Ahn, Bortz and Hopper [5] defined it as the property that an adversary was able to learn something about the sender or the recipient of a particular message, but could not narrow down its search to a set of less than k participants. In other words, k-anonymity guarantees that the adversary is not able to guess the sender or the recipient of a particular message with a probability non-negligibly greater than $1/k$. Ahn, Bortz and Hopper further defined the *sender (recipient) k-anonymity* as the property that the sender (recipient) of a transmitted message is indistinguishable from at least other $k-1$ honest participants.

2.2 Cryptographic Tools

One-way key chain [24] is a chain of cryptographic keys generated by repeatedly applying a one-way (hash) function $\mathcal{H}()$ to a random number (key chain seed). For example, to construct a key chain of size L, the user randomly chooses a key chain seed K_L, and then computes $K_{L-1} = \mathcal{H}(K_L)$, $K_{L-2} = \mathcal{H}(K_{L-1})$, ..., until $K_0 = \mathcal{H}(K_1)$. K_0 generally is referred to as the *commitment* of the key chain. Due to the one-way property of the function $\mathcal{H}()$, given a disclosed k_i, it is computationally infeasible to compute any undisclosed k_j for $j > i$. However, a user can compute any K_j for $j < i$ efficiently, i.e., $K_j = \mathcal{H}^{(i-j)}(K_i)$. In the proposed protocol, we use one-way key chains to authenticate the origins of transmitted messages (in the message batch). The order of keys in a key chain, therefore, represents the order of a node's messages and thus prevents the replay attacks effectively. For convenience, we refer to the keys in a key chain as the *key-chain keys*, in order to distinguish them from the encryption keys used by the proposed protocol.

We consider two types of encryptions in our proposed protocol: symmetric key encryption and public key encryption. In symmetric key encryption, two hosts share a common secret key that is used for both encryption and decryption. In public key encryption, a host publishes its public key which is used for encryption by any other node, and secretly keeps the private key which is used for decryption. We require the *semantic security* for both encryptions. That is, the encryption is randomized, and a polynomial time adversary cannot distinguish the encryptions of two input messages.

2.3 Previous Work

As mentioned earlier, most anonymous communication protocols are stemmed from Chaum's two seminal approaches: mixnet [11] and DC-net [12]. In mixnet, a sender encrypts an outgoing message and the ID of recipient using the public key of a trusted server, called a "mix". The mix accumulates a batch of encrypted messages, decrypts and reorders these messages, and forwards them to the recipients. An eavesdropper cannot link a decrypted output message with a specific (encrypted) input message. Mixnet thus protects the secrecy of users' communication relationships. To deal with the possibility of compromising the single mix, mixnet has been extended in [18, 23, 29, 32], where a set of mix servers are used. Recently, Möller presented a provably secure public-key encryption algorithm for mixnet [26]. This algorithm has been adopted by Mixminion [14]. Since mixnet-like protocols rely on the statistical properties of background traffic, they cannot provide provable anonymity. Wright et al. showed the degradation of anonymity of some protocols in the face of persistence attackers [38]. Our protocol is mixnet-like; however, it removes the requirement of trusted servers and provides provable anonymity.

DC-net [12, 37] is an anonymous broadcast protocol employing secure multiparty computation. It provides provable anonymity in the absence of trusted servers. But it is vulnerable to the transmission collision problem. That is, two (or more) players may transmit in the same message slot and thus no message will be delivered successfully, even if all players are honest. There is no practical solution

to solve such a transmission collision problem [17]. Furthermore, this protocol has poor scalability, as it requires $O(N^3)$ protocol messages for each transmitted anonymous message in a network of N users. Pfitzmann and Waidner suggested to implement superposed sending of DC-net on a physical ring network to reduce the communication overhead [30]. It is important to note that the ring is used for improving efficiency and reliability instead of the anonymous message delivery directly. Recently, Ahn et al. extended DC-net to achieve $k-$anonymity for message transmission [5]. Golle and Juels presented new DC-net constructions to detect cheating with high probability [17]. Compared with DC-net family protocols, our protocol avoids the problem of transmission collision, and it is more efficient (i.e., $O(n^2)$ protocol messages per anonymous message, where n is the number of nodes in the ring).

Crowds is designed for anonymous web browsing, i.e., hiding the sender from the recipient [33]. However, it does not hide the recipient and the packet content from the nodes en route. Hordes is an extension of Crowds, intending to reduce packet processing at members [35]. Compared with both Crowds and Hordes, our protocol provides stronger anonymous protection. It not only hides the sender from the recipient, but also hides both of them from the other nodes in the network.

Beimel and Dolev designed a family of anonymous message delivery protocols, i.e., the BUS system, simulating the public transportation system in digital world [7]. BUS protocol suffers from the huge size of transmitted message, i.e., $O(N^2)$. Furthermore, the "seat" in the "bus" discloses the sender's identity to the recipient. The authors suggested to use random seating and clusters to reduce the number of seats and hide senders from recipients. However, it makes the recipient unable to reply, and information may lose due to seating collision. Compared with the BUS system, our protocol is more efficient, collision-free, and supports the detection of active adversaries.

3. MODELS AND NOTATION

We assume the participating nodes voluntarily cooperate with each other to provide an anonymizing service. All nodes are potential originators of anonymous communication. Each node has a unique ID. For simplicity, we assume a node's ID is its IP address, and we will not distinguish between the node's ID with its IP address in the rest of the paper. Each node has a public/private key pair authenticated by a trusted authority. Two nodes can authenticate each other and establish a secure and reliable end-to-end connection by employing, e.g., IPsec [20, 21, 22]. For simplicity, we assume the nodes' public keys have a unform size, e.g., 1024 bits.

We assume the adversary can eavesdrop all network traffic. The adversary may also control a fraction of nodes in the network, and it may modify/replay/drop a transmitted message. The objective of the adversary may be to identify the sender/recipient of a given message, or trace the end-to-end communication relationship.

Table 1 shows the major notation used throughout the paper.

Table 1: Notation

S_{ID}	session ID
B_{ID}	message batch ID
PK_A	the public key of node A
$PK_A\{M\}$	encryption of message M using public key PK_A
K_{A_i}	the i^{th} secret key generated by node A
$K_{A_i}\{M\}$	encryption of message M using secret key K_{A_i}
$K_{A_i}^{-1}\{M\}$	decryption of message M using secret key K_{A_i}
M_{A_i}	the i^{th} message sent by node A
MAC_{A_i}	message authentication code computed using secret key K_{A_i}
R_{AB}	one-way key chain generated by node A and assigned to node B
R_{AB_i}	the i^{th} key-chain key in key chain R_{AB}
$\mathcal{H}()$	the function used to generate one-way key chain
$f()$	a secure one-way function whose output is 1024-bit long
KF_i	key chain update flag with value i, where $i = 0$ means the flag is not set
CK	commitment of key chain update. Contains a node ID and a commitment of new key chain if KF is set. Otherwise, it is a random value
RF	a receiver flag
N_i	a random number

4. THE PROPOSED PROTOCOL

In the proposed protocol, the participating nodes are organized with a collection of logical ring structures. Within each ring, a transmission mechanism using message batching ensures a node can anonymously send messages to any other member node without disclosing its identity. To anonymously communicate with a recipient that may reside in another ring, a sender follows the transmission mechanism and sends its messages to a randomly selected agent node in the local ring. These messages contain an ID of the destination ring in which the recipient resides. If the destination ring is the local ring, i.e., the sender and the recipient are in the same ring, the agent node broadcasts the messages locally, i.e., to all member nodes in the ring. Otherwise, it forwards the messages to a node in the destination ring. The latter broadcasts the received messages locally. The recipient thus receives the messages without disclosing its identity. If each ring has at least k honest nodes, as we will show through our analysis, the proposed protocol will provide provable k-anonymity protection to both the sender and the recipient.

In the following subsections, we will present the proposed protocol in detail. We first introduce its network topology. Next, we describe the transmission mechanism that allows a node to anonymously send messages to other member node in the local ring. After that, we show the solution that is based on the above transmission mechanism and allows arbitrary two nodes to communicate anonymously. Finally, we discuss the update of key chains used by the nodes to anonymously identify the origin and the order of their messages in the proposed protocol.

4.1 Network Topology

As illustrated by figure 1, the proposed protocol adopts a hierarchical topology used in many peer-to-peer systems (or protocols), e.g., KaZaa [4], Gnutella v0.6 [3] and Herbivore [16], to organize the network. That is the participating nodes are classified into a set of small subgroups. Since the nodes in each subgroup are further organized with a logical ring structure, each subgroup is also referred to as a ring in this paper. Each ring is uniquely identified by a ring ID.

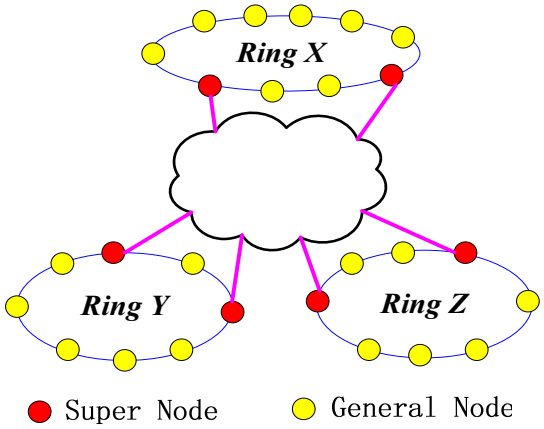

Figure 1: An illustration of network topology of the proposed protocol.

It has at least ϑ nodes, where $\vartheta > k$ and k is a predetermined parameter representing the least degree of anonymity provided by the ring. We will discuss how to determine the value of ϑ later.

For convenience, we classify the member nodes of a ring into two categories, *regular node* and *super node*. A regular node is a node that has no direct connection to the nodes in other rings, and it relies on the local super node(s) to forward packets across rings. A super node is one that provides the message forwarding service across rings. It also has the responsibility for local ring management, such as accepting new nodes. Each ring may have multiple super nodes in order to avoid a single point of failure.

It is worth noting that (i) we adopt the *admission rule* used in many peer-to-peer systems to group the nodes, i.e., the lower v bits of the hash value of a member node's ID should be equal to those of the ring ID, where v is the number of bits used for representing the ring ID. It ensures that the rings are approximately the same size, and also prevents attackers from crowding out all but one honest node in a targeted ring. (ii) Even though many super node selection mechanisms e.g., [10, 25, 36], have been presented, we suggest adopting the election mechanism in [10], since it ensures all nodes have equal probability to be elected as the super node by using Byzantine fault-tolerant method when less than 1/3 nodes are malicious. And (iii) the super node should inform all nodes (in the local ring) about the ring configuration information, e.g., the list of member nodes, their positions in the ring and their public keys, and other parameters, including the current session ID, uniform message size, and encryption algorithms. A *session* is defined by the event of ring structure changing, e.g., nodes joining/leaving or the position changing of a node.

4.2 Anonymous Transmission in a Local Ring

The anonymous transmission mechanism in a local ring is the cornerstone of the proposed protocol. It effectively hides the sender of an anonymous message (from the other nodes including the super node forwarding this message). In this subsection, we present the detail of this transmission mechanism: the transmission initialization and the message transmission along the ring, and the detection of adversaries that do not follow this transmission mechanism. We defer the solution that allows arbitrary two nodes to anonymously exchange messages to the next subsection.

4.2.1 Transmission Initialization

A ring needs a two-stage initialization once a new session is triggered. The purpose of such an initialization is two-fold. First, the nodes need to set up a message batch that contains exactly one message from each node in the ring without disclosing the sender's identity of each message. A node's message is thus mixed with the other nodes' messages in the message batch. Second, each node secretly assigns one unique one-way key chain to each of the other nodes in the ring without disclosing its identity. These key chains are used to identify the order of received messages and thus prevent the message replay attacks. They also allows a node to verify whether the received message batch contains exactly one message from each node in the ring. Our later analysis will show these key chains will not affect the anonymity provided by the proposed protocol.

After the two-stage initialization, a node can send data to any other node in the ring anonymously by replacing its old message in the received message batch. Figures 2 and 3 illustrate the procedure of transmission initialization.

Stage One: Message Batch Construction To start the initialization, a specific super node that is referred to as the *starting node* (e.g., the super node A in figure 2) sends an initialization packet (P_{A1}) to its next hop in the clockwise direction of the ring (referred to as the *ring direction*). This initialization packet is signed by node A. It consists of a n-slot key vector and an encrypted dummy message, where n is the number of nodes in the ring. Each slot conceals a secret key, a session ID and a message authentication code. For instance, the first slot is $PK_0\{K_1, S_{ID}, MAC_1\}$ and the i^{th} slot is $K_{i-1}\{...K_1\{PK_i\{K_i, S_{ID}, MAC_i\}\}...\}$. The dummy message is used to for two purposes: (i) to construct the message batch, and (ii) to anonymously assign each member node a unique one-way key chain used in later anonymous communication. It has a standard size and a strict format, i.e., a n-slot key vector plus the anonymizing payload, where the elements in a key vector slot of the dummy message are different from those in the key vector slot of the initialization packet. Such standard message size and message format are two public parameters of the ring and are applied to all transmitted messages in the ring. We will discuss the message format and how the key chains are distributed later.

Once receiving the initialization packet from the starting node, the immediate downstream node (e.g., node B in figure 2) uses its private key to decrypt the first key vector slot. It thus gets the secret key, session ID and message authentication code concealed in this slot. The node then checks the packet signature, decrypted session ID, and message authentication code to ensure the correctness and integrity of the packet, while the message authentication code is calculated based on the concealed secret key and the rest part of the packet. If the checking result proves correct, the node removes the signature and the first key vector slot, and it uses the previously resulting secret key to decrypt the remaining slots and the dummy message. After that, the node sends the changed initialization packet and a new one generated by itself to the next hop at a random order. These two initialization packets are digitally signed by the node, and they have the same number of key vector slots (i.e., $n-1$) and the packet size.

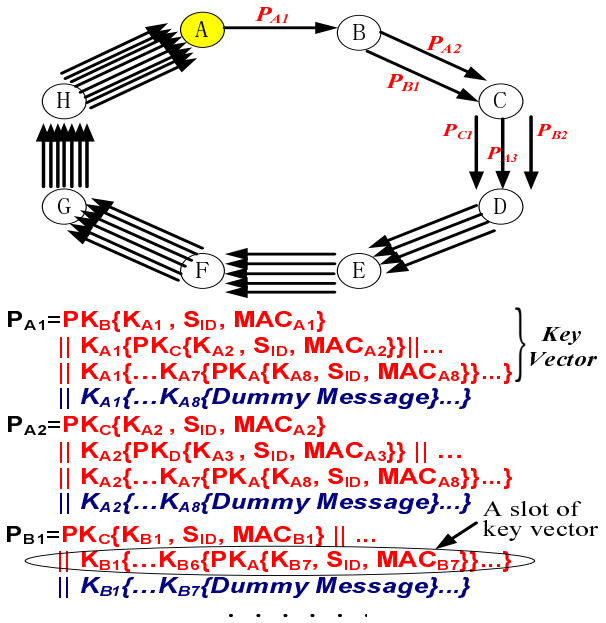

Figure 2: An illustration of message batching. The sender precalculates the MACs in a reversed order, e.g., MAC_{A8} first, then MAC_{A7}, and so on. MAC in a key vector slot is calculated based on the secret key concealed in this slot and the rest part of the packet assuming this slot becomes the first slot, e.g., MAC_{A2} is computed based on K_{A2} and $K_{A2}\{PK_D\{K_{A3}, S_{ID}, MAC_A3\}\}..||K_{A2}\{..\{DummyMessage\}..\}$. The digital signature is not shown in the figure.

The other member nodes in the ring follow the same steps, i.e., after receiving d initialization packets from the previous hop, decrypt each of them, remove the first key vector slot, and then forward $d+1$ initialization packets to the next hop at a random order, where d is the number of hops between a node and the starting node. As a result, the number of transmitted initialization packets increase by one at each hop, while the size of packet decreases gradually. Finally, the starting node receives n dummy messages that have uniform size. These n dummy messages form the message batch.

Detection of Packet Modification: A node may get a negative result when it checks the MAC of a received initialization packet. The attacker can be the previous hop node who modifies the packet or the packet's original sender who intently conceals a wrong MAC. To detect the attacker, the receiver, e.g., node B in figure 2, immediately reports an error about the received packet, i.e., broadcasting the received initialization packet and the decrypted K_{A1}, S_{ID} and MAC_{A1}. The other nodes can verify the error report because (i) P_{A1} is signed by node A and thus is non-repudiated, and (ii) the encryption of (broadcast) K_{A1}, S_{ID} and MAC_{A1} using node B's public key should equal the first key vector slot of P_{A1}. Node B's previous hop (node A) then needs to prove it forwarded the packet correctly, i.e., to show a received packet whose forwarding result should be P_{A1} and the decryption of the first key vector of this packet. If node A cannot show the corresponding packet, it is identified as the attacker. Otherwise, the detection continues until one upstream node cannot prove it forwarded the packet correctly. This node is identified as the attacker. The nodes revoke the attacker from the ring and restart the transmission initialization.

LEMMA 1. *A polynomial time adversary cannot link an honest node with a specific (received) initialization packet or reversely, if there are more than one received initialization packets originated from honest nodes.*

PROOF. (Sketch) Suppose an honest node A receives l initialization packets while only one of them (M_0) originated from an honest node (suppose node B). Node A decrypts M_0 with a secret key K concealed in the first key vector slot of M_0. The resulting new packet is denoted as M_1. Node A sends M_1 and its own initialization packet M_2 (at a random order) to the next hop. Since K is encrypted using node A's public key, it is computationally infeasible for a polynomial time adversary $\mathcal{A}$ to calculate K from M_0. Without knowing K, $\mathcal{A}$ cannot link M_0 with M_1 with a probability significantly better than 1/2 (i.e., random guessing), recalling the semantic security of symmetric key encryption. In other words, $\mathcal{A}$ cannot link node A with M_2 or link node B with M_1 with a probability significantly better than 1/2, even $\mathcal{A}$ can link node B with M_0. Similarly, if there are r received initialization packets originated from different honest nodes, $\mathcal{A}$ cannot successfully identify the sender of each (honest) initialization packet with a probability significantly greater than $1/r$. □

Stage Two: Message Batch Checking and Key Chain Distribution As an intermediate node may maliciously replace the received initialization packets in stage one, the nodes need to verify whether the resulting message batch contains exactly one dummy message from each node in the ring. Therefore, the nodes need to assign a unique key chain to each member node secretly in stage two. These key chains are used to identify the origin and the order of the nodes' messages without disclosing the latter's identities. They not only prevent the potential replay attacks in later anonymous communication but also allow a node to verify whether a received message batch contains exactly one message from each member node in the ring. Figure 3 illustrates the procedure of message batch checking and key chain distribution by showing the propagation of node E's dummy message without losing universality.

The starting node (e.g., node A in figure 3) forwards the message batch (formed at stage one) to the next hop, while each message is digitally signed by the starting node. As stated earlier, a dummy message and the later transmitted anonymous data message consist of a n-slot key vector and the fixed-lengthed anonymizing payload. Each key vector slot has a standard size decided by the key length of the employed public key encryption algorithm. As illustrated in figure 3, each slot contains a receiver ID, a session ID (S_{ID}), a recipient flag (RF), a key chain update flag (KF), a new key-chain key (R_n) in the assigned key chain, a key chain update commitment (CK), a one-time secret key (K_i), a random number (N_i), and a message authentication code (MAC_i).

When a node receives a message in the message batch, e.g., node B receives M_{E1}, it decrypts the first key vector slot and checks the integrity of M_{E1} based on the value of each field in the slot. If the checking result proves negative, node B starts the detection of modification as described in

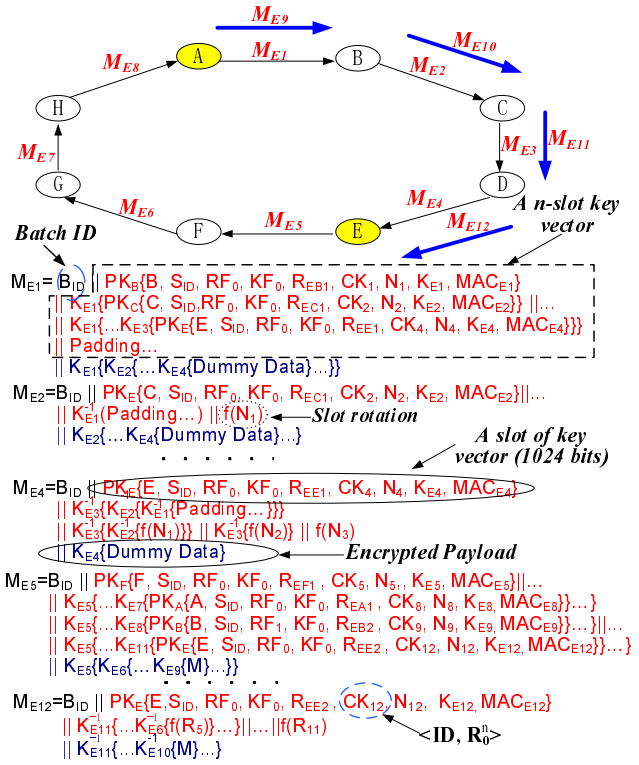

Figure 3: An illustration of message batch checking and key chain distribution. We assume M is encrypted data. The digital signature of each message is not shown in the figure. Clearly, the sender precalculates the MACs in a reversed order, e.g., in message M_{E5}, node E computes MAC_{E12} first, then MAC_{E11}, and so on.

stage one. Otherwise, node B continues processing M_{E1}. From the receiver flag RF, node B knows whether it is the targeted receiver. As the transmission initialization is not finished yet, the RF flag should not be set, i.e., node B is not the receiver. Node B also knows it is not the original sender of M_{E1} since the key-chain key R_{EB1} is not the expected one (i.e., R_{BB1}) generated by itself, without knowing the sender's identity as proved in Lemma 1.

Node B buffers R_{EB1} (node E thus secretly assigns a key chain R_{EB} to node B without disclosing its identity). It uses K_{E1} to decrypt the rest key vector slots and the encrypted payload, replaces the first key vector slot with $f(N_1)$, rotates the slots one step leftwards (in order to keep the message size unchanged), removes the previous signature, and signs the changed message (i.e., M_{E2}).

Node B processes the other $n-1$ received messages similarly and forwards the message batch (i.e., the processed messages) to the next hop at a random order. It is worth noting that one of the n received messages must originate from node B itself, i.e., the decrypted key-chain key should equal R_{BB1}. Node B replaces this message with a new one. If there is no such a message, it means node B's previous initialization packet was maliciously replaced by some node in the ring. Node B should stop processing the message batch and immediately start the detection of packet replacement which will be discussed later.

After that, node B generates a commitment of assigned key chains. This commitment is signed by node B. It contains the current session ID, the current batch ID, a batch counter whose value is set to 0, and the n decrypted key-chain keys $\{R_{AB1}, R_{BB1}, ..., R_{HB1}\}$. Node B broadcasts the commitment locally. Once a node receives node B's commitment, it checks whether this commitment includes the key-chain key it assigns to node B. If true, the node buffers this commitment. Otherwise, it means the node's previous initialization packet/message was replaced by some node. The victim node starts the detection of packet replacement.

Detection of Packet Replacement: The victim node, e.g., node B, asks all nodes in the ring to broadcast their buffered packets/messages. By emulating the propagation of its previous initialization packet/dummy message and comparing the emulation results with the broadcast packets/messages at each hop, node B can locate the adversary. It next reports the adversary by broadcasting its input packet/message at the adversary and the decryption result of the first slot of this input packet/message. For instance, if P_{B1} was maliciously replaced by node C in figure 2, i.e., node C sent P'_{B2} instead of P_{B2} to the next hop, node B broadcasts P_{B1}, K_{EB1}, S_{ID} and MAC_{B1}. The other nodes can verify the authority of the report as presented previously. They calculate the correctly forwarded packet P_{B2} based on P_{B1} and K_{EB1}. If they cannot find P_{B2} in node D's received packets, the nodes revoke node C from the ring and restart the transmission initialization.

We notice that the key chain used by a node to identify its own messages might collide with the one assigned by another node, e.g., $R_{BBi} = R_{EBi}$. Such a collision may cause the sender cannot correctly recognize its own messages and thus break the anonymous communication. We address that the probability of the collision is ignorable according to the birthday paradox, as the length of a key-chain key R_{XY_i} is large ($\geq$160 bits) and the number of nodes in a ring is small (< 100). Furthermore, the sender can use the encryption key or the MAC embedded in the slot to identify its own message in the case of a key chain collision, at a cost of increased storage.

When the starting node (i.e., node A) receives the message batch once more and no adversary is reported, the transmission initialization is done. The resulting message batch contains exactly one message from each node in the ring, and all nodes get the assigned key chain correctly. The nodes flush their previously buffered initialization packets (i.e., they only buffer the most recently received message batch). The stating node forwards the message batch to its next hop. The nodes in the ring thus can start the anonymous data transmission that will be discussed next.

LEMMA 2. *A polynomial time adversary cannot link an assigned key chain to a specific (honest) node or reversely, if more than one messages in the received message batch are originated from honest nodes.*

PROOF. (Sketch) Given an assigned key chain (i.e., a key-chain key R_0) decrypted from a received (honest) dummy message M. Since R_0 (and later key-chain key R_i) does not contain any information about the sender, a polynomial time adversary $\mathcal{A}$ has to use M to identify the corresponding sender. As proved in Lemma 1, $\mathcal{A}$ cannot link M with a specific (honest) node or reversely, if there are more than one received messages originated from honest nodes. Therefore,

$\mathcal{A}$ also cannot link R_0 with a specific (honest) node with a probability significantly better than random guessing. □

4.2.2 Message Transmission along the Ring

In this part, we show how a sender uses the message batch constructed as above to anonymously send a message to a node in the local ring. That is when a node receives the message batch from its previous hop, it first increases the local counter (e.g., t and it is used to record the number of message batch received in current session) by one. The node then decrypts the first key vector slot of each received message using its private key. It next checks whether the hash values of the decrypted key-chain keys (R_{XY_i}) one-to-one match with the previously buffered ones (i.e., whether the message batch contains exactly one message from each node in the ring). For instance, $\mathcal{H}(R_{EB2})$ in figure 3 should equal the previously buffered R_{EB1}. Such a checking also prevents the modification or replay attacks. If the checking result proves negative, it means there is an adversary who either maliciously modified/replayed the transmitted packets or set a trap for the node, e.g., node E in figure 3 encrypts $R^{'}_{EB2}$ instead of the correct R_{EB2} in message M_{E5}. The node starts the detection of adversary.

Detection of Adversary: Given the hash value of decrypted key-chain key R_{XY_i} in a received message M does not match with any (previously) buffered ones, the node broadcasts a report about M. This report contains M and the decryption of $M's$ first key vector slot. The other nodes can verify the authority of the received report, i.e., the decryption is correct based on the node's public key and $\mathcal{H}^t(R_{XY_i})$ should but does not equal any element in the node's (previously broadcasted) key chain commitment. Next, the previous hop node should prove that M is a forwarded form of its received message M^*, i.e., broadcasting a report about M^*. If the previous hop node cannot show a valid report, it means it is the adversary. Otherwise, the detection continues until one upstream node cannot show a valid report. This upstream node is identified as the adversary who is the sender of M, recalling that each key vector slot contains a MAC and thus any malicious modification of a transmitted message will be discovered by the next hop node of the adversary. The nodes revoke the adversary from the ring and start a new session. Similarly, if multiple received messages have the same decrypted key-chain keys, the node broadcasts theses messages and the decryption results of their first key vector slots. The other nodes detect the adversary as just presented, i.e., the upstream nodes need to prove they forwarded the received message batch correctly.

If the hash values of decrypted key-chain keys one-to-one match with the buffer ones, the node updates its buffered key-chain keys with the new ones. It next verifies the integrity of each received message using the decrypted MAC. If the MAC of a message proves incorrect, it means either the message was modified by the previous hop or the message sender sets a trap. The node starts the detection of packet modification as presented above. Otherwise, the node updates its previously buffered messages with the received ones and continues processing the received messages.

For each message in the received message batch, the node knows whether it is the original sender of this message based on the key-chain key R_{XY_i} decrypted from the first key vector slot. If the node is the sender, it replaces the message with a new one as illustrated by M_{E5} in figure 3. Otherwise, the node uses the decrypted secret key (K) to decrypt the rest key vector slots and the encrypted payload. If the RF flag in the first key vector slot is set, the node extracts the payload, as it is the targeted receiver. The node then replaces the first key vector slot with $f(N_i)$, rotates the slots one step leftwards, removes the previous signature, and resigns the changed message. After that, the node sends the processed messages to the next hop at a random order.

LEMMA 3. *Given all nodes in the ring publish their key chain commitments correctly, it is computationally infeasible for an adversary to successfully modify/replay an (honest) node's message. As a result, the message batch contains exactly one message from each node in the ring.*

PROOF. (Sketch) Assume an adversary $\mathcal{A}$ modifies/replays an (honest) node's message $\mathcal{M}_0$ in the received message batch, i.e., it forwards $\widetilde{\mathcal{M}_1}$ instead of the correct one $\mathcal{M}_1$ to the next hop (honest) node $\mathcal{B}$, and node $\mathcal{B}$ accepts $\widetilde{\mathcal{M}_1}$. Due to the one-to-one relationships between the decrypted key-chain keys and the previously buffered ones, the first key vector of $\widetilde{\mathcal{M}_1}$ must contain the key-chain key R_j that matches with previously buffered R_{j-1}. The order of key-chain keys in the one-way key chain ensures the received $\widetilde{\mathcal{M}_1}$ is a new message, i.e., not a replayed one. The one-way property of the one-way function $\mathcal{H}$ makes it computationally infeasible to calculate R_j from R_{j-1}, even if $\mathcal{A}$ knows R_{j-1}. Also, R_j and the other variables are encrypted using $\mathcal{B}$'s public key, it is computationally infeasible for $\mathcal{A}$ to calculate R_j from $\mathcal{M}_1$ without knowing the corresponding private key, recalling the semantic security of the public key encryption. Thus, the first key vector of $\widetilde{\mathcal{M}_1}$ must be same as that in $\mathcal{M}_1$. As this first key vector contains a message authentication code of $\mathcal{M}_1$, the rest key vectors and the encrypted payload in $\widetilde{\mathcal{M}_1}$ must be same as those in $\mathcal{M}_1$, assuming the authentication mechanism is secure. As a result, $\widetilde{\mathcal{M}_1}$ must equal $\mathcal{M}_1$, otherwise, $\mathcal{B}$ will detect the violation of message authentication. This contradicts with our assumption. It, therefore, proves an adversary cannot succeed in message modification and replay. As $\mathcal{B}$ has an assigned key chain from each node in the ring, the received message batch thus contains exactly one message from each node in the ring, if it passes the verification. □

LEMMA 4. *The message replacement made by the sender is computationally indistinguishable, if the ring has more than one honest nodes.*

PROOF. (Sketch) Suppose an honest node S receives a message batch $\mathfrak{B}$ and $\mathfrak{B}$ passes all verifications. As proved in Lemma 2 and Lemma 3, $\mathfrak{B}$ contains exactly one message from each node in the ring, and a polynomial time adversary $\mathcal{A}$ cannot link S with S's input message M_o (in $\mathfrak{B}$) with a probability better than random guessing. Given S replaces M_o with M_n in its forwarded message batch, it is computationally infeasible for $\mathcal{A}$ to tell whether M_n is the decryption of M_o using some unknown secret key K, even if $\mathcal{A}$ may be able to link M_o with M_n as the messages originated from the same sender, assuming the semantic security of symmetric key encryption and the security of one-way function $f()$. Therefore, $\mathcal{A}$ cannot link S with M_n with a probability better than random guessing either. The message replacement made by S is thus computationally indistinguishable. □

4.3 Anonymous Communication between Arbitrary Two Nodes

In previous subsection, we present the mechanism that allows a node to anonymously send messages to another node in the same ring. In this subsection, we show how to use this mechanism to support the anonymous communication between arbitrary two nodes in the network.

To anonymously communicate with a recipient that may reside in a different ring, the sender embeds the ID of the *destination ring* into the anonymizing payload of its message. The destination ring is the ring in which the recipient resides. The sender then anonymously sends this message to a (randomly selected) local super node as presented in previous subsection, i.e., by setting the receiver flag RF in the corresponding key vector slot.

Once receiving the message, the selected local super node extracts the destination ring ID. If the destination ring is the local ring, i.e., the sender and the recipient are in the same ring, the super node broadcasts the message locally. Otherwise, it forwards the message to a super node in the destination ring. The latter broadcasts the received message to all nodes in its ring. The recipient thus receives the message. As the sender's message is indistinguishable from the messages sent by other nodes in the message batch and the local super node that forwards the message is not necessary the sender, the sender is thus hidden among other nodes in its ring. Similarly, the transmitted message is received by all nodes in the recipient's ring, the recipient is thus indistinguishable from the other honest nodes in its ring.

A variant of anonymous communication is anonymous file sharing in which the sender and the recipient not only need to be hidden from the other nodes in the network, but also need to be concealed from each other. The proposed protocol does support such anonymous file sharing, as the sender only needs to know the ID of the ring in which the recipient resides, instead of the recipient's identity. Appendix A briefly discusses the deployment of anonymous file sharing. The proposed protocol also supports the dynamic change of anonymous transmission rate. Please refer to Appendix B for the detail.

4.4 Key Chain Update

A node needs to update its key chain assigned to another node (or itself) and inform all other nodes about the key chain update, once the old key chain runs out. Such a key chain update should be anonymous, i.e., without disclosing the node's identity. A simple solution is that the node requests the local ring to start a new session. However, it is not a good solution as it breaks down other nodes' normal communication. We introduce an alterative solution that allows a node to update its key chains without interrupting others' communication or compromising the anonymity. Figure 4 illustrates this solution where we assume node A wants to update its key chain assigned to node C.

To update the key chain assigned to node C, in its outgoing message M_1, node A sets the flag KF (in the key vector slot that will be decrypted by node C using its private key) to one, sets the KFs in the following slots to two, and fills the corresponding CK fields. Once node C receives M_2, it extracts the new key chain commitment CK from the first key vector. It then broadcasts a new commitment of assigned key chains. This commitment contains the current session ID and batch ID, the value of node's batch counter,

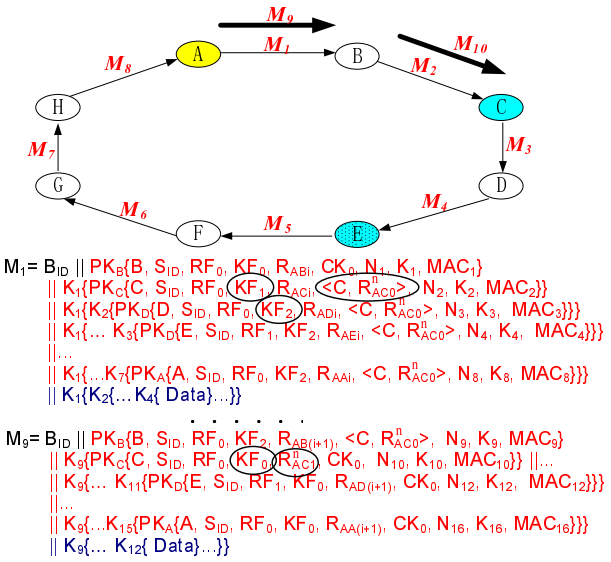

Figure 4: An illustration of key chain update, where node A updates key chain assigned to node C and simultaneously sends data to node E. $R^n_{XY_i}$ denotes a key-chain key from the new key chain.

the $n-1$ key-chain keys decrypted from currently received message batch, and $R^n_{AC_0}$ and the last key-chain key in the old key chain which are specially marked. Node C then forwards the processed message batch to the next hop, noting that node C cannot successfully modify any forwarded message as stated previously.

After receiving the broadcast commitment, a node can verify if the other $n-1$ key-chain keys (except the marked $R^n_{AC_0}$) one-to-one match with the ones in node C's previously broadcast key chain commitment, based on the embedded batch counters in these two commitments, the nodes' positions in the ring, and also its local batch counter. If false, it means node C is an attacker that maliciously broadcasts a wrong commitment for assigned key chains. The nodes revoke node C from the ring and start a new session.

Otherwise, the node buffers this newly received commitment, waits for the coming message batch to verify $R^n_{AC_0}$, and then replaces its buffered node C's (old) commitment with the new one. That is in the coming message batch, the KF flag decrypted from the first key vector slot of a message should equal 2, the ID and the key-chain key in the corresponding CK should equal C and $R^n_{AC_0}$ respectively. Otherwise, either the original sender (i.e., node A) set a trap or node C maliciously broadcasted a wrong commitment, recalling it is computationally infeasible to successfully modify a forwarded message. The node needs to detect and revoke the adversary from the ring.

Detection of Malicious Key Chain Update: If a node (suppose node F) finds the decrypted CK from message M_5 does not match with the newly broadcast commitment (either wrong ID or wrong key-chain key), while the KF flag is set to 2, it broadcasts a report that contains M_5 and the decryption result of M_5's first key vector slot. The other nodes can verify this report and then start the detection of the malicious sender, i.e., the upstream nodes need to prove they forwarded messages correctly as presented previously while

the decrypted KFs in their received messages are equal to 2. Similarly, if the node finds no decrypted KF equals 2, it has to ask its previous hop from which message it should get the decrypted KF equaling 2, and then it starts the detection of malicious sender. For instance, node F in figure 4 finds no decrypted KF equals 2. It sends an inquiry to previous hop node E. The latter points out message M_5, as the decrypted KF in its received message M_4 equals 2. Node F then broadcasts a report that contains the feedback from node E and the decrypted result of the first key vector slot of M_5, showing the decrypted KF in M_5 does not equal 2. The upstream nodes then need to prove they forwarded the message correctly as presented above.

A node X may find the decrypted KF in a received message M equals 2 while the corresponding node Y did not broadcast a updated commitment. In such a scenario, the attacker can be node Y who refuses to broadcast the new commitment or the original sender of M who intently sets a trap, recalling it is computationally infeasible for an adversary to successfully modified a forwarded message. Node X starts the detection of attacker by broadcasting a report that contains M and the decryption result of M's first key vector slot. The upstream nodes then have to prove they forwarded the message correctly. As a result, the nodes in the ring will catch the attacker.

LEMMA 5. *The update of key chains does not help an adversary to identify the sender of a transmitted message, given the ring has more than one honest nodes and each honest node independently updates its key chains assigned to itself and other nodes.*

PROOF. (Sketch) Suppose there are only two honest nodes (A and B) in the ring. Node B broadcasts a updated key chain commitment which is triggered by an honest node. A polynomial time adversary $\mathcal{A}$ receives a data message M (even if the KF flag in M is set to 2). Let $Pr\{Trigger = A\}$ denote the probability that this key chain update is triggered by node A and $Pr\{M = A\}$ denote the probability that $\mathcal{A}$ successfully links M with its sender A. Given node A and node B independently update their key chains, $\mathcal{A}$ will get $Pr\{Trigger = A\} = Pr\{Trigger = B\} = 1/2$, as we have prove $\mathcal{A}$ cannot link node A (or node B) with a specific key chain with a probability significantly better than random guessing. Since an (honest) node's key chain update is independent from $\mathcal{A}$' guess of message sender, we can get $Pr\{M = A\} = Pr\{M = A|Trigger = A\} + Pr\{M = A|Trigger = B\} = 1/2 + 0 = 1/2$ (i.e., random guessing). Similarly, we can prove if there are k honest nodes in the ring and there are multiple rounds of key chain updates, the adversary $\mathcal{A}$ still cannot link an honest node with a specific message or reversely with a probability significantly better than random guessing, given each honest node independently updates its key chains. □

5. ANALYSIS

5.1 Anonymity

THEOREM 1. *If a ring has at least k honest nodes, the proposed protocol provides sender k-anonymity.*

PROOF. (Sketch) As proved in Lemma 3, a transmitted message batch contains exactly one message from each node in the ring, and the one-way key chains secretly assigned by the nodes prevent an adversary from maliciously modifying/replaying a transmitted message in the received message batch. Recall that we use semantically secure public key encryption and secret key encryption. Given a node randomly puts its own message into other $n-1$ forwarded messages, if there are at least other $k-1$ honest nodes, an adversary and the nodes in the local ring (including the local super node that is selected to forward the node's message), cannot distinguish the node's message from the other $k-1$ messages originated from honest nodes. Within a session, a node is able to link two messages transmitted in two different message batches to a same sender, i.e., by using the assigned one-way key chains. However, the node cannot identify the sender, as it cannot link an assigned one-way key chain with a specific node as proved in Lemma 3. Therefore, a node cannot distinguish the sender of a message from the other $k-1$ honest nodes, even it accumulates a set of transmitted message batches. Lemma 4 and Lemma 5 further prove that the adversary cannot identify the sender by observing message replacements and key chain updates. □

THEOREM 2. *If every ring has at least k honest nodes, the proposed protocol provides recipient k-anonymity.*

PROOF. (Sketch) The message is broadcasted locally in the ring where the recipient resides. All nodes in the ring receives the message. Since there are at least k honest nodes in ring, the adversaries cannot distinguish these nodes as the recipients. □

5.1.1 Anonymity Degree k and Ring Size

The value of anonymity degree k is an important parameter in the proposed protocol. In general, k can be any number between 1 and $\mathcal{N}$, while k equals $\mathcal{N}$ means the proposed protocol provides full anonymity. However, as pointed out in [5], 2-anonymity would be enough to cast reasonable doubt, thus invalidating a criminal charge in the United States legal system, and 3-anonymity would be enough to invalidate a civil charge in the absence of other evidence. A small value of k, therefore, is sufficient enough.

The minimum ring size ϑ is determined by the anonymity degree k. It can be $\frac{2k}{1-\beta}$ as presented in [5], where β is the fraction of nodes that are compromised by the adversary. Each ring thus will have at least k honest nodes with a very high probability.

5.2 Security

Adversaries may attempt to trace the senders or the recipients by watching traffic patterns or message encodings. Such attacks are referred to as the traffic analysis attacks. A survey on traffic analysis attacks can be found in [6] and [31]. In the proposed protocol, the nodes' traffic patterns are publicly known, i.e., their messages propagate around the rings. Since a transmitted message batch contains exactly one message from each node in the ring and the messages are re-encrypted hop-by-hop using secret keys extracted from the messages themselves, adversaries cannot link an honest node with a specific message in the message batch or reversely, as shown in Lemmas 2 to 5. Therefore, even given the traffic patten and message encoding algorithm publicly known, adversaries cannot distinguish the sender and the recipient of a transmitted message from the other $k-1$ honest nodes in the local ring.

The adversaries may maliciously modify, replay or tag a transmitted message in order to locate the sender or the recipient. As shown in Lemma 3, the proposed protocol effectively prevents these attacks by employing message preauthentication and one-way key chains. Also, the data payload of a transmitted message is encrypted using a one-time secret key. Therefore, even the same data payload is transmitted multiple times, the adversary cannot link them without knowing the encryption keys.

An adversary may drop some forwarding messages. In the proposed protocol, a message batch contains exactly one message from each node in the ring. A node knows how many messages in a message batch it should receive from its previous hop. If a node drops some forwarding messages, it will be immediately identified by the next hop node. We notice that the adversary who drops a message batch can deny the receipt of message batch. To thwart such an attack, the super node can re-organize the ring structure once the drop of message batch is identified, i.e., by changing the nodes' positions in the ring (e.g., to put the node who denies receipt of message batch as its next hop). If a suspect has been reported by a fraction of nodes (e.g., over 1/3 nodes, considering the Byzantine fault tolerant method only allows less than 1/3 malicious nodes in a group) after the ring re-organizations, this suspect then can be identified as an attacker. If there is a message board system in the network, i.e., all nodes sequentially exact (i.e., receive) the message batches from the board and then put their (forwarded) message batches back to the board, such a message dropping can be prevented completely.

As a node's joining and leaving will interrupt current anonymous communication in the ring and trigger a new round of transmission initialization, an adversary may attempt to destroy the proposed protocol by frequently joining and leaving the network. The proposed protocol cannot prevent such an attack. However, the influence of the attack is localized to a ring, and it is relatively easy to identify the attacker, as the attacker will join in the same ring with high probability due to the node admission rule.

An adversary that is elected as a super node may refuse forwarding messages across the rings and thus block the anonymous communication between the sender and the recipient. It is hardly to detect such an attack if the sender does not have the capability to monitor all network traffic. The proposed protocol employs multiple super nodes in a ring to mitigate such an attack. The sender can try another super node, if it suspects its previous message might not be forwarded by a super node. Therefore, unless the adversary compromises all super nodes in the ring, the sender still be able to communicate with the recipient anonymously. As the adopted selection mechanism of super node [10] ensures all nodes have equal chance to be elected, it will be a high probability of at least one honest super node in the ring.

We notice that a malicious super node may send faulty ring configuration information to each node to ruin the establishment of the ring and the anonymous communication. This attack can be prevented by adopting the Byzantine method discussed in [34] at a cost of increased communication overhead and the constrain that less than 1/3 nodes in a ring can be malicious. Furthermore, a malicious super node may refuse to accept a new node. The proposed protocol cannot prevent such an attack. However, as there are multiple super nodes in a ring, the new node can contact another super node as discussed previously.

5.3 Efficiency

The proposed protocol requires a sender's message to sequently traverse through all nodes in the local ring. It thus increases the communication overhead and the average data latency. Furthermore, the key vector in each transmitted message and the dummy messages transmitted by the nodes in the ring present another sources of communication overhead. In terms of message (or bit) complexity, i.e., the messages (or bits) transmitted in the network for every anonymous message (bit) sent, as stated in previous work [5, 7], the communication overhead of the proposed protocol is $O(n^2)$.

In the proposed protocol, each node has to buffer the latest message batch it received, and save n key chain commitments and n key-chain keys. It also needs to store n one-way key chains that it assigns to itself and other nodes in the ring, if it does not want to calculate the key-chain keys online when sending messages. Jakobsson [19], and Coppersmith and Jakobsson [13] proposed schemes to improve the performance of one-way key chain, which requires $O(log(L))$ storage and $O(log(L))$ computation to access an element, where L is the length of the key chain. The storage overhead at each node, therefore, is $O(n \times log(L))$ measured by the number of stored key-chain keys while ignoring the buffered messages (or $O(n)$ if measured by the number of buffered messages while ignoring the buffering of key chains).

The proposed protocol also imposes heavy computation overhead on each node. A node has to decrypt every received message using its private key. It also has to verify the digital signature of each received message and generate a digital signature for every forwarded message (by signing the whole message batch instead of each individual message, the computation overhead caused by this part can be significantly reduced). As the message batch contains exactly one message from each node in the ring, the computation overhead at each node is $O(n)$, measured by the number of public key operations.

5.4 Implementation Evaluation

We have implemented the basic communication module of the proposed protocol in C on Linux in order to validate the feasibility, assuming the rings have been constructed and the nodes know each other's public key. We adopted RSA public key encryption algorithm with 1024-bit key size and Blowfish secret key encryption in CBC mode with 160-bit key size and 64-bit initialization vector. We did not use one-way key chains in this implementation. The cryptographic operations were implemented by the OpenSSL [1] library and no assembly language was used. We tested this basic component on the PlanetLab [2]. In our tests, the nodes establish TCP connections. The sender and the recipient are located in different rings, where each ring has variant number of nodes. The broadcast is simulated by unicast, i.e., the local super node unicasts the data message to the recipient directly. Table 2 shows the average processing time of a message batch, the estimated maximum bandwidth for anonymous communication, and the average end-to-end communication delay, for 100 rounds of test.

As shown, the proposed protocol provides reasonable performance. Its throughput slightly degrades as the number of nodes in the ring increases. We notice that such a perfor-

Table 2: Processing Time of Message Batch and Latency (Across Rings)

Number of Nodes in the Ring	Payload Size	Message Batch Processing Time†	Estimated Maximum Bandwidth	End-to-End Latency††
4	128Bytes	23.31ms	42.9Kbps	62.39ms
	512Bytes	23.34ms	171.4Kbps	71.09ms
5	128Bytes	29.17ms	34.3Kbps	76.73ms
	512Bytes	29.46ms	135.8Kbps	85.90ms
6	128Bytes	35.12ms	28.5Kbps	97.16ms
	512Bytes	35.14ms	113.8Kbps	104.81ms

† Run on a P4 2.2Ghz machine with 512M memory under Redhat Fedora 2.
†† The sender is one-hop away from its super node, i.e., total 3 hops between the sender and the recipient.

mance is not as good as the one of low-latency anonymous communication services like Freedom [9] or Tarzan [15]. Furthermore, like the previous DC-net family protocols, a node in the proposed protocol has to wait (i.e., until receiving a message batch) before being able to send a data message out, while the previous mixnet-liked protocols allow a sender to independently decide its activity of message sending. They are tradeoffs between performance and anonymity.

6. CONCLUSIONS AND FUTURE WORK

The privacy has become a critical issue in electrical communication. In this paper, we introduce a k-anonymity communication protocol. This protocol is based on the construction of logical rings and the deployment of multicast. It ensures the sender and the recipient are indistinguishable from the other $k-1$ honest nodes and also hides them from each other. Future research will focus on the full implementation and qualitative evaluation of the proposed protocol.

7. REFERENCES

[1] Openssl. `http://www.openssl.org`.

[2] Planetlab. `http://www.planet-lab.org`.

[3] The homepage of Gnutella. `http://gnutella.wego.com`.

[4] The homepage of KaZaa. `http://www.kazza.com`.

[5] L. Ahn, A. Bortz, and N. Hopper. K-anonymous message transmission. In *Proceedings of the 10th ACM conference on Computer and Communications Security*, pages 122–130, Washington D.C., USA, 2003.

[6] A. Back, U. Möller, and A. Stiglic. Traffic Analysis Attacks and Trade-Offs in Anonymity Providing Systems. In *Procings of 4th Information Hiding Workshop*, Pittsburgh, PA, 2001.

[7] A. Beimel and S. Dolev. Buses for Anonymous Message Delivery. *J. Cryptology*, 16:25–39, 2003.

[8] O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: A System for Anonymous and Unobservable Internet Access. *Lecture Notes in Computer Science*, pages 115–129, 2001.

[9] P. Boucher, A. Shostack, and I. Goldberg. Freedom systems 2.0 architecture. White paper, Zero Knowledge Systems, Inc., December 2000.

[10] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance and Proactive Recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, 2002.

[11] D. Chaum. Untraceable Electrical Mail, Retrun Address, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[12] D. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *J. Cryptology*, 1:65–75, 1988.

[13] D. Coppersmith and M. Jakobsson. Almost Optimal Hash Sequence Traversal. In *Proceedings of International Conference on Financial Cryptography*, Southampton, Bermuda, 2002.

[14] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, 2003.

[15] M. Freedman and R. Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, Washington, D.C, USA, 2002.

[16] S. Goel, M. Robson, M. Polte, and E. Sirer. Herbivore: A Scalable and Efficient Protocol for Anonymous Communication. Technical Report 2003-1890, Cornell University, Ithaca, NY, February 2003.

[17] P. Golle and A. Juels. Parallel Mixing. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, Washingto D.C, USA, 2004.

[18] C. Gülcü and G. Tsudik. Mixing email with babel. In *Proceedings of the Symposium on Network and Distributed System Security*, San Diego, CA, 1996.

[19] M. Jakobsson. Fractal Hash Sequence Representation and Traversal. In *Proceedings of the IEEE International Symposium on Information Theory*, Lausanne, Switzerland, 2002.

[20] S. Kent and R. Atkinson. IP Authentication Header. *RFC 2402, IETF*, 1998.

[21] S. Kent and R. Atkinson. IP Encapsulating Security Payload (ESP). *RFC 2406, IETF*, 1998.

[22] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. *RFC 2401, IETF*, 1998.

[23] D. Kesdogan, J. Egner, and R. Büschkes. Stop-and-Go MIXes: Providing Probabilistic Anonymity in an Open System. In *Proceedings of Information Hiding Workshop*, Portland, OR, 1998.

[24] L. Lamport. Password Authentication with Insecure Communication. *Communications of the ACM*, 24(11):770–772, November 1981.

[25] V. Lo, D. Zhou, Y. Liu, and et al. Scalable Supernode Selection in Peer-to-Peer Overlay Networks. In *Proceedings of the 2nd International Workshop on Hot Topics in Peer-to-Peer Systems*, La Jolla, CA, 2005.

[26] B. Möller. Provably Secure Public-Key Encryption for Length-Preserving Chaumian Mixes. *Proceedings of CT-RSA 2003*, LNCS 2612:244–262, April 2003.

[27] U. Möller, L. Cottrell, P. Palfrader, and L. Sassaman. Mixmaster Protocol — Version 2, July 2003.

[28] A. Pfitzmann and M. Köhntopp. Anonymity, Unobservability, and Pseudonymity - A Proposal for Terminology. In *Proceedings of Workshop on Design Issues in Anonymity and Unobservability*, pages 1–9, Berkeley, CA, 2000.

[29] A. Pfitzmann, B. Pfitzmann, and M. Waidner. ISDN-mixes: Untraceable Communication with Very Small Bandwidth Overhead. In *Proceedings of the GI/ITG Conference on Communication in Distributed Systems*, pages 451–463, Mannheim, Germany, 1991.

[30] A. Pfitzmann and M. Waidner. Networks without User Observability. *Computers & Security*, 2(6):158–166, 1987.

[31] J. Raymond. Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems. In *Proceedings of Privacy Enhancing Technologies Workshop*, pages 10–29, Berkeley, CA, 2000.

[32] M. Reed, P. Syverson, and D. Goldschlag. Anonymous Connections and Onion Routing. *IEEE J. on Selected Areas in Coomunications, Special Issue on Copyright and Privacy Protection*, 16(4):482–494, 1998.

[33] M. Reiter and A. Rubin. Crowds: Anonymity for Web Transaction. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[34] R. Rodrigues, B. Liskov, and L. Shrira. The Design of a Robust Peer-to-Peer System. In *Proceedings of the 10th ACM SIGOPS European Workshop*, Saint Emilion, France, 2002.

[35] C. Shields and B. Levine. A Protocol for Anonymous Communication over the Internet. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*, pages 33–42, Athens, Greece, 2000.

[36] I. Stoica, R. Morris, D. Karger, and et. al. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In *Proceedings of ACM SIGCOMM*, pages 149–160, San Diego, CA, 2001.

[37] M. Waidner. Unconditional Sender and Recipient Untraceability in Spite of Active Attacks. *Advances in Cryptology: EUROCRYPT'89*, LNCS 434:302–319, 1989.

[38] M. Wright, M. Adler, B. Levine, and C. Shields. An Analysis of the Degradation of Anonymous Protocols. In *Proceedings of the Network and Distributed Security Symposium*, San Diego, CA, 2002.

APPENDIX

A. ANONYMOUS FILE SHARING

The sender, i.e., file requester, sends its query to a local super node using the anonymous message transmission described above. This query contains the information about the desired file, the sender's ring ID and a random query ID. Once upon receiving the query, the local super node broadcasts this query in its local ring. It also forwards the query to all other super nodes in the network. The other super nodes broadcast the received query in their local rings. As the result, all nodes in the network receive the query. But they cannot link this query with a specific node.

When the recipient, i.e., file provider, receives the query, it sends a reply message to a local super node along the ring anonymously. This reply message includes the sender's ring ID, the query ID, the recipient's ring ID, and a random reply ID generated by the recipient. The super node forwards the reply message to a corresponding super node based on the sender's ring ID in the reply message. The latter broadcasts the reply message in its local ring. The file requester thus receives the (broadcast) reply message without disclosing its identity and knowing the identity of file provider.

B. TRANSMISSION RATE ADJUSTMENT

A node may want to change the transmission rate, once it has an amount of data to transmit or after it finishes the data transmission. The proposed protocol supports such a transmission rate adjustment.

To increase the transmission rate, a node first anonymously sends a request to the local super node that is responsible for transmission initiation. This request contains a random request ID (e.g., r). The anonymous message transmission mechanism in the ring ensures only the node and the corresponding super node know this random ID. Once receiving the request, the super node buffers the request ID r and initiates another message batch, if the number of message batches in the ring does not reach a (predetermined) maximum value. As a result, the transmission rate increases.

Once the node finishes data transmission, it sends another request to the corresponding super node to decrease the transmission rate. This request contains the previous random ID r. After receiving the request, the super node checks whether the received r matches with the previously buffered one. Such a checking prevents other nodes from maliciously decreasing the transmission rate. If the checking result proves correct, the super node broadcasts a message to stop one message batch. The transmission rate thus decreases. The super node can actively decrease the transmission rate, if it finds no data message forwarded/broadcasted in the local ring for a while, e.g., the node may forget to decrease the transmission rate after finishing data transmission. Noting that the node may leave the network abruptly without sending a request to decrease the transmission rate. In this scenario, as the remaining nodes will start a new session due to the change of ring structure, the transmission rate will be reset to the default value in the new session. Analogously, the message size in a ring can also be adjusted dynamically at a cost of increased communication overhead.